NTNU
Norwegian University of
Science and Technology

Explanation-Aware Army Builder for Warhammer 40k

Nenad Zikic

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Anders Kofod-Petersen, IDI

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

Explanation-aware army builder for Warhammer 40k

Nenad Zikic

Master Thesis - Spring Semester 2016

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology

Supervised by Anders-Kofod Petersen

Abstract

The goal of this thesis is to design, implement and test the explanation-aware case base reasoning system for Warhammer 40k as described in the project of the same name.

The system uses object oriented case base reasoning, using JSON as case bases and using general domain knowledge to simulate the Warhammer 40k domain and create armies. Ground work has been laid in for the system to be able to expand its own case base proactively, by learning and simulating armies and battles. The policy was not fully automated due to time and complexity constraints, and manual simulation is a necessary substitute at this time. Explanations are used to to raise confidence in the system and to provide a satisfactory justification of the systems actions. Explanations are also used to instruct new and expert users about the system and the game, both implicitly and explicitly.

The paper fully succeeds in fulfilling two out of the three goals it has set out to do and presents the problems with the domain together with the solutions concerning the uncompleted goal. The thesis follows the scientific method and completes it, developing testable predictions, presenting their results and the methods to replicate them. The paper evaluates and discusses the limitations of the domain and implementation, the contributions of the thesis as a whole and the future work to be done.

Preface

This master thesis was written and developed during the spring semester of 2016, in the Computer Science (Datateknikk) programme of study, at the Norwegian University of Sciency and Technology (NTNU). The thesis has been conducted at NTNU, at the department of Computer and Information Science (IDI), in the Artificial Intelligence (AI) department.

This thesis was supervised by Anders Kofod-Petersen. I would like to extend my thanks to him first and foremost for helping me with the thesis as well as for granting me the opportunity to work on the thesis and topic.

I would also like to extend my thanks to the other professors at the faculty and the department of AI for helping me with the procedure for writing the master thesis, as well as providing motivation and knowledge sources for the thesis.

Finally, I would like to thank my friends, Drikus Kuiper, Martin Andersen, Olve Kroknes and Adrian Johansen Rinde, as well as all the helpful people at the Wartrond gaming club, for their assistance with Warhammer 40k resources, development and testing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis builds upon the work done in the Specialization Project of the same name (Zikic, 2015). The project was focused on the fundamentals of Case-Based Reasoning (CBR) and explanation aware computing, with an extended focus in proactive Artificial Intelligence (AI), or in other words an AI that can maintain and evolve its own case base and strategy. The project represents one half of the Scientific method, and formulates a hypothesis, which is the design for this master thesis. The thesis will focus on the part of the scientific method that was not covered by the project, namely developing testable predictions, gathering test data and evaluating the hypothesis.

Case-Based Reasoning first emerged from the study of human memorization and understanding (Schank, 1982). CBR is a lazy learning method, which means that it does not generalize until a query is made. As such it is particularly suited for a domain such as Warhammer 40k, where the problem domain constantly changes. At its simplest, a CBR system uses previous cases, which are often called solutions, when presented with a problem (query). Since its conception, it has been worked on in many different ways, and many branches of CBR systems have emerged (de Mantaras et al. 2006).

Explanation-aware systems can reason about their actions. One sign of intelligence is the ability to reason about ones own actions (Sormo et al. 2005). Being able to reason about an action means that we have a purpose or a meaning for that action. If a system can reason and explain its actions, it then also has a purpose or a meaning and we can say that we are looking at more than just an algorithm; we are then we are looking at an intelligent system.

As we seek to build artificial intelligence, explanation-aware systems provide a step forward in mimicking human intelligence. Furthermore, explanation-aware systems can be used to teach and instruct novice users, as well as aid expert users (Roth-Berghofer 2004).

The goals of the thesis are presented in Section 1.1, while the research methodology is presented in Section 1.2. Section 1.3 gives a brief overview on the structure of the thesis.

1.1 Goals

This section will present the three goals of the thesis:

Goal 1 Develop and test the CBR system for building an army in Warhammer 40k

The first goal of the thesis is to create and test the underlying CBR system for building an army in Warhammer 40k. The system needs to be able to present to the user a solution army when presented with a problem army, such that the solution army has a good chance of winning. The aim of this goal is to create a system that will win at least 50% of the games played.

Goal 2 Evaluate the usefulness of explanations in the system

The system that we are creating needs to reason about its choices to be called an understanding system (Schank, 1986). To achieve this, we will use explanations. The aim of the explanations is to both reassure, and potentially instruct, novice users and to raise the confidence towards the system from both novice and expert users. The goal is to evaluate the usefulness of the explanations within the domain, so that we can better understand their role and influence in this kind of a system.

Goal 3 Test the application of maintenance policies to the evolution and maintenance of the system within the Warhammer 40k domain

The third and final goal is to create a system that will be able to evolve. Warhammer 40k was created in 1987, and has so far undergone seven different editions, the last was released in 2014, and the final iteration is yet to be completed. Update packages are released regularly for the game, and it is vital that our system is able to analyze new data and integrate it into the system. It is also vital for our system to be able to maintain itself so that it does not reach the utility problem. Furthermore, the metagame, or the best strategy, can change even in between update releases, and therefore our system needs to change as well. The goal with maintenance policies will be to have the system evolve on its own, both in respects to the case base itself, and to the parameters in the system, such as weights. A system that can constantly evolve on its own is a proactive system. This kind of system can think and learn even while the user is away from the system.

1.2 Research Method

From a high-level point of view, this master thesis will complete the scientific method, by focusing on the implementation, experimentation, evaluation and the expansion of the hypothesis presented in the Specialization Project (Zikic, 2015).

From a more low-level approach, we will be using the steps set out by Paul R. Cohen and Adele E. Howe in their paper *How Evaluation Guides AI Research* (1988). We will focus on the three latter stages of evaluation presented in the paper: Building the Program, Designing the Experiments and Analyzing their results. The evaluation and discussion of the work will be presented in Chapter 5.

The two main reasons to follow this methodology is to finish the study already in place, and to be able to evaluate all stages of research. The secondary reason is to have a well documented solution to a unique instance of a problem, which can hopefully be applied to other fields. As a whole, we are not just attempting to create a solution for Warhammer 40k, but rather to use Warhammer 40k as a domain for the overarching problem, which is the division of resources to tackle a defined problem in a complex environment that is subject to change.

1.3 Thesis Structure

It is recommended that this thesis is read in conjunction with the Specialization Project. The focus of the thesis will not be on the background knowledge, but rather on the implementation, testing and evaluation. Some parts of the thesis will expand upon or alter the Specialization Project, and that will be reflected in the thesis. Where appropriate, a summary of the most important parts of the Specialization Project will be presented.

Chapter 2 introduces some of the background theory for the thesis. Chapter 3 focuses on the design, model and implementation of the system. Chapter 4 presents the testing and results of the model and implementation, while Chapter 5 will discuss and evaluate those results. Chapter 6 concludes the project and discusses potential future work that can be done on the system.

Chapter 2

Background Theory

This chapter will introduce the additions to the theory presented in the specialization project (Zikic, 2015). A brief summary of the most important points discussed in the specialization project will be presented in Section 2.1. Section 2.2 will introduce additional theory to army creation in Warhammer 40k, as well as briefly introducing some of the general tactics and heuristics for army creation. Section 2.3 will introduce the Many Are Called/Few Are Chosen (MAC/FAC) retrieval method, which will serve as the retrieval method in the CBR system. The design choice to use the MAC/FAC retrieval is presented in Section 3.2.3.

2.1 Theoretical Summary

Warhammer 40k

Warhammer 40k is a tabletop board game played with two or more people. The game is competitive, involving both strategy and luck, in the form of dice rolling. The players create an army chosen from one of the eighteen factions, and then battle in a set amount of rounds. Achieving objectives scores points, and the player with the highest points at the end of the game is the winner.

The army creation is vital in this process. There is a plethora of units and equipment to create an army with, and a good army can utilize combinations of units, rules and equipment it provides to make sure that it can tackle any obstacle. Units and equipment have a large amount of statistics, special rules and options that can be applied to them and the points are the only limiting factor to creating an army. This is further discussed in Section 2.2.

Case Based Reasoning

Case Based Reasoning is an AI method that utilizes past solutions to solve new problems. It closely mimics the thought processes of humans. It is consistent of four distinct steps: Retrieve, Reuse, Revise and Retain. In the retrieval step a previous solution (called case) is retrieved from the case base. This solution is the closest match to the problem presented to the CBR system. The reuse steps applies changes to provide a better solution to the new problem. The revise step revises the solution with respects to its problem solving capability and the retention step saves the solution as a new case in the case base.

Explanation-Aware Computing

For a system to be called a reasoning system it needs to be capable of explaining its own actions. This can be achieved through explanations. However, explanations can and do differ between different systems and different domains. An explanation can be a justification of the actions that the system took, or it can be the entire process of how the system has performed the actions. It can also be the goals that the system is trying to achieve, or even a combination of all of the types of explanations. Explanations have a set of characteristics that they should follow in order for an explanation to be considered a good explanation. These are fidelity, understandability, sufficiency, low construction overhead and efficiency.

Maintenance Policies

Maintenance policies in CBR systems help maintain the case base, so that it may perform with better efficiency. One of the goals of this thesis and the project was to attempt to utilize maintenance policies so that the system

can not only improve its efficiency through performance, but also through accuracy. This is not only a convenience, but also a necessity, as the domain constantly changes, with new updates and editions released fairly frequently.

Most maintenance policies are reactive and require a user to be present. The project and the thesis introduce a proactive maintenance policy that is designed to be capable of improving the system without user interference. This is done through simulation of the domain and is possible because the domain can be simulated with certainty. This maintenance policy is called the metagame maintenance policy and is used to improve the strategy of the system directly.

2.2 Warhammer 40k

This section will introduce additional theory for army creation and some general tactics and heuristics when creating an army in Warhammer 40k. This is in addition to the theory already presented in the specialization project.

2.2.1 Army Creation

Every unit and piece of equipment in Warhammer 40k has a point cost. Often, the units are presented in squads, which have a point cost, and a set of options which can alter the squad. These options often include the addition or replacement of equipment, the addition of extra units (or models) to the squad, or the replacement of weaker units for stronger units. Every faction, and many of the squads, have more specialized rules as well, allowing for even more variance between units.

When creating an army, it is important to keep the point limit in mind. A 1000 point army is very different from an army consisting of 1500 points. In the same light, an army of 1500 points is not an army of 1000 points with 500 points attached to it. The more points that are present, the more army compositions tend to change. Some of the most common point limits are included in Table 2.1.[1]

[1]These point limits were researched by talking to experts, visiting various tactics-related webpages, such as `http://1d4chan.org/wiki/Warhammer_40,000/Tactics`, as well as Warhammer 40k tournaments.

Point Limit	Description
200 Points	Often called Kill Teams, this point limit is focused on a single squad, or a group of up to four squads
1000 Points	This is often played as a more casual, shorter encounter, or as an introduction to newer players
1500 Points	The lowest point total for tournaments, it often lends itself to longer casual games.
1850 Points	A fairly common point total for many tournaments in the United States and a fairly large point pool for generally serious players. This point limit lends itself to a lot of creativity, as it allows for more varied armies and more expensive singular units
2000 Points	Some of the largest tournament armies. Games usually last an entire day, or at least a large portion of the afternoon
2500+ Points	Fairly uncommon, these games are usually played for fun or to bring out the most expensive units to show off the miniatures

Table 2.1: Typical Point Limits for Warhammer 40k Armies

As we implement our system we need to be aware of these point limits. When retrieving an army from the case base we not only have to pay attention to armies that exceed the point limits, but also to the armies that are significantly below the point limit. Adapting an army that is at the point limit, or very close to it, will be substantially easier and faster than adapting one that is substantially underneath the point limit. Therefore, understanding point limit sizes is very important for our retrieval method.

2.2.2 Tactics and Heuristics for Army Creation

Understanding the point limits is only the first step in creating (and retrieving) a good army in Warhammer 40k. There are some other general rules that we need to adhere to. These rules are taken as an amalgamation from the tactics found on the Internet, the rulebooks, as well as experts, with additional weight being placed on expert responses.

In the specialization project we have discussed Unit Types, and we have split them into Non-Vehicle and Vehicle unit types. Almost all of the armies in

Warhammer 40k fall in between Non-Vehicle, or Infantry focused, and Vehicle focused armies. Good armies often have a mixture of infantry and vehicles, as different equipment is useful against different kinds of units. Knowing and understanding the ratios of these units is vital for an army to succeed.

Battle Forged and Unbound armies were also briefly discussed in the specialization project. An unbound army represents a selection of any squads, without attention given to the squad type. Battle Forged armies have specific detachments that are formed by specific squad types. A combined arms detachment for example, consists of a squad of HQ type and two squads of Troop type. This detachment can be expanded by various other unit types, but all units in the detachment gain some sort of bonus. These bonuses are quite important and players always build battle forged armies. Furthermore, each faction has a number of custom detachments which give further specialized bonuses. Understanding the detachments and the rules for battle forged armies is vital for our system, and maintaining these detachments as well as rewarding armies that have detachments will be the top priority of the retrieval, reuse and revise steps of our system.

2.3 The MAC/FAC Retrieval Method

The MAC/FAC retrieval method is a two-level retrieval method. When a query is presented, some nearest neighboring candidates are chosen, and then retrieval is performed on these candidates.

The first part of the retrieval functions both as a filter, and as a performance enhancer. As a filter, the MAC step is able to filter out not just cases that are not useful, but also cases that have different, unwanted merits. This is not achievable by just setting similarity to 0 on an attribute. This process excludes cases entirely, without regarding similarity (Richter and Weber, 2013). As a performance enhancer the MAC step is a wide-net simple and computationally cheap step, which leaves us with fewer cases to perform the more expensive structural similarity retrieval on.

The FAC step uses more complex matching to determine similarity values, and then produces the best case. In this way, the computationally expensive retrieval is only performed on a relatively few cases. An illustration of the process can be seen below, in Figure 2.1.

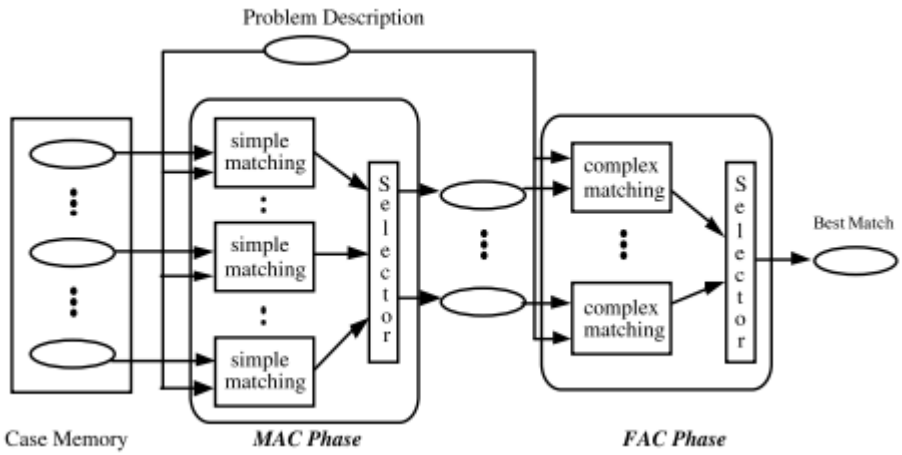

Figure 2.1: The MAC/FAC Retrieval (Adapted from Richter and Weber, 2013)

Chapter 3

Design and Implementation

In this chapter we will discuss the design and implementation of the system in detail. This chapter represents a reflection and an improvement of the design presented in the Specialization Project (Zikic, 2015). Each part of the system will be presented in detail, and the reason behind the design and implementation choices will be discussed. The design and implementation will be further discussed and evaluated in Chapter 5.

The chapter is divided into four sections. The Section 3.1 will provide a brief overview of the system as a whole. The remaining sections follow the goals presented in Section 1.1 and describe the system in more detail. Section 3.2 will discuss the design and implementation of the case based reasoning part of the system. The implementation of maintenance policies and the evolution of the system will be discussed in Section 3.3. Finally, explanation design and implementation will be discussed in Section 3.4.

3.1 System Overview

The system consists of three major parts. The first part is the CBR part, which consists of the four steps of CBR (referred to as the CBR Method): Retrieve, Reuse, Revise and Retain. The CBR part also includes the General Knowledge, including the General Knowledge and Rules for Army Building, as well as the Case Base. The explanation part consists of both the explanation logic and the context of the explanations. The maintenance part includes maintenance policies for utility and metagame. The maintenance policy for updates has not been included, due to the nature of the system. This is described more in detail in Section 3.4 The parts of the system can be seen in the System Architecture diagram presented in Figure 3.1.

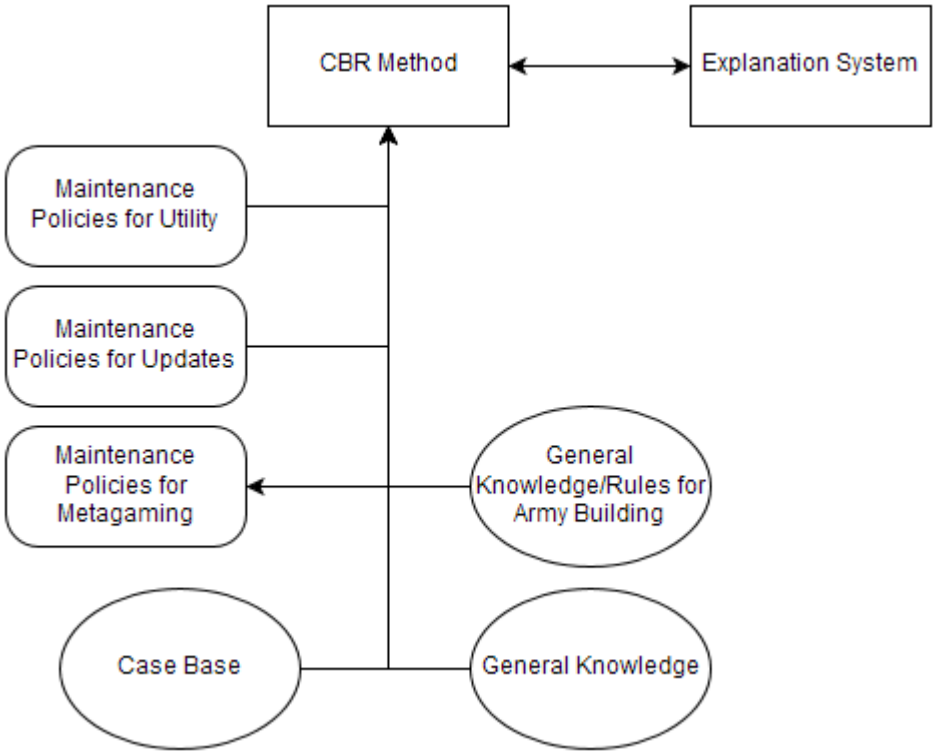

Figure 3.1: System Architecture and Overview, (Adapted from specialization project (Zikic, 2015)

3.2 Case Base Reasoning

This section will present the case base reasoning part of the system. This part of the system will be designed and implemented with three things in mind. First, we are creating an army building system, not a system that will be able to play the game. Therefore, we assume that the user of the system, that is to say the player, will be able to play to the best of their ability. Secondly, we will consider only the statistical average for dice rolls. In discussions with the experts it has been determined that more often than not luck in Warhammer 40k is indeed a statistical average, and therefore this should not impose a limitation to the system. While Warhammer 40k is technically dependant on luck and skill, a skillful player will be able to create an army that minimizes the luck factor.

Lastly, we will not discuss the topic of balance within Warhammer 40k. In discussion with the experts it has been determined that Warhammer 40k suffers from balancing problems, or in other words, there are clear advantages for playing with certain factions, as they get more advantageous rules or cheaper armies that do as well as the costlier armies from other factions. This system is created with the mindset that it is balanced and discussing balance further would be outside of the scope of the thesis. Should balance become an issue on the system as a whole, it will be discussed and evaluated separately in Chapter 5.

3.2.1 Case Representation and Case Base

In the specialization project, the intention was to have three objects, the equipment, unit, and squad objects, and the army class. This implementation, however, has evolved towards two objects, the squad and the equipment, and the army class. In other words, the unit and the squad objects have been merged together.

The cases are represented in the JSON[1] object format. JSON stands for JavaScript Object Notation and it is used as a data-interchange format. JSON is constructed completely in strings and has a simple syntax, thus it is easy to read and understand for humans. It is also easy for computers to read and parse, and parsers for almost all programming languages exist.

[1] `http://www.json.org/`

There are several reasons to represent the cases as JSON objects. The first reason is the already mentioned ease of parsing for computers, which in general means easier reading from and writing to cases. Secondly, a JSON object is capable of holding other objects within itself, and even arrays of objects. This is necessary in order to present the more complex variables of a unit, such as special rules or squad options. While we could potentially store these in a more traditional SQL, CSV or XML case base, we would need to make a separate case base for every single array object, which would be very difficult and time consuming, and not to mention that it would be difficult to load and parse.

Furthermore, as JSON objects are consistent of strings they are is easy to write by hand, so to speak, and users can generate their own custom made case bases. Many automatic, user guided interfaces for JSON also exist, making the process even simpler. If used responsibly, that is to say if the system is not flooded by bad cases or exceptions that take advantage of the system in some way, this can attribute to the teaching and learning aspects of the system. It can also aid in adapting the system to other domains. Finally, the author has a good deal of experience with using JSON as a database and far less experience with other databases.

The format of the squad and equipment objects, alongside the army class, are presented in Figure 3.2. The JSON representation of these objects can be found in Appendix D.

The Equipment Object has not changed substantially from the Specialization Project. The cost has been removed as an attribute, since different squads can have different costs for equipment, and this can vary both within the same faction and in between different factions. The name component has been added, so that we may be able to identify the equipment in question.

The Army Class representation has not changed at all, except for the addition of the name and ID component. The ID component helps the system understand which army to update after performing the CBR cycle. The Army Class has an array of squads, which have a different representation than the Squad Object, to make it easier to create armies. All that is needed for the squad object in the Army Class is the squad name and any options and parameters applied to the squad. If a squad is not present in the squad case base, the squad needs to be entered once. After that it can be used in any army composition. This eliminates the need to enter a lot of squad statistics each time we enter an army.

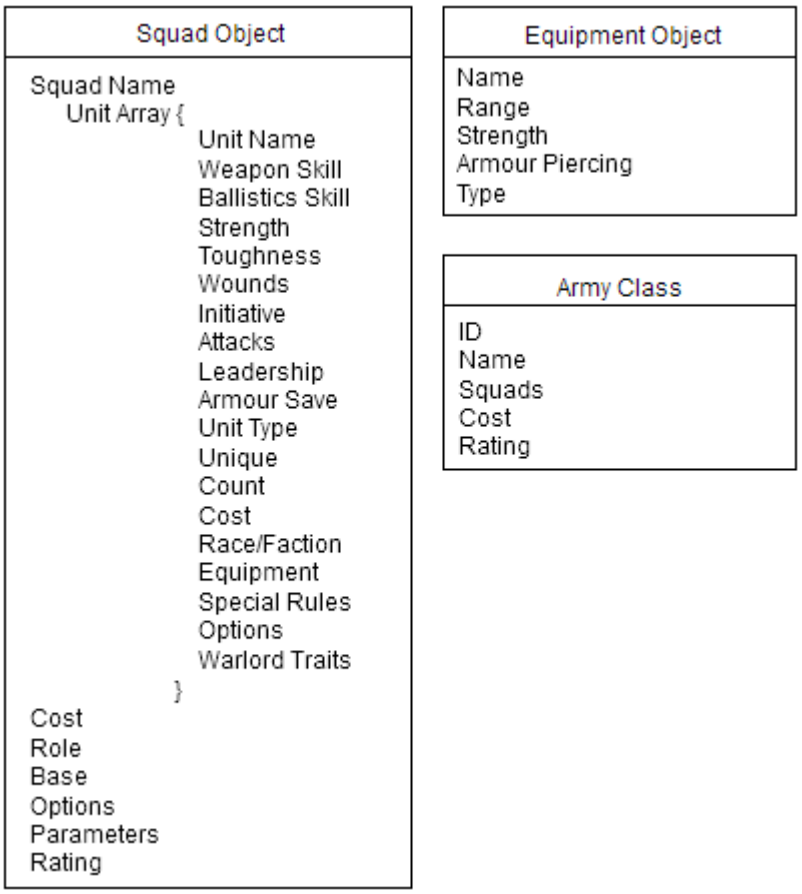

Figure 3.2: Squad and Equipment objects, and Army Class

The Squad object has been merged with the unit object. This was done because multiple units can have the same name, but not necessarily the same statistics. Furthermore, a unit from one squad can have completely different options, and even rules than a unit from another squad. In order to keep track of all the units, it was easier to add the unit object to the squad object. The Unit Array holds the Unit, which itself has changed. It now contains a Unit Name variable, which is a string value. It also contains a Boolean value called Unique, which denotes if the unit is unique or not. There can be only one unique unit of one type in the army at all times. The Count variable denotes how many units of this particular unit exist within this squad. The squad object itself contains a Role variable, which denotes the particular role this squad fulfills, when creating a battle-forged army.

The Base variable is a Boolean variable that denotes that this squad is a base squad, taken directly from the codex. Base squads are pivotal squad cases, and they can never be deleted from the case base. This is done to prevent the system from forgetting the original squad. The original squad is necessary, as without it, we could not create any squads based on that particular squad, unless we directly entered them into the system. This would nullify any advantage that we have made for creating the army class, and would make entering armies into the system a much longer process.

Finally, the Options and Parameters values are the same values as they are in the Squad component of the army. This is done so that the system understands which options and parameters have been applied to an already saved squad. It also enables easier saving of armies, without having to reverse-engineer squads made by the reuse and revise steps.

There are two more representations of the Squad Object, or rather, the Unit part of the Squad Object. Vehicles have different attributes, such as frontal, side and rear armor, that normal infantry which is presented in Figure 3.2 does not have. There are two distinctions of vehicles, walkers, which behave closer to infantry and have more similar attributes, and other vehicles, which share only one attribute with infantry, the ballistics skill. Both of these representations can also be seen in Appendix D.

From the equipment and squad objects, and the army class, the case bases are created. There are two case bases, one for the Armies and one for the Squads. The army case base is predominantly used in retrieval step, whereas the squad case base is predominantly used in the reuse step. The equipment object is stored in the same fashion, but it is a database not a case base, as we have a complete listing of all of the equipment and no new equipment can be created by the system.

Both the equipment database and the two case bases are loaded when the implementation starts, and from then on out they are not used anymore. If the case base is changed during system runtime, it will not affect the system. Once the system is shut down, the cases that are determined to be useful by the retention step are written into the case bases. It is important to note that all of the Armies will impart their squads to the case base, should their squads not be already located in the case base.

3.2.2 General Knowledge Representation and Implementation

Table 3.1 adapted from the Specialization Project (Zikic, 2015) depicts the General Domain Knowledge in the Warhammer 40k domain and is presented here for quick reference. In this subsection we will discuss how this knowledge is implemented into the system, or the reason for its absence.

Knowledge of	Description
Rules	The rules of Warhammer 40k, including the rules for constructing armies
Special Abilities	Special abilities and rules of all units and equipment
Options	Extra options provided to squads for changing them
Missions	Mission objectives and victory conditions for missions
Terrain	Terrain uses, types and heuristics for terrain advantage
Strategy	Heuristics gathered from experts which advise on general strategy for Warhammer 40k army creation

Table 3.1: General Domain Knowledge (Adapted from specialization project (Zikic, 2015)

The rules of the Warhammer 40k game are usually not implemented in the system as a table or a base of knowledge, but rather they are implemented where we need to use them. Some of the rules are not used in the game, and these will be discussed in the limitations that follow within this chapter. Other rules are used directly, while some rules are stored in the domain knowledge as methods. This extends to the special abilities of equipment as well.

The majority of the options for the Space Marine faction are fully represented in the domain knowledge. The options that are not presented are the result of system limitations, as they are very complex. If the system suffers due to this representation, a discussion will follow in Chapter 5. Other factions are not presented in the domain knowledge. The reason for this is two-fold. First, the Space Marines faction is a very popular faction, and is considered the baseline faction for Warhammer 40k. Secondly, there are around 250

options for the Space Marine faction alone. Many of the options are unique to a specific squad, and have special prerequisites for that squad. Some options are similar, like upgrading a unit or adding additional units to a squad. However, a large amount of options are quite dissimilar, and require a large amount of time-consuming work to implement into the system.

Missions are not fixed in the game and can be anything the players decide they want them to be, from the missions from the rulebooks to player generated missions. As the missions can change the rules of the game as well as being very arbitrary, they are not implemented in the system. Instead, it is assumed that the mission played gives no one army a clear advantage and that the army is the determining factor for victory. Due to this, it was decided to focus on the general aspects that can help secure and perform these missions indirectly of what the mission is. These aspects include the maneuverability of the army, the effectiveness of the army in shooting and assaulting, as well as the presence of detachments, which play a large part in securing objectives.

Terrain is fully implemented in the system, as a ratio of buildings, difficult terrain, dangerous terrain and impassable terrain. The percentages of the terrain can be changed in the system by the user, as they are also completely different from game to game, and are agreed upon by the players. Games Workshop, the creators of Warhammer 40k, recommend using as much terrain as possible, while the majority of the players seem to prefer having roughly 20-30% of the battlefield covered by terrain. Therefore, the general influence of terrain is captured in the system, but the ratio of terrain has to be input by the user. Some terrain can have additional features or impart additional rules and that terrain is not represented in the system, as it is the players choice to use this special terrain and its use is not mandatory.

The strategy of the game is implemented in the system as weights and in the reuse as the rules for adapting an army. The strategy is a combination of Internet sources, such as guides, videos and battles, and the opinion of the experts at the gaming club. Furthermore, as the system evolves and performs proactive maintenance, these weights will slowly shift to accommodate better accuracy of the system. The weights are implemented through a *.txt* file called *weights.txt*, to allow for more global manipulation of weights that is not particular to any instance of the system. Like the case bases, if the file is manipulated while the system is running it will have no effect on that instance of the system, and unless the instance of the system is terminated, it will re-write the values for the weights that it acquires through its runtime.

3.2.3 Retrieval

As discussed in Section 2.3 the retrieval step is a two-level process. While the early conceptualization of the implementation involved the structure mapping engine (SME), it became apparent that an implementation involving a SME would be very time-consuming and require a good deal of creativity to implement. Furthermore, it was not guaranteed that a SME would aid the retrieval of the system. This was reflected upon and it was decided that the MAC/FAC retrieval would be a simpler algorithm overall, but still complex enough to capture the environment of Warhammer 40k.

Many Are Called - MAC Step

When the retrieval step is initiated, the MAC step analyzes the entire army case base and retrieves a set k amount of nearest neighbours. The nearest neighbours are determined by the total army point cost and the army rating attributes. The step retrieves armies with the highest rating that are less than the point cost. However, the cost is used as a filtration mechanic and only armies within 100 points or less than the agreed point total of the game are retrieved.

Adding units to fill up the point cost of the army is very difficult for humans, and even more so for the AI. As mentioned in Subsection 2.2.1, an army of 2000 points is not made up of 1000 points of units stacked on top of a 1000 point army. Rather, the entire structure of the army changes. This means that an entire army has to be built from the ground up, which requires our reuse step to be very complicated. Instead, we only retrieve armies that are close to the cost, so that we may use substitution with squad transformation to both speed up and simplify the process for the Reuse stage. Finally, the points usually represent army strengths and retrieving an army with a high rating at 1500 points and comparing it to an army of 2000 points will in almost all the cases give it a lower similarity result, and thus filter out armies that we may have wanted in the system. This would lower the accuracy and the performance of the entire system.

As was found in the discussion with the experts, unbound armies are never used. The MAC step performs a filtration step where armies that are not battle forged are eliminated from the k nearest neighbours. This further assists in picking good armies in our filtration step and eliminating those armies that would lower the accuracy of the system.

Finally, as discussed in the Specialization Project (Zikic, 2015), the MAC step is also used to filter factions[2] or races. If a user would prefer playing with one faction over another, they can simply specify the faction in the MAC step of retrieval. Should such a faction be unavailable, the system defaults to a non specific faction retrieval. However, if a a faction is available in the system, then the system retrieves armies from only that faction. With this, the user can compromise with the system, and the system can explain its choices in the MAC step more clearly, should the the faction the user wants be unavailable in the system.

Few Are Chosen - FAC step

After the MAC step is performed, the system is left with up to a k amount of armies for the FAC step. The FAC step analyzes the structures of both armies using seven different methods/algorithms. These are then weighed and added together to provide a similarity rating. Army Ratio (AR), Squad Ratio (SqR), Strength Ratio (SR) and Favour Ratio (FR) are used in the similarity calculation, and each will be briefly discussed.

The final similarity is calculated using the formula:

$$\frac{(AR + SqR + SR * 6 + FR * 3)}{11} = Similarity$$

Army Ratio - AR and Squad Ratio - SqR

The Army Ratio is calculated in the same way as it is for chess rankings. As can be seen from the Similarity formula, the Army Ratio weight is $\frac{1}{11}$ of the Similarity factor. While the Army Ratings are important in the MAC step, the importance of army ratings is overshadowed by army composition in the FAC step. Due to this, the Army Ratings are weighted far less, so that they will not interfere as much when calculating the similarity between

[2] *Faction* and *race* in the text are interchangeable and refer to the same concept.

new armies and very highly or poorly rated armies already in the system. A highly or poorly rated army in a system should still carry some weight as it has managed to attain that rating. The exact formula for the Army Ratio is shown below:

$$\frac{10^{SolutionArmyRating/400}}{10^{SolutionArmyRating/400} + 10^{ProblemArmyRating/400}} = ArmyRatio$$

The Army Ratio will be closer to 1 when the solution army is rated much more highly than the problem army, while it will be closer to 0 when the opposite is true. At 400 or higher rating differences, the ratio is within 0.1 of either 1 or 0.

The Squad Ratio carries the exact same weight as the Army Ratio does, and is calculated using the exact same formula. The ratings that are compared are the average ratings of all of the squads combined on the solution side and the problem side. Squad Ratings are important to consider, as newer armies built up from very good squads should be higher rated, as we know those squads perform well. Furthermore, the metagame maintenance policies will adjust squad ratings based on the systems own observations, and this should be reflected in both retrieval and later reuse.

Strength Ratio - SR

The Strength Ratio itself is made up of four different ratios, each representing the four main phases of a player turn in Warhammer 40k. There is the start of turn and the end of turn phases, but as they are more of an indication when certain actions, such as scoring objectives, should be performed, they are not useful to us. The four main phases are: Movement Phase (MP), Psychic Phase (PP), Shooting Phase (SP) and Assault Phase (AsP). The formula to calculate the Strength Ratio is:

$$\frac{(MP * 1.1 + PP * 0.9 + SP * 1.1 + AsP * 0.9)}{4} = StrengthRatio$$

As the strength ratio of the armies is the main determining factor in retrieving a good army, it weighs $\frac{6}{11}$ of the Similarity factor. The weights of each individual phase was determined in discussions with experts. These are

subject to change, whether by the user or by the metagame maintenance policy.

An important note to make for the calculations here is the player actions. It is assumed that the players can take a full advantage of the active phases. A simple example involves units with heavy weapons and movement. A unit with a heavy weapon that moves can only fire what is known as a Snap Shot, a shot that only hits on a roll of 6, regardless of the Ballistics Skill of the unit. In this example, we assume that the player will not move the unit unless the advantages of moving outweigh the disadvantages. The system calculates the average for each phase, keeping in mind that the players take the full advantage of each specific phase.

In the MP the two armies are compared by their maneuverability. An army that can move faster is an army that can maneuver better on the battlefield, get to objectives, cover, and important choke points faster, or move quickly from one objective to another, or even from one enemy squad to another. The movement of each unit type is stored in the general domain knowledge.

As expected, vehicles and transports greatly increase the maneuverability of units, but so do quick units, and units that ignore certain terrain features. Terrain features are another part of general domain knowledge that we use here, and it is directly integrated in the movement calculations. Units can also move through the shooting phase, which is often referred to as the *run* action, and dice is usually rolled to determine their movement. This type of movement is also included in the movement calculations, as some units have an extra advantage while moving quickly around the battlefield, or can move a consistent amount or gain more dice for their movement. The opposite is true as well, and some units can not perform the run move action at all.

The two movement speeds, the normal and the run movement, are calculated for the problem and solution armies and then compared. In discussions with the experts, it is expected that units will use normal movement 90% of the time, and thus it weighs appropriately when calculating the ratio of movement. The remainder 10% is the Run movement.

The formulas for the movement phase are shown below:

Normal Movement:

$$0.9 * (0.5 + 0.166 * (NormalMovementDifference))$$

Run Movement:

$$0.1 * (0.5 + 0.166 * (RunMovementDifference))$$

The Normal and Run movements are added to form the similarity for the movement phase. If the problem army is slower by 3 inches, the similarity is 1, and if it is faster on average by 3 inches the similarity is 0. The speed of 3 inches is chosen as it is half of the movement of a normal infantry unit, which is the most common type of unit and therefore a good determining factor.

In the PP the two armies are compared by their psychic strengths. Every unit with psychic abilities generates their abilities before the game start, using a table and random dice rolls. Some units also start with abilities before hand. Furthermore, psychic abilities come from a few different tables, depending usually on the faction of the unit. With all this in mind, we simply can not predict what the psychic phase is going to look like, and some psychic abilities may not be used at all during the game. Nonetheless, the psychic phase is an important part of the game.

A unit that is a psyker has either the *Psyker*, *Psychic Pilot* or *Brotherhood of Psykers* ability. The psyker abilities have different levels, from 1, which is the most common, to 3 and even 4, which are extremely rare. In the game, all of the Psyker levels in an army are added, and then a six-sided die is rolled to determine the extra dice acquired on top of the combination of Psyker levels. As the six-sided die result is the same for the defender and the attacker, we can just simply compare the levels of the Psykers and draw a conclusion as to who has the advantage in the psychic phase from there.

The formula becomes the difference of the Psyker levels:

$$0.5 + 0.166 * (SPP - PrPP) = PsychicRatio$$

Where SPP stands for Solution Psychic Power and PrPP stands for Problem Psychic Power. Similarly to movement, when the difference is three or more,

the ratio will be either 1 or 0, and if the levels are the same the ratio will be 0.5.

The reason to choose the same formula in this case is due to a rule called the Perils of the Warp. Each time a Psychic ability is invoked, a unit must meet a set amount of psychic points by rolling dice and getting results of four or more. Two or more sixes rolled on this roll, however, triggers the Perils of the Warp, making something unpredictable (and usually quite bad) happen. Furthermore, many armies simply have no Psykers, which means they skip this phase entirely. Therefore, it is our belief that a fine-grained linear or polynomial scale will not provide much of a difference when it comes to the power in this phase.

In the SP we compare the effectiveness of each army when it comes to shooting. The effectiveness is measured in the amount of wounds they inflict on the enemy army. To measure this, the Ballistics Skill of the unit is used to calculate the hit chance, and each of the units ranged weapons is checked to pick the most effective weapon to fight with. This is compared to the average toughness of the enemy army, and in the case of vehicles the average between the front and side armor, as this is the expected angle that a vehicle will receive fire from. General domain knowledge is used to determine how many shots a specific weapon gets, which will then add to its effectiveness. The effectiveness is further augmented by range, and longer ranged equipment gets an incremental bonus, up to 36 inches, which is the typical width from the table edge to the deployment line of an army. Terrain is also taken into account, and provides either cover saves or complete obstructions on the battlefield. Finally, a ratio of both armies is produced, and the more effective the solution army is against the problem army, the better its ratio, up to 1 for double the effectiveness.

The final phase, the ArP is very similar to the SP. We compare the efficiency of the two armies in melee combat, using the characteristics of the units, their weapons, as well as the environment. This is compared to the problem army, and vice versa, to obtain the effectiveness ratio for the ArP. Again, terrain can halt or hinder an army, which is represented in the calculations as well. Both the SP and the ArP calculations are more fine grained, and the formula to calculate them is:

$$0.5 * \frac{SolutionEffectiveness}{ProblemEffectiveness}$$

Favour Ratio - FR

The Favour Ratio uses domain knowledge to check if the armies are using detachments[3] or not, or in other words, if they are battle-forged or unbound. Battle-forged armies gain many advantages over unbound armies and are always a better choice, even if they are stricter on the composition of the armies. Furthermore, a presence or an absence of a warlord character is checked, as the character and its warlord ability could be influential in the battle, although not as much as the battle-forged and unbound difference is.

The ratio first checks the formations for each specific faction. Then it checks for the core formations that are present in the main rulebook. This assures that all the formations and detachments are correctly analyzed by the system. The system uses the names of the squads, their individual characteristics if necessary, and the roles of the squads to determine how much of an army is battle-forged or in formation. This is then compared to the problem army, and a ratio is produced.

The favour ratio is the final calculation of the similarity, and weighs $\frac{3}{11}$ of the similarity calculation. It is an important ratio to have, and while not as impactful as the SR, it does separate the bound from unbound armies in the similarity factor. The simplest explanation for the favour ratio is that it punishes unbound problem armies.

3.2.4 Retrieval Limitations

The retrieval step is not without its limitations. As it was mentioned several times, Warhammer 40k is a very complex game and capturing every element is very difficult. In order to create a system that could represent the domain and still be functional some limitations had to be made.

The first major limitation of the retrieval step is the average values. Many of the steps use average values to determine some kind of ratio, which then calculates the Similarity index. Using average values is not perfect by any means, but it is a necessary limitation. We can not predict how a game will develop, and therefore we can not afford ourselves to measure each unit individually to each other. We must use an average value to represent the

[3]Detachments and formations represent the same concept, a formation is faction specific whereas a detachment may or may not be faction specific.

statistical average of each unit, as otherwise we are simply unable to come close to any kind of valuable result. Furthermore, even if performance is not a goal of the system, comparing each unit to each other unit for each army we retrieve will slow down the performance considerably.

The second limitation is the quantification of the complexity of the game. To produce any kind of calculable value, we need to quantify elements of the game. This means quantifying the squads, equipment and their properties to ultimately produce a similarity index for the armies. Quantifying rules is difficult and often quite precarious. There are many rules that have too many variables, many of those based on dice rolls and a certain board position, that we simply can not quantify them to a number that will be statistically sound, at least not with the time and resources we have present. Over a long period of time and after hundreds or thousands of games played and recorded, we can begin quantifying these more special rules. Quantifying rules wrongly would almost certainly lead to a drop in accuracy of the system. Therefore, only the main rules, as well as any quantifiable rules are present in the system. An example of a quantifiable rule is the twin-linked property on a weapon. If a unit with a twin-linked property misses, it can re-roll the miss to see if it could hit again. As we assume the dice are fair, this property is easily quantified using statistics.

Even with quantifying the main rules only, we are still prone to certain assumptions, which is the last limitation of the retrieval step. Some rules are such that we can not ignore them, as doing so would not present the domain fully. Therefore, we need to make assumptions for some rules. An example of the assumptions in the system is the blast marker. The blast marker is a two inch marker that is placed with its centre on top of a model. The weapon will then hit all of the models underneath the blast marker. Knowing that your opponent has blast weapons one may use maximum unit coherency, which is also two inches, to spread out the models thus allowing only one hit on a unit at a given time, even if the weapon is a blast type weapon. While this can happen, the opposite is also true, and blast type weapons could hit up to six models. By discussing with experts and looking at various Internet sources as well as battle reports involving these weapons, it had been found out that the average hit count for a blast weapon is 2. Other similar examples include average strengths on Sniper Rifles and Graviton Weapons, which have specific rules based on the situation in the game and the opponent. These assumptions are the result of necessity and over time, much like with quantification, these will reach a more stable average value that will increase the accuracy of the system.

3.2.5 Reuse

The reuse step starts immediately after retrieving the best army. As the majority of the important parameters are specified before retrieval starts, such as point total and desired faction, there is no need to implement any kind of user interaction between these two steps.

The reuse step is computationally more expensive than the retrieval step. Therefore, the step is made out of two separate stages. The first stage serve a similar purpose to the MAC step, in that it prevents the main reuse algorithm from running if the winning percentage, or similarity, is above a threshold. The second stage runs the costlier reuse algorithm.

The threshold for the first step in Reuse is 67.3% win rate, or 0.673 similarity. This is based on the newest tournament results held at the NOVA[4] tournament in the US in late summer in 2015. The results can be seen in Figure 3.3.

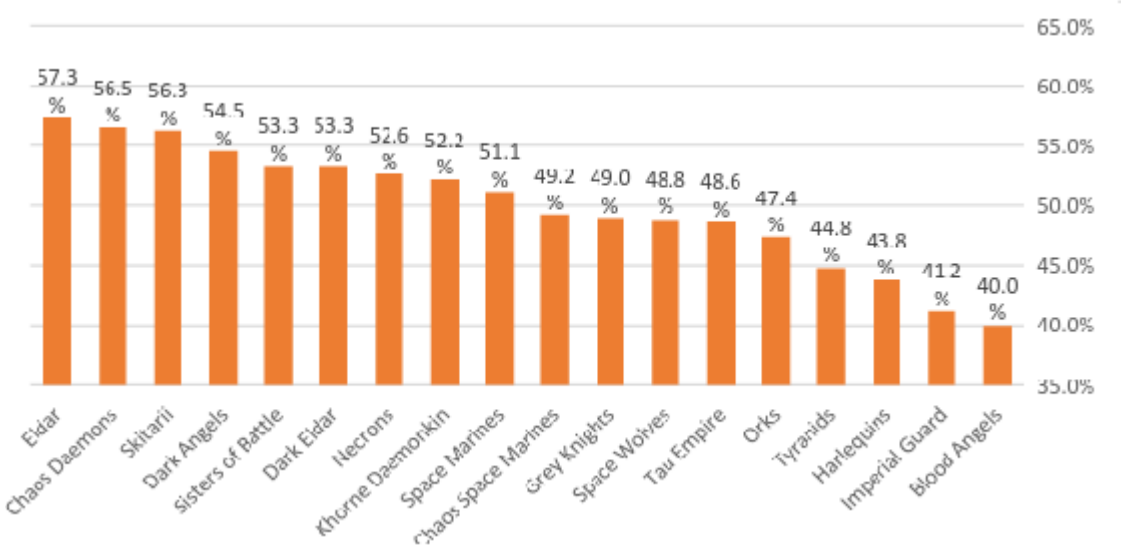

Figure 3.3: NOVA tournament results (Adapted from http://www.torrentoffire.com, 2015)

The percentage is acquired from the retrieval step, and it can easily be changed to accommodate the metagame. If the similarity index is over 0.673, 0.1 or 10% higher than the best performing army in the tournament, the reuse step is skipped, and the system moves on to the revise step.

[4] `http://www.torrentoffire.com/7287/nova-2015-recap`

However, if the reuse step is not skipped, the reuse algorithm is started. The algorithm consists of three parts: substitution, transformation and deletion. Each of the steps takes precedence over the others as the algorithm runs. Thus, substitution has a priority over transformation, and transformation over deletion. At each step it is ensured that an army is legal, by making sure that the army stays battle forged, or in detachments, and that there are no illegal combinations of units in the army, such as two of the same unique units. However, there are some very special rules that we will discuss in Subsection 3.2.6, when we discuss reuse limitations.

In the first step of reuse, we run a similarity test of each of the squads in the retrieved army individually, and compare this to the problem army, in a similar manner that retrieval does. The efficiency is calculated for the movement, psychic, shooting and assault of a specific squad, and the squad rating calculation is incorporated as well, to provide a total effectiveness table of each squad in the retrieved army. The formula for calculating the total is:

$$Total = Rank + Movement * 1.1 + Psychic * 0.9 + Shooting * 1.1 + Assault * 0.9$$

The lowest squad from the total table is then checked by squad role to see if any substitution can be made. This is done so that any detachments may be preserved, and as well so that the squad fulfills the same role. A squad of troops does not fulfill the same role as a squad of heavy support, and therefore the two should not be substituted for one another. The total efficiency of all the squads in the squad case base that meets the point value for substitution, which is the value of the squad being substituted with the addition of any leftover points, is then calculated. If a squad with a higher efficiency is found it is substituted and the algorithm starts another iteration.

If there is no squad with higher efficiency within the point cost of the army that can be legally substituted, the algorithm will attempt to use general knowledge and squad options to change the lowest efficiency squad. This could be anything from replacing equipment to replacing or adding squad members. This usually costs less than the point cost for an entire squad, and uses any leftover points, if any exist. For each such option, a new total is calculated and compared, and the best option will be applied to the squad. Again, we check for the legality of the army before we perform the transformation and after it is performed to make sure that we do not have an illegal army.

If the substitution and transformation fail to produce any results, we can assume we are at a maximum value for the army. This could either be a local maximum or a global maximum for the army. To make sure we explore the neighbouring maximums, we attempt to delete the squad from the army. This will free up points that can be used in substitution and transformation. With this we can explore the nearest maximums and find the best army based on the retrieved army. Every single time a deletion is performed, the army prior to deletion is stored as a new army in a temporary case base. After the reuse step is fully finished, we perform the retrieval step again on the temporary case base of all iterations of armies that we have. The best army is finally retrieved, adapted from the originally retrieved army.

The reuse step is executed until all of the squads have been exhausted. A squad is exhausted when it can not be substituted, as we can not find a better squad with the points we have available; it can not be transformed, as we can not find a transformation that will better the squad in the points we have available; it can not be deleted, because we would destroy the formations or detachments. Each squad, from the least to the most efficient squad is exhausted in this manner, and should a squad receive a substitution, transformation or be deleted, all of the squads are readied and the system starts from the least efficient squad again. In this way, we ensure that the reuse step considers every single option it has available.

3.2.6 Reuse Limitations

The reuse step, much like the retrieval step, has some limitations. In fact, as the reuse step also uses the retrieval step at the end of its cycle, all of the limitations of retrieval also apply to reuse as well. However, there are additional limitations that are specific to the reuse step.

The first limitation is the very specific special rules of some unique squads. Such an example is the Sergeant Chronus squad, consistent of the Sergeant Chronus unit. The unit in this squad must start out with a vehicle joined to this squad, paying the cost listed for that vehicle. This would mean that for this particular squad, we would have to create an entire exception in the algorithm, which would be both time consuming as well as influence the performance of the system. These types of special rules are quite rare, usually one per twenty squads, and it is not expected that they will impact the system too much.

The second limitation of Reuse is its dependency on the general domain knowledge. For the transformation step specifically, the reuse step depends entirely on the implementation of the general domain knowledge. Every squad option would need to be presented in the general domain knowledge, which over the course of all factions, totals to 2000 to 4000 squad options. As we have mentioned before, at least half of these would be fairly unique, and thus they would require an extraordinary amount of effort and time to implement and test, to make sure that each and every one of them is correct. We will limit ourselves to the Space Marines options, as described in Subsection 3.2.2 and therefore the majority of reuse for other factions of the system will be either substitution, or substitution followed by deletion. An expansion of the general domain knowledge in the future will greatly affect the accuracy of the reuse step.

3.2.7 Revise

Revision is performed after Reuse, though not immediately. It is expected that the actual game take place at this point, and the revision step will wait until the game is finished. It is also the only part of the CBR cycle that requires more input from the user than just the basic configuration changes.

The Revise step consists of two separate stages. The first stage is the input from the user. The user is first asked what the outcome of the match was: S if the solution army won, P if the problem army won or D if the match was a draw. After the user inputs the outcome, the step calculates the ratings of both armies, based on the formula for calculating the elo ratings.

The exact formula uses the Expected Rating formula. This is the formula we use to calculate the Army Ratio, which is presented in the Army and Squad Ratio paragraph in Subsection 3.2.3. The final rating is then calculated using the following formula, with the K value of 40:

$$OldRating + K * (Score - ExpectedRating) = NewRating$$

The K value determines the severity of change. The typical K values range between 20, which represents a minor change in ratings for a win, draw or loss, to around 40, which represents a substantial change in ratings. We use the K value of 40 by default, however, this value can be changed at any time

should it not be found ideal in the long run. The Score is a representation of a win, loss or draw, with the respective values of 1, 0 and 0.5. Appendix G contains more information on the calculation of ratings.

Once the user enters the winning army, he or she will be tasked to enter the performance of each squad. The original intent was to enter the total amount of damage caused by the squad, in points, to the enemy army. However, this faced two problems. The first problem was the inconsistency of battles and ratings. For example, if we had three identical squads, two in one army and one in the second, and the two squads killed the squad in the opposing squad together, they would gain points based on the damage done. Since all the squads are the same, the points could not exceed or equal the squad cost, and thus all squads would be considered as defeated, or lost. This would create inaccurate data over time that would hurt the systems accuracy. The second problem was the tediousness of keeping track of the damage each squad does. When discussing it with experts at the local gaming club, it was quickly dismissed as *tedious*, *time-consuming*, and *drawing away from the fun of playing the game*.

To preserve the system, but still solve these problems, we adapted a much more streamlined, albeit subjective, way of entering squad performance. Instead of keeping track of damage, the users would simply type in w, d, or l, indicating a win, draw or loss, respectively. This is the subjective performance of the user towards a particular squad and how they performed. This also allows us to capture squads that have held the objectives without doing damage, squads that have performed special abilities, or even squads that have sacrificed themselves for the win. If a user feels that a squad performed well despite the annihilation of said squad, they are allowed to enter a win into the system. Likewise, if the user is dissatisfied with a squads performance, he or she could enter a loss into the system for that squad.

The squad ratings are then calculated for each squad as the percieved performance is entered, using the same formula for calculating the Army Ratings. The score is 1 for a win, 0.5 for a draw, and 0 for a loss. The Expected Rating uses the average rating of the squads in the problem army, and the K value is always half of what it is for Army Ratings, or 20 by default. The K value is lower to provide a more granular approach to squads, because they are harder to quantify, and make a larger impact on the system as a whole, since Army Ratings are not used in any stage past retrieval.

The second stage of the Revision step is the revision of the solution army, after the ratings have all been entered. This is simply another attempt at adapting an army, in the same fashion as it is done in the Reuse step, with the new ratings in mind. As the ranking of a squad represents 20% of the total efficiency of a squad when performing adaption, there may be changes to the army based on the new value of the squad rating.

The limitations of the revision step are inherited from the reuse step. While it is a necessary drawback to keep track of perceived performance for each squad, it is still a better solution then to meticulously keep track of each damage point, and thus is considered a light limitation, but it is a limitation nonetheless. It would be possible to simply input the winning or losing army, but that would negate much of the accuracy that the reuse step could have, and the system may never be able to evolve past the quantification of rules. By having a varying variable, such as ranking, we are able to influence the system past the rules, and then update the rules through updates when we perform maintenance.

3.2.8 Retain

The Retention step is performed immediately after the Revision step. In the retention step, the armies are saved to the case base, or if they already existed in the case base their ratings are updated. For squads, the initial intention was to not save squads that perform poorly. However, poorly performing squads do help the system in Reuse, as the system knows which squads not to pick. Through discussion with experts, and a review of the Specialization Project, it has been decided to keep all of the squads in both of the armies, regardless of their performance. Squads that are not used at all, such as those squads that are created while performing reuse, and potentially revision, are not saved. However, squads that have been used in the armies are saved. Accuracy of the system is valued over its performance, which is why this choice was made. Any squad that had existed in the system already and has been directly reused instead has its rating updated.

The retention step saves the current work performed in the system. As such, we could perform several Retrieve-Reuse-Revise cycles before retaining the solution, and in some cases, such as a tournament, it may be more desirable to insert all the data from an entire day worth of battles and not just one particular battle. If we were to utilize this feature, we would need to perform

the retain function once for each cycle, as retain is performed on the solution and problem armies of a specific cycle. By default the system will execute the standard CBR cycle and then terminate. This also means that any changes performed to the case base after the system is in runtime, but before the system is finished, will be overwritten, unless the system is terminated before the Retention step can take place.

3.3 Maintenance Policies

We have discussed three specific maintenance policies in the Specialization Project: the utility maintenance, the update, or consistency maintenance, and the metagame maintenance. In this section we will discuss the implementation, or lack of such, for each policy.

3.3.1 Utility Maintenance

The utility maintenance is performed after the retention stage, and it is only performed if the system determines that it is necessary to perform it. The determining factor is the ratio between the time it takes to perform the filtered (MAC) retrieval step and the time it takes to perform the reuse step. If this ratio exceeds 0.9, or 90% the maintenance policy will start.

The implementation of the MAC part of retrieval is by nature very cheap. While we do have to iterate through the entire case base, iterating through ratings is a simple comparison, and thus takes very little time. Therefore, the bottleneck of MAC retrieval lies in the battle forged, or detachment, function, which means that the bottleneck can mostly be controlled by the size of K. Depending on how many army cases we have in the case base, the worst case scenario is iteration through the entire case base with every next case having a larger rating than the k cases already chosen. This is highly unlikely to happen, and it is likely that the average worst case will be half of the case base, due to no case base sorting.

When the maintenance policy triggers, it will collect and iterate through both the armies and the squads in the case bases. Any army that has low ranking is a candidate for deletion, but the system will always keep a minimum amount of armies, denoted by n in the system, of a specific faction, and

will instead pick the lowest ranking armies from a more numerous faction. This is done so that the system would not forget about a specific faction, even if it is lower ranked than others. The same approach is taken to squads. Squads with lower rankings are candidates for elimination, unless they are considered base squads. As the system is used, the lower ranked squads will be naturally eliminated from armies, through the process of reuse and revision, and therefore it is assumed most lower ranked squads will be removed from any useful armies, so their deletion should not create any problems in the system. Even if some armies are chosen that use these deleted squads, they can be recreated by loading that army in the future without any issues. Similarly to the armies, squads of a specific faction will only be deleted if there are more than a minimum amount of non basic squads, denoted by m in the system.

It is very unlikely that the utility will be of any kind of a problem in the system for the duration of the thesis. We would require an extensive amount of armies and squads stored in the database before the utility reached the critical point, as the reuse step is far more expensive. Nonetheless, we need to test the utility maintenance, and especially the k value and its role on the performance. If we are to have a larger case base in the future, it is vital that we understand the limitations of the implementation. Both accuracy, in the FAC step of the retrieval, as well as performance, both in the MAC and FAC steps, are effected by the k value. By inserting dummy cases, we will be able to test the k value limitations, as well as test the correctness and usefulness of the utility maintenance.

3.3.2 Consistency Maintenance

There is no firm reason for us to implement consistency maintenance. It is untestable, as we would need to wait for a new edition of one of the codices to be published, as well as implement that specific codex into the system. Currently, the system is focused on implementing the Space Marines codex, and as the Space Marines are updated to 7th edition, there is no need for further updates. The consistency maintenance is something that the project could focus on in the future, when there is a real need for it. Even so, the system is fully modular and additions and changes in the system can be made to parts of the system without the need to rewrite the entirety of the system. This should aid any future work concerning updates made to the system.

3.3.3 Metagame Maintenance

Unlike the other two maintenance policies, which are reactive, the metagame maintenance is a proactive maintenance policy. The metagame maintenance is implemented as a separate method, and can be run at any time after loading the case bases and before terminating the system. The intention is to run it whenever the system is not in use, as indicated by the user. The metagame maintenance flow is summarized in Figure 3.4. When a set amount of completions is performed, the metagame maintenance will update the weights. It can then start again, or it can be terminated.

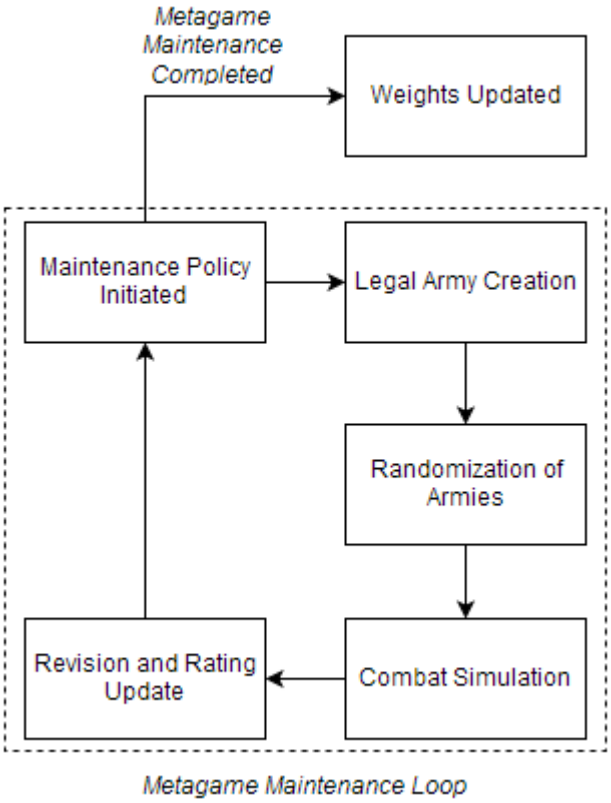

Figure 3.4: Metagame Maintenance Flow (Revised from specialization project (Zikic, 2015)

Initially, the metagame maintenance will pick a random set of squads totalling to 200 points. As previously mentioned in Table 2.1, a 200 point game is called Kill Teams, and focuses on a few squads, usually one to four, depending on their point cost. The random selection is done twice, once for the *problem army* and once for the *solution army*. The problem and solution

armies here are just a representation of the two collections of squads, and are intended to be randomized set pieces from a greater army. As such, they ignore rules for battle forged armies.

After the squads are created, but before they are collected into the army class, they are further randomized by a set of options that the squads can have. The options are also selected at random. If an option does not exist in the domain knowledge, it is simply skipped. This is done through several iterations, or until the cost is equal to two hundred points, in which case the armies are finally created.

It was initially intended that the simulation be a part of the policy as well. Unfortunately, implementing such a simulation is very time consuming, and quite difficult. Rather than being a part of this system, it would be an entirely different application, that would simulate a small time battle in different phases, with randomly generated environments. Games Workshop, creators of Warhammer 40k, currently have no such implementation[5], and to create one would also require their permission as well. Creating the simulation is therefore considered to be outside of the scope of this thesis.

This does not mean that we can not implement and test the system. It means that the system can not be fully automatized like it was conceptualized. We are still able to test the system manually by playing the game and then inputting the results in the system. Since we use kill teams, the games should be finished much faster than regular battles, allowing us to test several of them in a day.

The system will use a much lower value of K for the metagame maintenance than it is usually used for the regular battles. It is intended for the metagame maintenance to run multiple times and using a large K value would change the ratings very quickly, which is undesirable. We want to slowly reach an improvement by running the system over and over again and simulating the results.

Given time, the maintenance will not only change the ratings of the squads, but also the weights for the different similarity calculations. To do this, the maintenance policy keeps a track of advantages and disadvantages in battles. If squads with a disadvantage wins, or a squad with advantage loses, the weight of that specific advantage will go down, indicating that it contributes less to the accuracy of the system. If an advantage is correct most of the

[5] `https://en.wikipedia.org/wiki/List_of_Games_Workshop_video_games`

times, the weight will be increased, as the advantage contributes more to the accuracy of the system.

The random generation is unfortunately prone to some errors. The order in which options are applied may sometimes provide incorrect or weaker squads, which humans are able to see, but the random generation is not able to correctly predict. As the change in the system is fairly slow due to the application of the metagame policies, as opposed to the manual use of the system, such errors should be considered negligible in the long term, but can still disrupt the database and force manual change of squads, which is undesirable. The random generation is also unable to generate parameters. As such it is limited to the options that do not use parameters. This does not hurt the system, and there are still a large amount of options that the policy can apply to generate a wide variety of squads. If the random generation were to include parameters, we would need to split the general methods into unique methods for many of the options.

This ties in with the previously mentioned problem with random generation and brings us to a final limitation that can not be overcome: the method size limitation of Java. The general domain method becomes so large that each option requires its own specific method, which makes the generation process much more complex. It may be possible to restructure the code in the future, but care should be taken to preserve the accuracy of the system.

3.4 Explanation

Explanations in the system are implemented as strings. The explanations consist of generated strings, written in easy to understand language and parameters derived from the particular stage that the system is explaining. The two types of explanations that the system focuses on are the why-explanation, or the justification explanation, and the how-explanation, or the transparency explanation. At each stage of the system runtime the user will be shown the justification explanation and will be presented with a prompt that asks them if they would like a more in-depth explanation, which represents the transparency explanation.

Each step of the CBR system has its own explanation table, that is written out to the console window once the step is performed. It is assumed that no users will interact with the system as a whole in the this development

stage and the developer can in this case assist the user with reading the console, thus negating the need to build a complex user interface where it is not necessary.

After the retrieval step is performed, the system will indicate the calculated percentage chance to win for the best army picked from the case base. This is the justification explanation for this step. This is further assisted by the army composition, the name of the army and its rating, so that the explanation provides context, or the *what* explanation. If the user agrees to, a more in depth explanation is shown detailing every parameter of the retrieval process, the basic formulas used, as well as the current scope of the system. The explanation also aims to be instructive and teaches the user what is important to have in an army, although the instruction is presented through an explanation and thus is not explicit. The fine details of the retrieval method are not included, as it is believed that they would only prolong the explanation and confuse the user.

After the reuse step is performed, the system will once again indicate the win percentage, and if there was no adaption, or at least none that increased the winning percentage, this will be indicated as well. If the user agrees to, an in depth explanation will be presented describing the process and changes of each step of the iteration. The revision step produces nearly the same kinds of explanations, as it uses the reuse step again to revise the army.

After the retention is performed, the system denotes that there were changes in the system based on the rankings and that the rankings have all been updated. If the user wishes, the rankings for both the armies and the squads can be shown. As the user has manually input the point totals in the revise step, as explained in Subsection 3.2.7, the user has an inherent grasp on the situation and thus no further explanation is deemed necessary.

The utility maintenance explanations are combined justification and transparency explanations and fulfill both roles at once, as there is little to explain besides why an army or squad is deleted, for which the reason will always be the rating of the said army or squad. A list of deleted armies and squads will be displayed to the user, should the user agree to it, as an extension of the explanation.

Rather than the previously conceptualized logs for the metagame maintenance, the system uses few explanations besides the mandatory text input. As the majority of maintenance will be performed by the developer, it is not

necessary for these explanations to be very verbose or instructive. Rather, they are there to give the developer a simple indication of what should be done at the simulation step. We feel that this is sufficient, as the explanations need only be satisfactory for the developer.

Explanations are limited by the creativity of the developer, as well as the scope of the thesis. The developer has to ensure that the generated strings can satisfy the user, make them feel confident about the systems decisions, as well as teach them. Explanations are also limited by the scope of the thesis and we can not provide explanations on how to play the game, or what a good move with a specific army is and it is expected that the system serve as an assisting tool to create armies. Ultimately, the player may need to make a final decision if an army suits them, even if the army is of the faction that they wanted to play. By determining the usefulness of explanations and fulfilling the second goal of the thesis, we should be able to learn how to mitigate these limitations and if it is possible to do so.

3.5 Other Technologies Used

The CBR system is designed and implemented from the ground-up in Java. It does not use or implement any existing CBR methods, algorithms or technologies, either partially or fully.

The storage of the case bases and the equipment database is performed in MongoDB, a non-sql database solution. MongoDB, along with its drivers is an open-source, free relational database that uses JSON to store documents. While MongoDB supports their own BSON format for dynamic schemas, the case bases are in written completely in JSON and can be stored in any JSON database solution. MongoDB is run through a console locally and the system connects to it when it is started. This is the default setting. It is also possible to host the database on any kind of server, but connection information will need to be entered in the system to connect to the correct server.

The use of JSON as a case base and database and the reasoning behind it has already been discussed in Subsection 3.2.1. The use of the MongoDB technology is to simplify testing for the developer, as well as provide a cleaner interface that checks the validity of the syntax in the case bases before the system is executed.

Battlescribe, a program that aids players in creating armies, is used to create and ensure that experimentation data is correct. It enables the creation of legal armies through an easy to use interface. It is also capable of exporting these armies into an easy to read html file, which enables others to see the armies clearly as well. The program is freely available (with advertisements), and has a wide variety of knowledge bases, including Warhammer 40k. The knowledge base is created by the user community and it offers no guarantee of correctness. Each and every squad and army was therefore checked with the rulebooks when using this program to ensure the correctness of the program and the subsequent experiments. While armies can be created without the aid of this program it significantly saved time and effort in experimentation.

Chapter 4

Experiments and Results

This chapter consists of two distinct sections. Section 4.1 will introduce the experiments that will be performed in order to insure that the goals of the thesis are fulfilled and to complete the scientific method. The results of those tests will be presented in Section 4.2. The results will be presented as is, so that others have an opportunity to attempt to replicate and compare the results. The discussion, evaluation and analysis of the results, as well as the thesis and the goals will be presented in Chapter 5.

4.1 Experiments

This section will introduce the three experiments that will be performed to test the three goals presented in Section 1.1. The experiments here are related directly to the goals. Additionally, tests related to functional requirements, as presented in Chapter 3 of the Specialization Project (Zikic, 2015), are considered to be performed internally and will not be documented, unless they relate to the goals directly.

4.1.1 Experiment 1 - CBR System for Army Creation

The first experiment deals with testing if the CBR system is capable of creating an army that will have a higher than 50% chance at winning. To fully perform this experiment we must have two or more players, excluding the author so that the results are completely objective, play a game of Warhammer 40k. This game is to be played with the problem army on one side and the solution army on the other side.

To get meaningful results, we should repeat this experiment multiple times. A higher than 50% chance at winning statistically does not tell us much. As the tests are statistically independent from each other an army that has, for example, a 55% chance to win could still lose three matches before winning four. However, playing each and every game is also quite time consuming, both for the players and for the author, as the games tend to take at least a couple of hours. On the other hand, we could play several different games and aggregate all of the results and their percentages together. This would allow us to test several different armies, which would be a more meaningful test as well due to the variety of Warhammer 40k.

Unfortunately, while the local gaming club does host quite a few players that have played Warhammer 40k, many of those players have decided to move onto other gaming systems. In discussion with experts it has been determined that Warhammer 40k is in a state of unbalance at the moment, which means that many players can win simply due to rules and armies working in their favour from the start. This has caused many players to migrate to other war games.

This limitation, along with the requirements to correctly perform this experiment leads us to a problem. We can not force players to play a system, and if they do not enjoy playing it they may produce skewed results. Alternatively, even if we find willing players, they may not wish to play more than a couple of games. We do not have the resources, monetary or otherwise, to recruit any players to test the game in a controlled environment.

The solution to this experiment becomes two-fold: We will attempt to find as many players that are willing to play the game. Should this fail, we will attempt to use the knowledge of the experienced Warhammer 40k players to present their opinion on the accuracy of the system. Many experienced players are able to judge whether or not one army has an advantage, and the results, while somewhat subjective, would be the best results we can acquire

at this point of time. This search will not be limited to the local gaming club, and will involve forums and other Internet related sources. Table 4.1 shows a brief summary of how the test will be executed.

Experiment Name	Testing the CBR System for Army Creation
State of the system before experiment:	The system must fully include the problem army, as well as one or more solution armies in the case base, which includes the relevant equipment and all the squads.
Expected Results:	The system will retrieve the best army, adapt it and present it for the players. The system will then await results and make adjustments based on those results. It is expected that the chance of winning for an army will be correct, and it is expected that given several solution armies it will be higher than 50%.
State of the system after experiment:	The system will now include the new ratings for the solution army and the solution army itself, if it was adapted. The system will also adjust the ratings for the problem army, and for all the squads that participated in the match.

Table 4.1: Experiment 1 - CBR System for Army Creation

4.1.2 Experiment 2 - Evaluation of the usefulness of explanations

To evaluate the usefulness of explanations, we require two things: a set of players, both novices and experts, who will evaluate the explanations, and a set of metrics that those explanations will be evaluated by.

There are three metrics that we seek to evaluate: satisfaction, which indicates how satisfied the user was with the explanation; confidence, which indicates the users level of confidence that the system has chosen a correct or meaningful answer; learning, which indicates how much the explanation has helped the user learn about the system and the game. These metrics will be evaluated on a 1 to 5 point scale, with 1 being a completely unsatisfactory answer, and 5 being a fully satisfactory answer.

Each user will be presented with a set of explanations, and their skill level in Warhammer 40k will be recorded. The set of explanations will be the same for all users, and will encompass the explanations from the retrieval and the reuse steps. These two steps are the most important steps to explain to the user and the two remaining steps use explanations that are either largely similar or very intuitive, and therefore do not need to be evaluated. A user will be asked to rate both the justification and the transparency explanations using the metrics. This process will be repeated with as many users that are willing to participate as the author can find.

4.1.3 Experiment 3 - Application of maintenance policies

The application of maintenance policies includes the tests for the utility maintenance and the metagame maintenance policy. The utility maintenance will involve a test with dummy armies[1] where we will adjust the value of k so that we can measure its influence on performance and therefore on the utility problem. As the armies are dummy armies, accuracy of the system will not rise in this test, but it is expected that the accuracy of the system will increase when using a real case base.

The metagame maintenance policy involves a simulation of the game, as explained in Subsection 3.3.3. This experiment is quite similar to the first experiment. However, it has two major benefits over the first experiment that make the execution of it much easier. First, the experiments in the metagame maintenance are a lot shorter, since the armies are a lot smaller. Secondly, they can be performed by the author alone, since the armies are only labeled as *problem* and *solution* armies, and the author can not be biased towards one or another, as the victory of one over another does not influence the system in a any negative way such that the system appears more or less accurate than it actually is.

The simulation will have to be performed manually, and it will ignore the detachment rules, as well as any deployment rules and any rules for an army as a whole. All of the other rules are utilized, even those that are not quantified in the system currently. While the main goal of the metagame maintenance

[1] Armies from the first Experiment will be duplicated by using different names and IDs. The system does not run duplication checking when loading the case bases, only when saving them.

is to see if the system can think by itself, by creating squads and updating their ratings to find the best squad, a part of the test will also reflect the accuracy of the system, since it uses the same underlying calculations as the CBR part of the system does, only on the scale of squads, rather than armies.

Experiment Name	Testing the Application of Metagame Maintenance and its usefulness
State of the system before experiment:	The system will include all of the squads from the Space Marine faction, as denoted in the Space Marines Codex.
Expected Results:	The system will generate squads that will be evaluated based on their performance in the simulation. It is expected that the squads will be saved, alongside their new rating, or that their ratings will be updated if they already exist. Furthermore, it is expected that this experiment will generate many new squads, to allow better adaption of future armies. It is also expected that the system will change weights based on predictions, in order to increase the accuracy of the system.
State of the system after experiment:	The case base will contain new ratings for squads, as well as new squads. The weights will have potentially changed as well resulting in increased accuracy of the system.

Table 4.2: Experiment 3 - Application of maintenance policies

4.2 Results and Method

This section contains the results of the experiments, and is divided into three subsections each relating to a specific experiment. Each section will also contain the method to perform these experiments, so that the experiment can be replicated as fully as possible.

4.2.1 Experiment 1 - Results and Method

The first experiment tests the underlying CBR method and the degree of correctness and success. For this experiment we decided on ten separate battles. This meant having ten different armies in the system. Ten armies were created using the simplest and most common detachment presented in the book, the Combined Arms detachment. These armies were added to the case base, and can also be seen in Appendix E. To replicate this test on a new system, these ten armies need to be added into the case base. Within the CBR main method is a line that states which of these armies is considered the problem army. The number indicates the ID in the army case base.

Army problemArmy = ArmyList.get(0); - Sets ID:0 as the problem army.

The system was executed ten times, each time increasing the ID by one to encompass all the armies and the winning percentage of the reuse stage was noted. Nothing else needs to be changed or added in the system to replicate this test. It is important to note that different weights, as well as other modifiers in the system, like rating of squads, will produce different results. The weights in this experiment were set to 1.1, 0.9, 1.1, 0.9 for movement, psychic, shooting and assault phases respectively. K was set to 10, to encompass all armies, and the terrain consisted of 15% buildings and 10% difficult terrain.

Finally, the ten battles were presented to three different players, two veteran players and one skilled player, for evaluation. This was done by a survey, which consisted of ten battles, each with the appropriate solution and problem army. For each battle the players would either agree or disagree with the system in the calculation, as well as make additional comments. Players could also be undecided.

The results of the survey are presented in Table 4.3. Solution army in the table is abbreviated as SA, and Army is abbreviated as A. The win chance is always presented from the side of the solution army. The Combined Agreement (CA) is the degree to which all the players agree with the system. Each undecided answer is worth 0.5, each agree answer is worth 1 and each disagree is worth 0.

Battle	Win Chance	Skilled Player	Veteran Player	Veteran Player	CA
SA1 vs. A1	60.45%	Agree	Undecided	Agree	2.5/3
SA2 vs. A2	51.66%	Undecided	Undecided	Undecided	1.5/3
SA3 vs. A3	55.90%	Undecided	Agree	Undecided	2.0/3
SA3 vs. A4	54.86%	Disagree	Agree	Disagree	1.0/3
SA3 vs. A5	56.23%	Agree	Undecided	Undecided	2.0/3
SA3 vs. A6	60.53%	Disagree	Agree	Undecided	1.5/3
SA3 vs. A7	53.11%	Undecided	Undecided	Undecided	1.5/3
SA3 vs. A8	62.01%	Disagree	Disagree	Undecided	0.5/3
A7 vs. A9	62.91%	Agree	Agree	Agree	3.0/3
SA4 vs. A10	60.40%	Undecided	Undecided	Disagree	1.0/3

Table 4.3: Experiment 1 - Results

Revision and Retention

We have applied revision and retention to three matches from Table 4.3. The matches are SA1 vs. A1, SA2 vs. A2 and SA3 vs. A8. The first match was counted as a win for the solution army, the second as a draw, and the third as a loss. The matches were entered in the system in that order. The squads all follow the resolution of the matches as well, so there are no draws for the squads in a win or loss situation. For this part of the experiment all maintenance functions have been turned off.

After inputting the data for the first match, no revision took place and all the squads from both of the armies were saved into the squad case base, and the SA1 was saved in the case base as the 11th army, with ID 10. The ratings were set, as expected, to 1520 for SA1 and 1480 for A1.

After inputting the data for the second match, once again no revision took place. Due to the previous match, the Kor'sarro Khan 2 Squad had its ratings changed, which increased the winning chance for SA2 by 0.03%, to 51.69%.

The ratings were adjusted accordingly for the draw, with previously unrated squads loosing or gaining small amounts of rating due to the ratings being uneven from the previous match. The army ratings did not change, as they were both even to start with.

After the third match, the army retrieved was the SA1 army, not SA3, due to the higher rating. As we wanted to experience a loss, we tried to perform a change to SA4 vs. A10, which also retrieved SA1. The last was the SA3 vs. A4, which again retrieved SA1. As the winning chance was the lowest for this instance, this instance was taken as a loss for SA1. No revision took place in the revision step, and the army ratings were adjusted again. After running the system one more time, SA3 was chosen instead of SA1 once again for the SA3 vs. A4 match, however no further experimentation was executed.

4.2.2 Experiment 2 - Results and Method

The second experiment is the evaluation of the usefulness of explanations. To perform this evaluation we picked the first army from the previous test and presented the explanations to an expert and three novice users. The explanations were written into the Java console from where they were copied onto a more visible sheet. However, the integrity of the explanation was not changed. Most of the explanations between different problems are similar, varying only in variables and the path that the system takes to achieve an answer. Therefore, any of the ten battles would suffice for our experiment.

Like in Experiment 1 a survey was created for the users to answer. The users were first presented with a simple statement from the retrieval step that states the winning chance of the solution army. They were tasked to rate this statement in three categories previously mentioned: Satisfaction, Confidence and Learning. This rating was scaled from 1 to 5, with 5 being the most positive answer and consequently 1 being the most negative answer.

Then, the more In-depth explanation was presented and again the same categories were evaluated. This was repeated for the reuse step as well. As mentioned before, we have skipped the revision and the retention steps. Another reason for skipping these two steps is the confusion that may arise if a user does not know why or how an army won. This would entail an explanation outside of the system and might not produce correct feedback. Finally, users were asked to rate the situation where no explanation was provided and only the adapted army was presented.

The results of the experiment can be seen in Table 4.4. The answers were averaged over the four users, with the expert answer presented additionally in brackets.

Explanation	Satisfaction	Confidence	Learning
Retrieval - Simple Explanation	3.5 (3)	2.5 (2)	1.5 (2)
Retrieval - In-depth Explanation	4.5 (4)	3.25 (3)	4 (4)
Reuse - Simple Explanation	3.75 (3)	2.75 (3)	2 (3)
Reuse - In-depth Explanation	3.75 (4)	3 (2)	2.75 (3)
No Explanation	3 (3)	2.5 (2)	1 (1)

Table 4.4: Experiment 2 - Results

Experiment 2 - Second Iteration

While performing the experiment a specific comment was noted from almost all the users. The in-depth explanation of the reuse step was not found very instructive, and the users felt they learned little from it. The explanation was restructured to resemble the in-depth explanation for retrieval and was put to a new test in a new survey. The results of the test are summarized in Table 4.5, alongside the previous results and the difference between the two.

Explanation	Satisfaction	Confidence	Learning
New in-depth Explanation	4.25 (4)	3.75 (3)	4 (4)
Old in-depth Explanation	3.75 (4)	3 (2)	2.75 (3)
Change	+0.5	+0.75	+1.25

Table 4.5: Experiment 2 - Second Iteration

4.2.3 Experiment 3 - Results and Method

Experiment 3 - Utility Maintenance

The first part of this experiment is the utility maintenance application. It is executed on incrementally larger case bases, to check the limitations of the k value, as well as the standard reuse time. To replicate the test, one would need to fill in the case base with any valid army, and then duplicate that army multiple times. The system should be then run with different values of k once there are a sufficient number of armies in the case base. For this experiment we have created 160 armies in the case base.

Depending on the specific machine, the time for execution will vary and exactly the same results should not be expected. However, the ratio between reuse and retrieval should remain constant, which is our main concern. The results of the test can be seen in Table 4.6. Each test has been performed three times, and an average of the performance was registered.

For the reuse step we have taken an average between two executions for each enemy army in the system, totalling 20 executions. It is expected that the reuse step will vary from army to army, which is the reason for measuring it in this fashion. The mean execution time of the reuse step was 399.05 milliseconds, with the shortest execution time being 251 milliseconds, and the longest being 480 milliseconds. The mean is very important to consider, as it is a rough estimate of when the utility maintenance will trigger, with the shortest execution time being the lowest threshold.

K-Value	Time (In milliseconds)	Mean (In milliseconds)
10	11, 9, 8	9.33
20	20, 20, 15	18.33
40	31, 42, 28	33.66
80	66, 55, 76	65.66
160	118, 157, 180	151.66

Table 4.6: Experiment 3 - Utility Maintenance

Experiment 3 - Metagame Maintenance

The second part of this experiment is the metagame maintenance application. It is executed in three sets of ten matches, after which the weights were updated if they needed to be. Due to the random generation of squads, it is impossible to replicate the exact results by running the maintenance policy by default. To replicate the results, one will have to manually enter each squad, with the options, in the same order as they are performed in the test, by following the IDs of the matches. The simulation for replication of results needs not be performed, and the outcome can be entered immediately if it is desired.

Each match of the metagame maintenance was randomly generated by the policy, and then simulated by the Author. The matches were played on a 90 by 60 centimetre table area[2]. The table had four deployment zones, one on each side, that were rolled randomly for each squad on a four sided dice. If the dice rolled the same number for the second squad, the roll was repeated until two different deployment zones were generated. The deployment was performed in the centre of the table side, in a line for miniatures larger than infantry, or a circle for infantry sized miniatures[3]. A 6 sided die was rolled for each side, and the higher dice won the first round. In the case of a tie the roll was done over. The playing field consisted of two obstacles that were 20 centimeters (8 inches) long, and were tilted 60 and 30 degrees respectively towards the long table edge. The first one contained a 4+ cover save, whereas the second one was debris and represented difficult terrain. The exact composition of armies, initiatives, placement, the new ranks acquired after the match, as well as the sketch of the playing field can be found in Appendix E.

It is important to note that during this test, any illegal squad was ignored and the policy was tasked with randomly generating a new set of squads. We were able to do this due to manual simulation. There were a total of 4 new matches generated out of 30, giving the metagame policy a 13.3% fail rate. This ties in with the limitation that was already discussed in Section 3.3.3.

[2]In the imperial system this is 36 by 24 inches, half the size of the regular 6 by 4 foot playing field.

[3]This is the smallest miniature in Warhammer 40k, measuring 25mm, or roughly 1 inch.

Each match was played to the authors best abilities to understand all the rules of each squad. The author attempted to utilize the every squad fully, maximizing the gains wherever possible, with one limitation. A squad will not attempt to *fish*[4] for a dice roll more than twice. In other words, squads that were extremely fast, like Land Speeders or Bikes, could in some cases just run away to their maximum distance and shoot, and never be caught by the slower units. They were allowed to do this for two rounds, before being forced to stand and fight. This was done to speed up the tests, as in some cases a unit could only be hurt with a very small chance, and this would make the tests last for well over half an hour to an hour of just dice rolling and probability checking. Furthermore, it is not a good way to simulate an actual combat scenario, and in the real world it would most likely be considered very unsportsmanlike.

After each set of ten matches was finished, the weights were adjusted. The weights at the start of the test were 1.1 for shooting and movement, and 0.9 for psychic and assault. For each ten accurate predictions, the weight would go up by 0.01, and for each ten inaccurate predictions down by 0.01. Each prediction is made on the level of a single squad, which means that the more squads were involved the more correct or incorrect predictions were acquired. The summation at the end of each table makes it easy to see, at a glance, if any changes occurred after the ten matches. For example, in Table 4.9 we can see that the weight after these ten matches was changed by -0.01 for the Shooting Phase and +0.01 for the Assault Phase.

All of the results can be seen in Tables 4.8-4.13. A legend of what each column means can be seen in Table 4.7.

[4]A common term used for trying to get a specific dice roll that is usually statistically difficult to get, such as getting three sixes in a row on a six sided die.

Acronym	Meaning
ID	Match ID. This matches up to the extended results in Appendix E.
SR	Squad Ratio. This is the strength of the two army squads compared. It ranges from 0-1, and it is presented from the Army 1 standpoint. The second armies squad ratio is 1 - squad ratio in the table.
MP/MP2	Movement Phase for Armies 1 and 2. This is their respective strength weighted in the movement phase.
PP/PP2	Psychic Phase for Armies 1 and 2. This is their respective strength weighted in the psychic phase.
SP/SP2	Shooting Phase for Armies 1 and 2. This is their respective strength weighted in the shooting phase.
AP/AP2	Assault Phase for Armies 1 and 2. This is their respective strength weighted in the assault phase.
pMP	The number of accurate predictions in this match that were made for the movement phase. If a unit does poorly but the movement score is higher than half of the maximum weighted predicted movement phase it will get -1 and vice versa. The other predictions also follow the same calculations
pPP	The number of accurate predictions in this match that were made for the psychic phase.
pSP	The number of accurate predictions in this match that were made for the shooting phase.
pAP	The number of accurate predictions in this match that were made for the assault phase.
pExpected	The expected win chance / strength ratios between the armies. This is modified not just by the ratios of strengths, but also by the total point strengths of the armies.
Outcome	This is the outcome of this specific match for Army 1 and 2 respectively. A w denotes a win, a d denotes a draw, and an l denotes a loss. The losing side all gets losses. Any squad on the winning side that does not survive until the end is marked as a draw.

Table 4.7: Experiment 3 - Legend

ID	SR	MP	PP	SP	AP	MP2	PP2	SP2	AP2
1	0.5	0.5	0.45	0	0.9	0.65	0.45	1.1	0
2	0.5	0.88	0.3	1.1	0.59	0.2	0.6	0	0.34
3	0.501	0.73	0.45	1.1	0.9	0.33	0.45	0	0
4	0.501	0.55	0.45	0.52	0	0.55	0.45	0.57	0.9
5	0.5	0	0.45	0	0.25	1.1	0.45	1.1	0.78
6	0.5	0.55	0.45	0.85	0.57	0.55	0.45	0.36	0.35
7	0.487	0.97	0.45	1.1	0.46	0.03	0.45	0	0.44
8	0.5	0.93	0.45	1.1	0.5	0.13	0.45	0	0.4
9	0.499	0.33	0.45	0	0.68	0.75	0.45	1.1	0.3
10	0.502	0.18	0.45	0	0.48	0.86	0.45	1.1	0.42

Table 4.8: Experiment 3 - Evaluated Ratios Test 1

ID	pMP	pPP	pSP	pAP	pExpected	Outcome
1	-4	0	-4	+4	0.494/0.504	w,w/l,l
2	+5	-5	+5	+5	0.773/0.229	w,w,w/l,l
3	+5	0	+5	+5	0.674/0.318	w,w/l,l,l
4	-3	0	-5	-5	0.405/0.594	d,d,w/l,l
5	+1	0	+1	+1	0.211/0.817	l/w, d
6	-1	0	-3	-3	0.621/0.406	l,l/w
7	-3	0	-3	-3	0.723/0.258	l,l/w
8	+3	0	+3	+3	0.694/0.298	d,w,w/l,l
9	-5	0	-5	+5	0.391/0.618	w,w/l,l,l
10	-6	0	-6	+4	0.317/0.675	w,d,w/l,l,l
Sum	-8	-5	-12	16	N/A	N/A

Table 4.9: Experiment 3 - Predicted Scores And Outcome Test 1

ID	SR	MP	PP	SP	AP	MP2	PP2	SP2	AP2
11	0.496	0.83	0.45	1.1	0	0.25	0.45	0	0.91
12	0.501	0.31	0.45	0.48	0	0.77	0.45	0.61	0.91
13	0.505	1.0	0.45	0.71	0.6	0.05	0.45	0.41	0.34
14	0.507	0	0.45	0.68	0.41	1.1	0.45	0.43	0.5
15	0.497	0.57	0.45	0.68	0.63	0.52	0.45	0.44	0.33
16	0.501	0	0.45	0.77	0	1.1	0.45	0.38	0.91
17	0.501	0.2	0.45	0	0	0.83	0.45	1.09	0.91
18	0.498	0.17	0.45	0.45	0.51	0.89	0.45	0.66	0.4
19	0.498	0.55	0.6	0.98	0	0.55	0.3	0.3	0.91
20	0.489	1.1	0.45	0.35	0.39	0	0.45	0.85	0.53

Table 4.10: Experiment 3 - Evaluated Ratios Test 2

ID	pMP	pPP	pSP	pAP	pExpected	Outcome
11	-3	0	-3	+3	0.611/0.388	l,l/w
12	+5	0	+5	+5	0.351/0.647	l,l,l/w,w
13	+2	0	+2	+2	0.651/0.354	d,w,d/l,l,l
14	-5	0	+3	-5	0.421/0.584	w,d,w/l,l
15	+4	0	+4	+4	0.578/0.436	w,d,w/l,l,l
16	-4	0	+4	-4	0.351/0.660	w,w/l,l
17	+3	0	+3	+3	0.230/0.758	l,l/d,w,w
18	-5	0	-5	+5	0.426/0.571	w,w,w/l,l
19	+1	+3	+3	-3	0.538/0.499	w,w/l
20	-5	0	+3	+3	0.564/0.461	l,l/w.d.w
Sum	-7	3	19	16	N/A	N/A

Table 4.11: Experiment 3 - Predicted Scores And Outcome Test 2

ID	SR	MP	PP	SP	AP	MP2	PP2	SP2	AP2
21	0.487	1.1	0.45	0.43	0.70	0	0.45	0.7	0.3
22	0.489	0.55	0.45	0	0.41	0.55	0.45	1.1	0.52
23	0.488	0.83	0.45	1.1	0	0.23	0.45	0	0.92
24	0.489	0.16	0.45	0	0.92	0.92	0.45	1.1	0
25	0.507	0.55	0.45	1.1	0.92	0.55	0.45	0	0
26	0.49	0.28	0.45	1.1	0	0.83	0.45	0	0.92
27	0.504	0.24	0.6	0.52	0.84	0.88	0.3	0.59	0.25
28	0.498	0.2	0.45	0.37	0.92	0.91	0.45	0.81	0
29	0.482	1.1	0.6	0.35	0	0	0.3	0.86	0.92
30	0.493	0.11	0.45	1.1	0.31	0	0.45	0	0.68

Table 4.12: Experiment 3 - Evaluated Ratios Test 3

ID	pMP	pPP	pSP	pAP	pExpected	Outcome
21	+1	0	-5	+1	0.636/0.386	d,w,d/l,l
22	+1	0	+3	+3	0.366/0.637	l/w,w
23	-5	0	-5	+3	0.542/0.453	l,l/w,d,w
24	-4	0	-4	+4	0.380/0.615	w/l,l,l
25	0	0	+2	+2	0.678/0.321	w/l
26	-4	0	+2	-4	0.450/0.552	w,d/l,l
27	+5	-5	+5	-5	0.539/0.502	l,l/w,w,w
28	-4	0	-4	+4	0.460/0.557	w/l,l,l
29	+3	+3	-3	-3	0.499/0.522	w,w/l
30	+2	0	-4	+4	0.514/0.304	l,l,l/w
Sum	-5	-2	-13	9	N/A	N/A

Table 4.13: Experiment 3 - Predicted Scores And Outcome Test 3

After the test was completed 34 new, unique squads were generated in the system. This has increased the number of squads in the system from 54 to 88, effectively increasing the case base by 63%. These squads can be seen in Appendix E, alongside the other squads in the system and are easily recognizable by the options or additional numbering applied to them.

Chapter 5

Evaluation and Discussion

This chapter consists of three sections. Section 5.1 will present the evaluation and analysis of results. Section 5.2 will discuss these results further, focusing on the limitations and the implementation and the thesis. Finally, the main contributions to the field will be presented in Section 5.3.

5.1 Evaluation

In this section we analyze and evaluate the results based on the three experiments performed. The evaluation is divided into three parts, much like the experimentation is, and is presented in the same order.

5.1.1 Experiment 1 - CBR System

From Table 4.3 we can see that the experts have overall agreed with the system in 40% of the cases, disagreed in 30% of the cases and were undecided in 30% of the cases.

If we take a look at the players on an individual level, we can see that there is, on average, no discrepancy between the Skilled Player evaluation and the Veteran Players evaluation and this can be seen in Table 5.1. The veteran players are in this case more experienced than the skilled player.

Player	Agreements	Undecidedness	Disagreements
Skilled Player	3	4	3
Veteran Player	4	5	1
Veteran Player	2	6	2

Table 5.1: Comparison of different player evaluations

From Table 5.1 we can conclude that the domain is truly difficult to evaluate and in many cases we can not make a certain evaluation, especially not without playing the matches many times. What we can state from this experiment is that in 70% of the cases the system was capable of producing an army that would not definitively lose.

As there are cases where the winning chance is quite high for both agreements (62.91%) and disagreements (62.01%), as well as quite low, 55.90% for agreements and 54.86% for disagreements, we can also state that the winning percentage seems to vary substantially and can not be considered a good indication of winning or loosing. A more thorough discussion of this problem will be presented in Section 5.2.

One result that is not clearly presented is the influence of the k number. We have set k to 10 to encompass the entire case base and in the majority of the cases in the experiment it did not make a difference. However, one case was based of off Army 7 and Army 7 in that case was also the best choice. If we had set k to 6 or lower, we would not have been able to reach Army 7, as all of the armies are equally rated, so the system will choose the first six armies in order from the case base.

When performing the experiment with k=6 instead, the system chooses Army 4 in the retrieval stage with a 61.22% chance of winning. The army is not adapted, as no better solution can be found. Comparing it to the winning chance of Army 7, which is 62.91%, we can see that there is a fairly substantial change in accuracy. While the accuracy may not be a great indicator of winning, it is clear that if we improve the accuracy we need to be aware of what the value of k in the system is. Setting it too low will impede accuracy, but setting it too high will impede performance. In this particular experiment, setting k to 10 was considered acceptable, as we wanted to test the full potential of the CBR system, and performance was not considered an issue. This may only play a part in creating a large case base, but it is something we need to be aware of.

The revision and retention parts of the experiment have for the most part performed as expected, but another issue has been raised, namely: should armies inherit their previous ratings, or should they have default ratings when being adapted. In other words, does the accuracy of the system suffer more if we inherit the rating of the base army in adaption, or if the default rating should be enforced whenever we change anything within an army. This will require an extended period of testing with a control group and a test group, and would be part of the future work done on the topic.

5.1.2 Experiment 2 - Usefulness of Explanations

Table 4.4 shows the results for the explanations experiment. An immediate result to notice is the learning aspect of an explanation. For all the explanations, the more a user feels they have learned, the better their satisfaction and confidence levels. On average, an increase in the learning score by 2 will increase satisfaction and confidence by 1. This is noticeable in the second part of the experiment as well, and in the comparison between the first in-depth reuse and the second one. The former had a learning score increase of only 0.75 from the simple explanation, and no increase in satisfaction and a minor increase in confidence. The latter had an increase of 2 points, with the increase in satisfaction by 0.5 and in confidence by 1.

This tells us that learning about the system and making an explanation instructive is very important for new users. Users that are not familiar with the domain will be more inclined to believe the domain if they know something about it, thus their confidence and satisfaction will increase. For the skilled player, the learning was almost always higher on average than for the new users. A skilled player is most likely able to recognize patterns in the context and draw knowledge past what is explicitly stated in the explanations, which is expected.

Another analysis to make is that in all the cases of in-depth explanations, the users were more satisfied and confident than in simple justification. This can be attributed to two things. First, the aforementioned learning increases the confidence and satisfaction in the system. Second, the nature of the users. All of the participants of the experiment have hobbies in playing games, be it board games, role-playing games or computer games. They are used to reading rule books in-depth to grasp game mechanics of the game they are playing.

This is not necessarily a bad thing. After all, most of the people attracted to Warhammer 40k most likely have an interest in it, and thus have more motivation to read and understand the in-depth explanation. However, it is possible that the in-depth explanation would be overwhelming for those without the same kind of motivation. As a certainty, however, we can say that those that are motivated to play the game, be it new or skilled users, will enjoy the in-depth explanations more than the simpler explanations.

Finally, in the comparison of no explanation to explanations, it is clear that most users prefer to have an explanation. However, it seems like most users would still be somewhat satisfied and confident without having an explanation. This will be further discussed in Section 5.2.

5.1.3 Experiment 3 - Application of Maintenance policies

The utility maintenance policy indicates that the mean performance is slightly below the k-value in milliseconds. This means that the performance decrease is roughly linearly proportional to the k-value. This, coupled with the reuse step needing between 251 to 480 milliseconds, with a mean of 399.05 milliseconds, indicates that the maximum k value for the system lies approximately at k = 359 (due to the 0.9 ratio trigger between retrieve and reuse), with a minimum of k = 225. This means that up to 225 armies at the minimum can be selected in the MAC step, which is most likely a sufficient amount to find a good solution army.

The metagame policy had managed to create 34 new squads after 30 iterations, increasing the squad case base by 63%, from 54 to 88 squads. This is an excellent result for the system, as it shows that the policy is able to create new cases without user interference, assuming a simulation was indeed created and not manually performed. The generation of squads was 15 through the first ten iterations, 10 through the second ten iterations, and 9 through the last ten iterations. Therefore, we can state that the system generates less squads as more iterations are performed, which is expected, as the generation is random. As the number of squads in the system increases, the chances of the system picking a squad it had made increases, and thus the chances of creating a new squad decreases. Naturally, it is possible to further adapt a new squad or choose a squad that can fit another squad combinations to force a permutation, and thus an adaption, on the squads. There are many adap-

tions and permutations we can perform, however performing them randomly with the point limit limits the selection of new permutations. Assuming this progression continues, with one less squad created per ten iterations, the system could create at least 70 new squads from the 54 base squads, before creating one new squad every once in a while. This is an increase of 129.62% on the main case base. Furthermore, if the systems are able to exchange case bases it is statistically certain that another system will follow a different random pattern and be able to create different squads. Therefore, we can state that the metagame policy is successful, and if a simulation is applied to it that can simulate the battles without manual interference, the system will be able to *think* and create cases on its own.

Another important part of the metagame policy is the strategy, and with it the accuracy of the system. Tables 4.8-4.13 show the results of the metagame policy application. From the tables we can see that the system had 12 good predictions, and 18 bad predictions. In other words, the system predicted the outcome correctly 12/30 times, or 40% of the time. However, there are two other important factors that can impact this: initiative and placement. Initiative means an army gets to play first, which can lead to a significant advantage. Placement can also hinder and help armies. There are four placement pairings (1,2),(1,4),(3,4),(2,4) that favour assault heavier armies, as the initial deployment is within 21 inches or less.

Following the rules of deployment armies could sometimes deploy as close as 12 inches apart, which is easily reachable for any unit within 1 round. The other two placements (2,4) and (1,3) start at about 22 and 34 inches apart respectively after deployment, making it more suited for shooting armies.

Table 5.2 shows the advantages in initiative and placement for correct predictions and for incorrect predictions, as well as the number of matches both advantages were present.

Prediction	Initiative	Placement	Both
Correct Prediction	6 / 50%	11 / 91.6%	6 / 50%
Incorrect Prediction	9 / 50%	11 / 61.1%	5 / 27.7%

Table 5.2: Comparison of predictions against placement and initiative advantages

From Table 5.2, we can see that in the percentage values for initiative are exactly the same, however the values for placement and both values favour the correct predictions somewhat. This can be contributed to a small sample size, but from the results at hand we can conclude that the advantages favoured the correct predictions.

Another factor we can consider is *even games*, or games that were within 0.1 similarity. In other words, how many incorrect and correct predictions would we have considered to be close matches. Out of the 30 matches, 5 matches were within 10% difference in predicted win chance. Of those, 4 were incorrect predictions and 1 was a correct prediction. In these 5 matches the advantages were split evenly, with both the correct and one of the incorrect predictions having both advantages, and the other incorrect predictions having 1 advantage. Therefore, the only determining possible factor statistically could have been luck, again due to the smaller sample size. Assuming that (even though impossible), 2.5 matches were correctly predicted and 2.5 matches incorrectly predicted, we would arrive at 13.5 correct and 16.5 incorrect predictions, which means that in the best case scenario, assuming that the small sample size is the reason for the percentage values of advantages being in favour of the correct predictions, the system was only accurate 45% of the time. It should be noted, however, that no draw conclusion was possible from this test, unlike in regular 40k games. This means that we may have to rethink our approach to assigning ratings in the metagame maintenance policy.

The final part of accuracy is the adjustment of weights and with it the strategy. The first ten iterations had 4 correct predictions. The second ten iterations, after adjusting the weights, had 5 correct predictions. After adjusting the weights once more, the number of correct predictions fell to 3. We can not ascertain the scope that the change in weights has had on the system, and many more iterations will need to be performed in the future to acquire a conclusive answer.

5.2 Discussion

This section will discuss the limitations of the system, both those that were made and how they impacted the results, as well as those that were found after performing experiments. The section will also discuss the shortcomings of the system as well.

5.2.1 The Case-Base Reasoning System

The quantification and generalization of the domain has had a clear impact on the accuracy of the system. The winning percentage seems to make little difference in our experiments, indicating that the accuracy of our system definitively needs to be increased. We should first produce a much larger set of results to see the extent of the deviation in accuracy, before attempting to integrate more rules.

A clear result that can be seen from the data is the armour values of units, the armour penetration values of equipment, and their combined influence on the match. As it is now, the system generalizes armour from infantry and armour from vehicles into a general statistic. This is done, because we can not predict whether or not an army will shoot at a vehicle or not. This does lead us into a problem where certain sets of units are practically invincible, since their armour is very tough and nothing in the enemy army can penetrate it. At the same time, the army as a whole is still weaker, leading to a flawed calculation. This is one of the limitations that we have discussed in Section 3.2.4, where we had to quantify a rule that would take away from the game if it was not quantified. Clearly, it has had a negative impact on the accuracy of the system and needs to be rethought and re-implemented, so that it has a more positive impact on the accuracy.

We had expected that the generalization and quantification of the domain would be a limitation to accuracy. However, one more limitation was found while performing the experiment, which is the matchup against other factions. Some of the armies that the system created would be considered *bad* when matched up against some other factions. The players commented that many of the factions have a possibility of shutting down multiple special abilities from another faction. This means that we must also impose penalties and bonuses for different factions as well.

This limitation, alongside the limitations previously discussed in Sections 3.2.1-3.3 lead us to a conclusion that creating a system that can accommodate all the variables of the domain is a very difficult task.

During the evaluation the users also reported that some of the armies would do much poorer in some missions in comparison to others. This was expected, as the system was designed to have a general approach to missions. However, the degree of feedback was such that this approach may have to be re-evaluated. The tactics and the army composition seem to vary far more than it was originally anticipated and in some cases the tactics are completely reversed, with what was once bad a bad strategy becoming a good strategy. Therefore, we need to be more aware of what kind of mission we are going to play and impose maluses on armies that do not meet those requirements and bonuses on those that do.

Alliances have not been implemented in the system despite the experts agreeing that they are consistently used. As explained in the specialization project, alliances have four different levels: blood brothers, allies of convenience, desperate allies and come the apocalypse. Blood brothers need not be implemented separately, as they do not impose any new rules on the game. All of the other levels impose new rules. These rules are always present in the game and alongside many of the rules can not be easily quantified. They always pose some kind of penalties, but intelligent deployment can rid the army of many of these penalties, as they are mostly based on the distance between the units. However, as we can not anticipate the mission, terrain or in general the setting for the game, we can not quantify these rules to increase the accuracy of the system and to represent alliances besides battle brothers correctly. Moreover, alliances may be seen as somewhat advanced and new players may not have the miniatures or the knowledge of two factions enough to create an alliance that would be beneficial. Finally, we would also need to represent multiple functions for an alliance to actually be active and we currently only represent one faction in the system. Therefore, we have chosen not to implement any alliance rules in the system. If we seek to expand the system to multiple factions, these rules will need to be implemented.

Instead of approaching the domain globally, it would perhaps be better to continue the focus on one faction and implement general domain knowledge tactics and strategies, with appropriate bonuses and penalties for the one faction. This would be achieved by introducing parameters that are specific for the game, such as mission types, night-fighting, special objectives and more before and doing the introduction of the enemy army. This will also

require us to understand how the units of the faction perform under these condition, as well as which tactics to employ, and thus how the system weighs the different factors. Limiting the choice to one faction will make this endeavour more probable, as capturing the data for all the factions will be a very lengthy, if not impossible, process, due to the fairly frequent domain changes. However, this process can be performed completely in parallel and as such it is possible to do should enough manpower be available.

Finally, the last issue discovered targets the entire domain and influences both the CBR part of the system as well as the explanation part of the system. The experimentation was performed on only a handful of people, but this was not due to the lack of trying. Help was requested from both the Warhammer 40k Reddit pages, which is frequented by hundreds of people per day and has over 23 thousand registered members, and the main page of the DakkaDakka forums, which is frequented by thousands of users per day and has over 100 thousand registered users. It was found that the majority of the users are satisfied with painting and collecting the models and playing within their own gaming club when they wish to, and few have responded to the request for help. None of those have answered any questions about the system, though in most cases they have provided other valuable information. It had become clear that to test this domain fully, monetary or other incentives would be necessary. Due to this limitation the results became quite limited. A part of this discussion is also reflected in Appendix F.

5.2.2 Explanations

While performing the experiment to evaluate the usefulness of explanations, two important points were discovered. Every user that had participated in the experiment had been, on average, satisfied with every explanation, even the lack of explanations. In contrast, the learning and confidence aspects were more varied. This may have been influenced by the system: the users were told the purpose of the system, and since the system fulfilled that purpose (creating an army in Warhammer 40k) the users were satisfied with the purpose, rather than the explanation.

It is clear that the level of satisfaction rose with increased learning and subsequently confidence, indicating that satisfaction has a base level, where a user is simply satisfied with the system. This in turn indicates that satisfaction may be a difficult metric to measure, as opposed to confidence and learning.

Secondly, the context of the explanations made the participants score the explanation higher overall. Providing more context in an explanation, as evident from the two different iterations of the second experiment (Section 4.2.2), increases the confidence, and more importantly, learning aspect for the users. While it can be overwhelming to put too much context, it is believed that in this specific domain context will only improve the explanation as the majority of those interested in the domain are motivated to read the lengthier explanations.

5.2.3 Maintenance

The utility maintenance result is excellent, as it shows that, at the very least, around 225 armies can be considered for in-depth retrieval. This has excellent implications for the accuracy of the system, as the number of armies largely increase the chance of selecting a successful solution army. However, selecting that many armies can be rather expensive, as the FAC step is about 10-15 times more expensive than the MAC step. This means that selecting 225 armies would take about 2.2 - 3.3 seconds at this stage, but much more if we expand and quantify more rules. Therefore, we should be wary about setting the k value too high in the future, as it may severely impede performance. In the state that the CBR system is in now, however, the performance is acceptable even with the k value set to 225.

For very large case bases the implementation allows both the MAC and the FAC steps of retrieval to be parallelized. Both of the steps require synchronization, however. In the case of the MAC step, the armies need to be synchronized correctly, so that the accuracy is completely preserved. For the FAC step, the similarity variable needs to be correctly synchronized between the threads or cores, depending on the choice of parallel programming. Otherwise, the armies are independent of each other and there is little other overhead, save for the need to copy the enemy army over in the FAC step should the parallel programming not utilize shared memory. This process should only be applied to case bases that have a large amount of cases and we need the system to be as accurate as possible. In this fashion, we could potentially increase the k threshold to very large values and allow our system to be very accurate in respects to its own case base.

While it is true that we may reach the utility problem, we will only reach it if we set the k-value to a high number. In fact, we can control the utility

problem, or even allow a decrease in performance for an increase in accuracy. As the filtering process is very cheap, it would take a significant amount of armies (in the thousands, or perhaps tens of thousands of cases) to decrease the performance. Thus, utility maintenance is of no true need. In the future, an option to manually or automatically delete cases could be present, to allow the user more control over the case base, but a triggered maintenance policy is not necessary. Furthermore, allowing the user to choose the k value may be a good solution to the performance versus accuracy problem and may increase the overall satisfaction with the system.

Finally, we must be cautious about the metagame maintenance, and the adjustment of strategy. The maintenance experiment was executed in three instance of ten matches each, which adjusted the weights from {1.1, 0.9, 1.1, 0.9} to {1.1, 0.9, 1.09, 0.92} for movement, psychic, shooting and assault respectively. If the experiment was instead executed in one instance of thirty matches, the weights would instead be adjusted to {1.08, 0.9, 1.1, 0.93}. As it is expected that the metagame maintenance policy will run hundreds of times while the user is away, this strategy adjustment must be taken into account and carefully balanced, so that the system ends up with a strategy adjustment that helps accuracy instead of hindering it.

5.3 Contributions

The system as a whole is the main contribution of the thesis. Together with the Specialization Project, the thesis completes the scientific method and creates a documented and reproducible work, which is often overlooked (Cohen and Howe, 1988). The architecture of the system should be applicable to any similar systems, and we have shown that it is possible to utilize the architecture to create the system. Any domain, where the main task is to divide a set number of resources to solve a presented problem can utilize parts or the full design and implementation that is presented in this thesis.

We have shown that JSON is capable of being used as a notation for cases. This is a significant step up from SQL, XML and CSV databases, as it allows us to model objects that have arrays of objects within them. In a simple example, a car can be black, red and white, and a number of colours greater than three could immediately in the CBR system indicate that a car is a sports car, instead of needing to have another variable that indicates such. While we have used a external solution (in the form of the Mongo Database)

to create an effective testing environment, in essence JSON requires nothing more than text files, eliminating the need for external resources.

We have shown that a maintenance policy can be used to evolve the system, by using a simulation. Any domain in which the cases can both be predicted and simulated could use the maintenance policies to fill the case base with more data. Thus it can not only solve problems more accurately, but also capture nuances that could be difficult for humans to see when dealing with the said domain.

The metagame maintenance policy is also capable of adjusting the strategy and accuracy of the domain, matching it to the users environment and results. Moreover, it is a proactive method of maintaining the CBR system and can produce results without the user specifically asking for them.

Finally, we have shown that in this domain instructive explanations have a larger merit than non-instructive explanations, and that a users satisfaction of an explanation may not be the best metric to measure the usefulness of it. This is in agreement with the explanation goals presented by Sormo et al. (2005), and may be an indication to approach complex domains from a contextual and learning aspect first, and satisfaction and confidence second.

Chapter 6

Conclusion and Future Work

This chapter concludes the thesis, both in respects to the goals of the thesis and as a whole, as well as outlines some of the possibilities to expand the thesis in the future. Section 6.1 discusses the goals of the thesis again, as presented in Section 1.1, and their completion or lack of. Section 6.2 concludes the thesis while Section 6.3 outlines the future work that can be performed to expand the thesis further.

6.1 Goals

Section 1.1 stated and described the goals of the thesis. Here we will look at and discuss if we have met the goals, and what were the significant shortcomings in relation to each goal.

Goal 1 Develop and test the CBR system for building an army in Warhammer 40k

We have developed a CBR system that is capable of picking an army that will in 70% of the cases not lose. While this does not correlate with the goal that we have set out with, it does show the somewhat unpredictable nature of the game. The system is also somewhat inaccurate, but it is capable of utilizing many of the basic rules of the game, and some advanced rules. It is clear that many of the rules will need to be quantified in order to improve the accuracy of the system.

Goal 2 Evaluate the usefulness of explanations in the system

We have managed to create explanations within the system and to evaluate their usefulness. Explanations were found to be very useful overall, and contrary to normal theory, transparency explanations have been found to be the most satisfactory. However, these explanations also hold a large amount of context and instructiveness and teach the user about the system. Learning about the system through explanations has raised both the confidence and satisfaction of the users using the system. Both novice users and expert users benefited from the explanations generated by the system in comparison to having no explanations generated by the system.

Goal 3 Test the application of maintenance policies to the evolution and maintenance of the system within the Warhammer 40k domain

We have created two out of the three maintenance policies, as there was no need to implement the update maintenance. The utility maintenance performed its function correctly, however, as it is fully tied with the k-value of the new retrieval step, it is not a necessary maintenance policy for the system. Instead, the user should be able to dictate how many armies they would like retrieved and through that indicate the levels of accuracy and performance they are willing to accept.

The metagame maintenance successfully expanded the squad case base by 63%, from 54 to 88 squads. This was done with manual simulation substituting automatic simulation, as automatic simulation was found to be impossible to implement accurately within the time frame of the thesis. The accuracy of the metagame maintenance was somewhat lacking, much like it was of the CBR part of the system, and improving the CBR part of the system will also improve the metagame policy as well. If automatic simulation is applied in the future, the system will be able to expand the case base on its own. Furthermore, we have shown that the maintenance is capable of adjusting the weights, and therefore the strategy, of the system.

6.2 Conclusion

This thesis has set out to create an explanation aware CBR system for army creation in Warhammer 40k, as well as attempt to utilize maintenance policies to proactively improve the system. It has succeeded in two out of three goals it has set out to do, and the problems with the third goal were identified and documented. It has done this by following the scientific method, allowing for replication of results as well as the reuse of ideas. While the final results of the thesis are not ideal, this thesis is a first pass at this domain in this fashion.

The thesis served to identify problems and present solutions, not only for this domain, but also for similar instances of the problem. While the case base, explanation system and the maintenance policies are not applicable directly to another domain, the methods that they use can be re-calibrated to another domain. We can not say with certainty that the system will work in other domains, as replication will be necessary to prove it, but we can say that the system can be re-purposed to work with other domains. Furthermore, CBR and explanation aware computing have been used in many applications already, as we have already discussed and we do not expect the same amount of bias when it comes to re-purposing those parts of the system to the class of problems. As for the metagame maintenance policy, we have insured that the selection of results was absolutely random, and that there could be no bias due to performing the experiment. Thus, it should also be possible to re-purpose the metagame maintenance to other systems, provided that the system can be simulated, as per the limitations.

6.3 Future Work

There are several next steps that the work presented here can head to. If the goal is to increase the accuracy of the system, work must be performed involving the Warhammer 40k game. Specifically, data needs to be collected about the various complex rules of the game, quantified and parametrized to provide a more robust view of the domain. Care should be taken to quantify variables above a threshold of certainty, as that could have an averse effect on the accuracy.

If the goal is to try to capture more of the domain, the general knowledge should be expanded with more factions, and the faction balance should be quantified. Care should be taken to not deteriorate the accuracy, as including new factions will also entail includes new rules.

If the goal is to complete the automation of the maintenance policies, a simulation will need to be created. This work is quite separate from the main CBR system and due to the modular nature of the architecture, it will be easy to integrate. The simulation must be accurate and have an AI of its own to control it, so that it may accurately replicate the game, and therefore not lower the accuracy of the system.

Replication of the system on another domain is external to the ideas and goals represented in this thesis, but is vital to the expansion of the scientific contribution of the system. Another domain could be another tabletop game, such as Warhammer 30k or Warmachine. These domains would be preferable as they are fairly similar and if the system can not be replicated in these domains, then it most certainly can not be replicated in more distant domains.

Finally, even though performance is still not considered a main issue, the system will need to be constantly optimized if we are to move to any of these goals. For very large case bases, one of the possible ways to achieve this would be to utilize parallel programming in retrieval.

Bibliography

[1] Schank, R.C. 1982 *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, New York, NY: Cambridge University Press.

[2] Ramon Lopez de Mantaras et al, May/June 2006 *Retrieval, reuse, revision, and retention in case-based reasoning*, IEEE Intelligent Systems, pp. 39-49.

[3] Thomas R. Roth-Berghofer, 2004 *Explanations and Case-Based Reasoning: Foundational Issues*, P. Funk and P.A. Gonzales Calero (Eds.): ECCBR 2004, LNAI 3155, pp. 389-403. Copyrighted by Springer-Verlag Berlin Heidelberg 2004

[4] Richter and Weber, 2013 *Case-Based Reasoning*, eBook, Springer Heidelberg New York Dordrecht London Copyrighted by Springer-Verlag Berlin Heidelberg 2013

[5] Cohen and Howe, 1988 *How Evaluation Guides AI Research.* AI Magazine Volume 9 Number 4 (1988), Copyrighted by AAAI.

[6] Schank, R.C 1986 *Explanation Patterns: Understanding Mechanically and Creatively*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[7] Johanna D. Moore and William R. Swartout 1988 *Explanation in expert systems: A survey*, Research Report RR-88-228, University of Southern California, Marina Del Rey, CA.

[8] McSherry, D. 2003 ***Similarity and Compromise***. *In proceedings of the Fifth International Conference on Case-Based Reasoning*, Berlin: Springer, pp. 291-305.

[9] Wizards of the Coast 2014, *Warhammer 40k, 7th Edition*, Games Workshop Ltd. Willow Road, Lenton, Nottingham, NG7 2WS, Copyrighted by Wizards of the coast and associates.

[10] Aamodt A. and Plaza E. 1994 *Case-based reasoning, foundational issues, methodological variations, and system approaches*, AI Communications 7(1), 39-59

[11] Aamodt A. 2004 ***Knowledge-intensive case-based reasoning in Creek***. *In Proceedings of the Seventh European Conference on Case-Based Reasoning*, Berlin: Springer, pp. 1-15

[12] Peter Spieker. 1991 *Naturlichsprachliche Erklarungen in technischen Expertensystemen*, Dissertation, University of Kaiserslautern.

[13] Anders Kofod-Petersen, October 8 2004, *How to do a Structured Literature Review in computer science.*

[14] Tintarev N. and Masthoff J. 2012, *Evaluating the effectiveness of explanations for recommender systems*, copyrighted by Springer Science+Business Media B.V.

[15] Vig J. and Sen S. and Riedl J. 2009, *Tagsplanations: Explaining Recommendations Using Tags*, IUI'09, Sanibel Island, Florida, USA.

[16] Cleger S. and Fernandez-Luna J.M and Huete J. F. 2014, *Learning from explanations in recommender systems*, Elsevier Inc. All rights reserved.

[17] Smyth B. and Keane M. T. 1995, ***Remembering to forget: A comptence-preserving case deletion policy for case-based reasoning systems.*** *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* San Mateo, CA: Morgan Kaufmann, pp. 377-383.

[18] Smyth B. and McClave P. 2001 ***Similarity vs. diversity.*** *In Proceedings of the Fourth International Computational Intelligence 17(2), 196-213.*

[19] Glinz M. 2007 *On Non-Functional Requirements*, 15th IEEE International Requirements Engineering Conference, New Delhi, India.

[20] Tidemann A. and Bjornson O. and Aamodt A. 2011 *Case-Based Reasoning in a System Architecture for Intelligent Fish Farming*, a joint work of SINTEF Fisheries and Aquaculture AS Trondheim, Norway and the Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway

[21] Sormo, F. and Cassens, J. and Aamodt, A. 2005 *Explanation in Case-Based Reasoning–Perspectives and Goals*, Artificial Intelligence Review (2005) 24: 109–143.

[22] Bass, L. and Clements. P, and Kazman, R. 2013 *Software Architecture in Practice: Third Edition*, Copyrighted by Pearson Education, Inc.

[23] Zhang, Z. and Yang, Q, 1998 *Towards lifetime maintenance of case base indexes for continual case based reasoning*, Volume 1480 of the series Lecture Notes in Computer Science pp 489-500.

[24] Leake, D. B. and Wilson, D. C, 1998 *Categorizing case-base maintenance: Dimensions and Directions*, Volume 1488 of the series Lecture Notes in Computer Science pp 196-207.

[25] Roth-Berghofer, T. and Iglezakis, I. 2001 *Six Steps in Case-Based Reasoning: Towards a maintenance methodology for case-based reasoning systems*, Includes the Proceedings of the 9th German Workshop on Case-Based Reasoning.

[26] Roth-Berghofer, T. and Cassens, J. 2005 *Mapping Goals and Kinds of Explanations to the Knowledge Containers of Case-Based Reasoning Systems*, Volume 3620 of the series Lecture Notes in Computer Science pp 451-464.

[27] Cunningham, P. and Doyle, D. and Loughrey, J. 2003 *An Evaluation of the Usefulness of Case-Based Explanation*, K.D. Ashley and D.G. Bridge (Eds.): ICCBR 2003, LNAI 2689, pp. 122–130, 2003. © Springer-Verlag Berlin Heidelberg.

[28] Diaz-Agudo, B. and Gonzales-Calero, P.A. 2000 *An Architecture for Knowledge Intensive CBR Systems*, E. Blanzieri and L. Portinale (Eds.): EWCBR 2000, LNAI 1898, pp. 37–48, 2000. © Springer-Verlag Berlin Heidelberg.

[29] Aamodt, A. 1994 *Knowledge-Intensive Case-Based Reasoning and Intelligent Tutoring.*

[30] Aamodt, A and Nygard, M. 1995 *Different roles and mutual dependencies of data, information, and knowledge — An AI perspective on their integration*, © 1995 Elsevier Science B.V. All rights reserved

[31] Park, C.S and Han, I. 2002 *A case-based reasoning with the feature weights derived by analytic hierarchy process for bankruptcy prediction*, © 2002 Elsevier Science Ltd. All rights reserved.

[32] Watson, I. 1999 *Case-based reasoning is a methodology not a technology*, © 1999 Elsevier Science B.V. All rights reserved

[33] Aamodt, A. 1994 *Explanation-driven case-based reasoning.*

[34] Aamodt, A. 1990 *Knowledge-Intensive Case-Based Reasoning and Sustained Learning*, Published in ECAI-90, Proceedings of the 9th European Conference on Artificial Intelligence, edited by Luigia Aiello, Stockholm, August, 6-10,1990. Pitman Publishing, London, 1990. Pages 1-6.

[35] Zikic, N. 2015 *Explanation-aware army builder for Warhammer 40k*, Specialization Project performed in the Autumn Semester of 2015 at the IDI department at NTNU.

Appendix

Appendix A - Glossary

AI - Artificial Intelligence
CBR - Case Based Reasoning
MAC/FAC - Many are called/Few are chosen retrieval method.
SQL - Structured Query Language. Used in relational databases for communication.
CSV - Comma Separated Values. Commonly used for simple case bases or databases.
XML - Extensible Markup Language. Much like CSV, commonly used for simple case bases or databases.
SME - Structure Mapping Engine. It is an implementation of an algorithm based on psychological theory of Dedre Gentner, used to calculate similarity when performing retrieval.
JSON - JavaScript Object Notation. A Non-SQL database solution written in purely string format.
BSON - Binary JSON. A format specific to MongoDB.
AR - Army Ratio. Used in Retrieval as one of the components for calculating the similarity index.
SqR - Squad Ratio. Used in Retrieval as one of the components for calculating the similarity index.
SR - Strength Ratio. Used in Retrieval as one of the components for calculating the similarity index.
FR - Faction Ratio. Used in Retrieval as one of the components for calculating the similarity index.
MP - Movement Phase. Used in calculating the Strength Ratio in the retrieval step.

PP - Psychic Phase. Used in calculating the Strength Ratio in the retrieval step.
SPP - Solution Psychic Power. Used in calculating the Psychic Phase in the retrieval step.
PrPP - Problem Psychic Power. Used in calculating the Psychic Phase in the retrieval step.
SP - Shooting Phase. Used in calculating the Strength Ratio in the retrieval step.
AsP - Assault Phase. Used in calculating the Strength Ratio in the retrieval step.
hbc - has a better calculated, used in explanations.
NTNU - Norwegian University of Science and Technology/Norges teknisk-naturvitenskapelige universitet
Warhammer 40k - Warhammer 40000, often referred to in text as Warhammer 40k, as it is easier to read

Appendix B - Software Used

Latex, a word processor and document markup language used to write this document. www.sharelatex.com was used to write the document. First accessed on Janury 19, 2016.

IntelliJ IDEA, a Integrated Developer Environment (IDE) for JAVA. First accessed on January 19, 2016.

MongoDB, a Non-SQL database. First accessed on January 19, 2016.

Mongo Plugin by David Boissier, a plugin to allow usage of MongoDB in the IDEA IDE. First accessed on January 20, 2016.

draw.io to draw flowcharts, from https://www.draw.io/. First accessed on January 25, 2016.

Battlescribe, used to create armies, as well as test their validity. Used in the first and second experiment. First accessed on April 2, 2016.

Appendix C - Interview With Experts

The interview with the experts was conducted by email and the questions are presented here. The combined answers are also presented below each question, though specific or contradicting answers are presented as well. The experts that were asked are all members or frequent visitors of the local wargaming club, Wartrond. They in no way refer to themselves as experts, but are rather players with experience with the game over a course of several months or years.

What is better in your opinion: An army prepared for (nearly) everything, or a specialized army?

For a tournament format, it is better to be prepared for everything. For casual gameplay both are fine.

In your experience, is a balance of vehicles and infantry better than more infantry/vehicle focused armies?

A balance is usually the best.

Alliances - How often are they used, and does anyone use anything besides battle brothers[1]?

Alliances are often used, even outside of Battle Brothers.

Would you say that some factions have advantages over other factions? For example Orks are better than Tau, etc. And is this consistent between factions or are there only a few examples?

There is a large unbalance between factions in Warhammer 40k.

I think that's a huge problem with 40k these days, and one of the reasons so many people are transferring over to 30k/Horus Heresy setting instead.

This is what turned me away from 40k. The imbalances are so clear, that two friends agreeing to have a friendly, balanced game, may have real problems with it. If friend A plays with his Eldar against friend B, who plays Orks, A

[1]Battle brothers is an alliance rule that imposes on penalties on the armies.

will almost always win, even if they make the lists together for balance. It's horrible!

How often do you do the run move in shooting? Is it very important/not so important?

Depends on the army, but usually not often. Some factions may have exceptions.

What are the most important phases of a player turn from top to bottom? Excluding start of turn and end of turn phases, so shooting, psychic, movement and assault.

Movement, Shooting, Assault and Psychic, in that order. Of course it depends on armies, and some armies can have great advantages in other phases.

Movement, shooting, assault, psychic, in my opinion. In 40k I think the psychic would be more important than the assault phase though, since they have more opportunities to make "stacking" buffs and rules to make the dreaded "deathstar" units. (Effectively un-killable units due to poorly thought out rules that stack with psychic buffs).

How often do you customize your squads using squad options? Is this very common for most factions or just for some?

It is common to customize all the time for all factions.

Would you say that the advantages of detachments (battleforged armies) are worth having in comparison to unbound armies that have none of those advantages? Things like objective secured, etc.

Tournaments usually do not usually allow unbound armies. Unbound armies have a major disadvantage over battleforged armies.

Nobody plays unbound. It's an abomination.

Generally speaking, how much terrain is usually present in a normal battle?

Generally, this depends on the players. Many like more, but the majority agree that 20% to 30% is a rough average.

Finally, in your experience, how often did you lose just because of bad rolls on the dice? And vice versa, how often did you win because of good rolls?

Rarely if ever. Most people make up a strategy that does not revolve on luck, but rather on statistical probability.

Appendix D - JSON Represenations of Objects

A sample of all the JSON representations of objects in their raw format is presented here.

A piece of equipment is represented as:


```
{
        "Name":"Autocannon",
        "Range":48,
        "Strength":"7",
        "AP":4,
        "Type":[{"Name":"Heavy 2"}]
},
```


Figure D1: Equipment JSON representation

An infantry, bike, artillery, or in general a squad without armor is represented as:


```
{
        "Name":"Chaplain Grimaldus",
        "Units":[
                {
                "Name":"Chaplain Grimaldus",
                "Weapon Skill":5,
                "Ballistics Skill":4,
                "Strength":4,
                "Toughness":4,
                "Wounds":3,
                "Initiative":4,
                "Attacks":3,
                "Leadership":10,
                "Armour Save":3,
                "Unit Type":"Infantry-Character",
                "Unique":true,
                "Count":1,
                "Faction":"Space Marines",
                "Equipment":[{"Name":"Master-crafted Plasma Pistol", "Count":1},
                "Special Rules":[{"Name":"Chapter Tactics (Black Templars)"}, {"
                "Options":[{"Name":2}],
                "Warlord Trait":"Rites of War"
                },
                {
                "Name":"Cenobyte Servitor",
                "Weapon Skill":3,
                "Ballistics Skill":3,
                "Strength":3,
                "Toughness":3,
                "Wounds":1,
                "Initiative":3,
                "Attacks":1,
                "Leadership":8,
                "Armour Save":4,
                "Unit Type":"Infantry",
                "Unique":false,
                "Count":0,
                "Faction":"Space Marines",
                "Equipment":[{"Name":"None", "Count":0}],
                "Special Rules":[{"Name":"Relics of Helsreach"}],
                "Options":[{"Name":0}],
                "Warlord Trait":"None"
                }
                ],
        "Cost":150,
        "Role":"HQ",
        "Base":true,
        "Options":[{"Name":0}],
        "Parameters":[{"Name":"None"}],
        "Rating":1500
},
```


Figure D2: No Armor Squad JSON representation

Note that it is not possible to fit the entirety of the representation on an A4 sized page. For example, the equipment array for the Chaplain Grimaldus Squad is:

”Equipment”:[{”Name”:”Master-crafted Plasma Pistol”, ”Count”:1}, {”Name”:”Crozius Arcanum”, ”Count”:1}, {”Name”:”Frag Grenade”, ”Count”:1}, {”Name”:”Krak Grenade”, ”Count”:1}, {”Name”:”Rosarius”, ”Count”:1}]

A walker unit is represented as:


```
{
        "Name":"Dreadnoughts",
        "Units":[
                {
                "Name":"Dreadnought",
                "Weapon Skill":4,
                "Ballistics Skill":4,
                "Strength":6,
                "F":12,
                "S":12,
                "R":10,
                "Initiative":4,
                "Attacks":4,
                "HP":3,
                "Unit Type":"Vehicle-Walker",
                "Unique":false,
                "Count":1,
                "Faction":"Space Marines",
                "Equipment":[{"Name":"Multi-melta", "Count":1}, {"Name":"Pow
                "Special Rules":[{"Name":"Chapter Tactics"}],
                "Options":[{"Name":106}, {"Name":107}, {"Name":108}, {"Name"
                "Warlord Trait":"None"
                }
                ],
        "Cost":100,
        "Role":"Elites",
        "Base":true,
        "Options":[{"Name":0}],
        "Parameters":[{"Name":"None"}],
        "Rating":1500
},
```


Figure D3: Walker Squad JSON representation

Again, some of the unit may not be fully representable on an A4 sized page.

Finally, any vehicle unit other than a walker and special cavalry-like units (such as bikes) are represented as:


```
{
        "Name":"Land Speeders",
        "Units":[
                {
                "Name":"Land Speeder",
                "Ballistics Skill":4,
                "F":10,
                "S":10,
                "R":10,
                "HP":2,
                "Unit Type":"Vehicle-Skimmer-Fast",
                "Unique":false,
                "Count":1,
                "Faction":"Space Marines",
                "Equipment":[{"Name":"Heavy Bolter", "Count":1}],
                "Special Rules":[{"Name":"Anti-grav Upwash"}, {"N
                "Options":[{"Name":174}, {"Name":175}, {"Name":17
                "Warlord Trait":"None"
                }
                ],
        "Cost":45,
        "Role":"Fast Attack",
        "Base":true,
        "Options":[{"Name":0}],
        "Parameters":[{"Name":"None"}],
        "Rating":1500
},
```


Figure D4: Vehicle Squad JSON representation

Parts of the Army Class can be seen in the figure below.


```
{
        "ID":"2",
        "Name":"Decent Army 1",
        "Squads":[
                {
                "Name":"Techmarine",
                "Options":[{"Option":23}, {"Option":23}, {"Option":23},
                "Parameter":[{"Name":"None"}, {"Name":"None"}, {"Name":'
                },
                {
                "Name":"Tactical Squad",
                "Options":[{"Option":33}, {"Option":33}, {"Option":33},
                "Parameter":[{"Name":"None"}, {"Name":"None"}, {"Name":'
                },
                {
                "Name":"Tactical Squad",
                "Options":[{"Option":33}, {"Option":33}, {"Option":33},
                "Parameter":[{"Name":"None"}, {"Name":"None"}, {"Name":'
                },
                {
                "Name":"Dreadnoughts",
                "Options":[{"Option":0}],
                "Parameter":[{"Name":"None"}]
                },
                {
                "Name":"Dreadnoughts",
                "Options":[{"Option":0}],
                "Parameter":[{"Name":"None"}]
                },
                {
                "Name":"Sternguard Veteran Squad",
                "Options":[{"Option":0}],
                "Parameter":[{"Name":"None"}]
                },
                {
                "Name":"Stormtalon Gunship",
                "Options":[{"Option":0}],
                "Parameter":[{"Name":"None"}]
                },
                {
                "Name":"Thunderfire Cannons",
                "Options":[{"Option":216}, {"Option":216}],
                "Parameter":[{"Name":"None"}, {"Name":"None"}]
                },
                {
                "Name":"Vindicators",
                "Options":[{"Option":0}],
                "Parameter":[{"Name":"None"}]
                },
                {
```


Figure D5: Army JSON representation

Appendix E - Additional Experiment Data

Additional data from the experiments can be found here. The subsections here share the names of the experiments, as can be seen in Section 4.2. The information here is additional information that provides context, but is too lengthy to be placed in the thesis.

CBR system for Army Creation

All of the army compositions used in the testing of the system can be found here. There are a total of fourteen armies, the first ten are the armies in the case base, and the last four (denoted as Solution Armies) are the armies the system retrieved and then adapted when presented with the first ten armies as problem armies. Each table entry represents one of the squads of the army, and their equipment, as well as their number. No special rules are presented unless they have been somehow changed from the original rules, via. squad options for example. Similarly, no extra equipment is presented unless it too differs from the equipment presented in the rule books.

Army Composition - Army 1 - Cost = 1840 points
Captain Chapter Master Power Armour [Bolt Pistol, Chainsword]
Dreadnoughts [3x Dreadnought]
Ironclad Dreadnoughts [3x Ironclad Dreadnought]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Attack Bike Squad [2x Attack Bike]
Predators [3x Predator]

Table E1: E3 - Army 1

Army Composition - Army 2 - Cost = 1840 points
Captain Chapter Master Power Armour [Bolt Pistol, Chainsword]
Terminator Squad [9x Terminator, 1x Terminator Sergeant]
Terminator Squad [9x Terminator, 1x Terminator Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Razorback [Twin-linked Heavy Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Razorback [Twin-linked Heavy Bolter]
Land Speeders Land Speeder [Heavy Bolter, Multi-melta]
Land Speeders Land Speeder [Heavy Bolter, Multi-melta]
Stormraven Gunship
Thunderfire Cannons [3x Thunderfire Cannons, 3x Techmarine Gunner]

Table E2: E3 - Army 2

Army Composition - Army 3 - Cost = 1850 points
Pedro Kantor
Centurion Assault Squad [5x Centurion, 1x Centurion Sergeant]
Scout Squad [6x Scouts, 1x Scout Sergeant]
Tactical Squad [9x Space Marines] Rhino [Storm Bolter] Space Marine Sergeant [Bolt Pistol, Boltgun, Melta Bombs]
Tactical Squad [9x Space Marines] Rhino [Storm Bolter] Space Marine Sergeant [Bolt Pistol, Boltgun, Melta Bombs]
Tactical Squad [9x Space Marines] Rhino [Storm Bolter] Space Marine Sergeant [Bolt Pistol, Boltgun, Melta Bombs]
Assault Squad [9x Space Marine, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Bike Squad [7x Space Marine Bikers, 1x Biker Sergeant, 1x Attack Bike]
Centurion Devastator Squad [5x Centurion, 1x Centurion Sergeant]

Table E3: E3 - Army 3

Army Composition - Army 4 - Cost = 1850 points
Librarian [Mastery Level 2]
Terminator Squad [9x Terminator, 1x Terminator Sergeant]
Venerable Dreadnoughts Drop Pod [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Assault Squad [9x Space Marine, 1x Space Marine Sergeant]
Stormtalon Gunship
Devastator Squad [9x Space Marine, 1x Space Marine Sergeant]
Devastator Squad [9x Space Marine, 1x Space Marine Sergeant]

Table E4: E3 - Army 4

Army Composition - Army 5 - Cost = 1850 points
Chaplain
Librarian [Mastery Level 2]
Sternguard Veteran Squad [9x Veteran, 1x Veteran Sergeant]
Sternguard Veteran Squad [9x Veteran, 1x Veteran Sergeant]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Razorback [Twin-linked Heavy Bolter]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Razorback [Twin-linked Heavy Bolter]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Rhino [Storm Bolter]
Land Raider
Predators Predator [Twin-linked Lascannon] Predator [Twin-linked Lascannon]

Table E5: E3 - Army 5

Army Composition - Army 6 - Cost = 1850 points
Captain Chapter Master Power Armour [Bolt Pistol, Chainsword]
Sternguard Veteran Squad [9x Veteran, 1x Veteran Sergeant] All 9x Veterans have [Bolt Pistol, Storm Bolter] Razorback [Twin-linked Heavy Bolter]
Sternguard Veteran Squad [9x Veteran, 1x Veteran Sergeant] All 9x Veterans have [Bolt Pistol, Storm Bolter] Razorback [Twin-linked Heavy Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Vindicators [3x Vindicator]

Table E6: E3 - Army 6

Army Composition - Army 7 - Cost = 1845 points
Chief Librarian Tigurius
Legion of the Damned [9x Legionnaire, 1x Legionnaire Sergeant]
Terminator Squad [8x Terminator, 1x Terminator Sergeant]
Venerable Dreadnoughts [3x Venerable Dreadnought]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Rhino [Storm Bolter]
Devastator Squad [9x Space Marines, 1x Veteran Sergeant] Rhino [Storm Bolter]
Devastator Squad [9x Space Marines, 1x Veteran Sergeant] Rhino [Storm Bolter]

Table E7: E3 - Army 7

Army Composition - Army 8 - Cost = 1845 points
Vulkan He'stan
Dreadnoughts [3x Dreadnought]
Dreadnoughts [3x Dreadnought]
Scout Squad [9x Scouts, 1x Veteran Scout Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant]
Attack Bike Squad [3x Attack Bike] All 3x Attack Bikes have [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Attack Bike Squad [3x Attack Bike] All 3x Attack Bikes have [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Land Speeders Land Speeder [Assault Cannon, Multi-melta]

Table E8: E3 - Army 8

Army Composition - Army 9 - Cost = 1825 points
The Emperor's Champion
Ironclad Dreadnoughts [1x Ironclad Dreadnought] Drop Pod [Storm Bolter]
Terminator Assault Squad [9x Terminator, 1x Terminator Sergeant]
Terminator Assault Squad [9x Terminator, 1x Terminator Sergeant]
Crusader Squad [5x Initiate, 5x Neophyte] Rhino [Storm Bolter]
Crusader Squad [5x Initiate, 5x Neophyte] Rhino [Storm Bolter]
Crusader Squad [5x Initiate, 5x Neophyte] Rhino [Storm Bolter]
Devastator Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Devastator Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]

Table E9: E3 - Army 9

Army Composition - Army 10 - Cost = 1844 points
Kor'sarro Khan [Bolt Pistol, Moonfang] Moondrakkan [Twin-linked Boltgun]
Ironclad Dreadnoughts [1x Ironclad Dreadnought] Drop Pod [Storm Bolter]
Vanguard Veteran Squad [9x Veteran, 1x Veteran Sergeant] Rhino [Storm Bolter]
Vanguard Veteran Squad [9x Veteran, 1x Veteran Sergeant] Rhino [Storm Bolter]
Bike Squad [6x Space Marine Biker, 1x Biker Sergeant
Bike Squad [6x Space Marine Biker, 1x Biker Sergeant
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Rhino [Storm Bolter]
Attack Bike Squad [3x Attack Bike] Attack Bikes have [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Predators [3x Predator]
Predators [3x Predator]

Table E10: E3 - Army 10

Army Composition - Solution Army 1 - Cost = 1850 points
Kor'sarro Khan [Bolt Pistol, Moonfang] Moondrakkan [Twin-linked Boltgun]
Terminator Assault Squad [9x Terminator, 1x Terminator Sergeant]
Terminator Assault Squad [9x Terminator, 1x Terminator Sergeant]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Razorback [Twin-linked Heavy Bolter]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Razorback [Twin-linked Heavy Bolter]
Attack Bike Squad [2x Attack Bike] 1x Attack Bike has [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Predators [2x Predator] Both Predators have [Twin-linked Lascannon]
Thunderfire Cannons [3x Thunderfire Cannons, 3x Techmarine Gunner]

Table E11: E3 - Solution Army 1

Army Composition - Solution Army 2 - Cost = 1831 points
Kor'sarro Khan [Bolt Pistol, Moonfang] Moondrakkan [Twin-linked Boltgun]
Centurion Assault Squad [5x Centurion, 1x Centurion Sergeant]
Scout Squad [9x Scouts] One Scout has [Heavy Bolter], One other Scout has [Space Marine Shotgun] Scout Sergeant [Bolt Pistol, Boltgun, Melta Bombs]
Tactical Squad [9x Space Marines] Razorback [Twin-linked Heavy Bolter] Space Marine Sergeant [Bolt Pistol, Boltgun, Melta Bombs]
Tactical Squad [9x Space Marines, 1x Veteran Sergeant] Rhino [Storm Bolter]
Tactical Squad [9x Space Marines] Rhino [Storm Bolter] Space Marine Sergeant [Bolt Pistol, Boltgun, Melta Bombs]
Attack Bike Squad [3x Attack Bike] Attack Bikes have [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Bike Squad [7x Space Marine Bikers, 1x Biker Sergeant, 1x Attack Bike]
Centurion Devastator Squad [5x Centurion, 1x Centurion Sergeant]

Table E12: E3 - Solution Army 2

Army Composition - Solution Army 3 - Cost = 1840 points
Sergeant Telion
Terminator Squad [9x Terminator, 1x Terminator Sergeant]
Terminator Squad [9x Terminator, 1x Terminator Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Razorback [Twin-linked Heavy Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Razorback [Twin-linked Heavy Bolter]
Attack Bike Squad [3x Attack Bike] Attack Bikes have [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Attack Bike Squad [1x Attack Bike] Attack Bike has [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Stormraven Gunship
Thunderfire Cannons [3x Thunderfire Cannons, 3x Techmarine Gunner]

Table E13: E3 - Solution Army 3

Army Composition - Solution Army 4 - Cost = 1840 points
Kor'sarro Khan [Bolt Pistol, Moonfang] Moondrakkan [Twin-linked Boltgun]
Terminator Assault Squad [9x Terminator, 1x Terminator Sergeant]
Terminator Assault Squad [9x Terminator, 1x Terminator Sergeant]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Razorback [Twin-linked Heavy Bolter]
Tactical Squad [9x Space Marines, 1x Space Marine Sergeant] Razorback [Twin-linked Heavy Bolter]
Attack Bike Squad [1x Attack Bike] Attack Bike has [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Attack Bike Squad [1x Attack Bike] Attack Bike has [Bolt Pistol, Multi-melta, Twin-Linked Boltgun]
Predators [2x Predator] Both Predators have [Twin-linked Lascannon]
Thunderfire Cannons [3x Thunderfire Cannons, 3x Techmarine Gunner]

Table E14: E3 - Solution Army 4

Application of maintenance policies

Each match here follows the same ID as presented in Subsection 4.2.3. Each Army has a dash and a number next to it to showcase deployment on the playing field. A star indicates that this army had won the initiative and had acted first. The numbers within the brackets show what options the system has applied to the squad. These are stored numerically in the case base to allow easier implementation with the general domain knowledge, since writing each option in text is considerably lengthier and the only advantage it provides is already reflected in the detailed army composition.

The battle table is shown in Figure E1. Following that, the thirty individual army compositions generated by the maintenance policy are shown in individual tables.

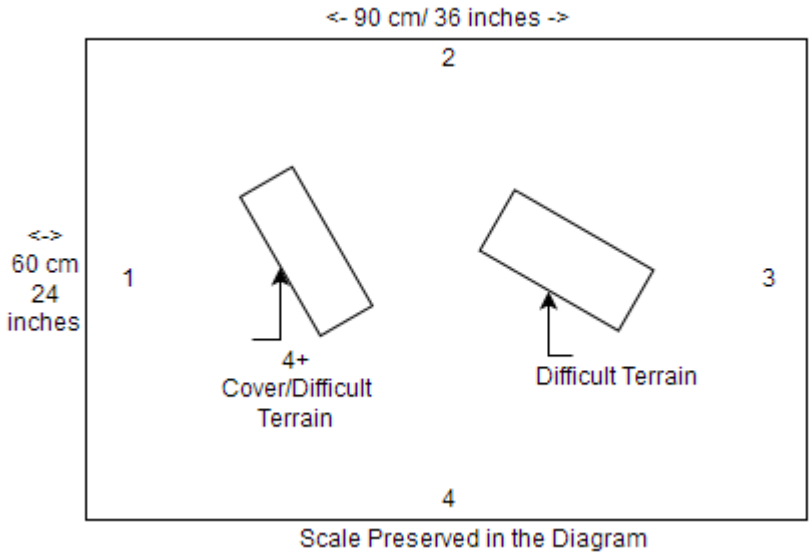

Figure E1: Battle table used in the Maintenance Policy Experiment

ID	Army 1 - 1 *	Outcome	New Rank
1	Vanguard Veteran Squad (97)	Win	1505
1	Dreadnoughts	Win	1505
ID	Army 2 - 2	Outcome	New Rank
1	Land Speeders (189,195)	Loss	1495
1	Vindicators	Loss	1495

Table E15: E3 - T1M1

ID	Army 1 - 2 *	Outcome	New Rank
2	Drop Pod	Win	1502
2	Tactical Squad (38, 40)	Win	1502
2	Techmarine (23)	Win	1502
ID	Army 2 - 3	Outcome	New Rank
2	Chaplain	Loss	1498
2	Librarian	Loss	1498

Table E16: E3 - T1M2

ID	Army 1 - 2 *	Outcome	New Rank
3	The Emperor's Champion	Win	1503
3	Drop Pod	Win	1505
ID	Army 2 - 3	Outcome	New Rank
3	Sergeant Telion	Loss	1497
3	Scout Squad (45)	Loss	1497
3	Honour Guard (75)	Loss	1497

Table E17: E3 - T1M3

ID	Army 1 - 2	Outcome	New Rank
4	Scout Squad 2 (45)	Draw	1497
4	Assault Squad	Draw	1500
4	Techmarine 2 (23)	Win	1508
ID	Army 2 - 3 *	Outcome	New Rank
4	Honour Guard 2 (75)	Loss	1490
4	Vanguard Veteran Squad (83, 97)	Loss	1494

Table E18: E3 - T1M4

ID	Army 1 - 1 *	Outcome	New Rank
5	Terminator Squad (156)	Loss	1497
ID	Army 2 - 2	Outcome	New Rank
5	Venerable Dreadnoughts	Win	1502
5	Scout Bike Squad (168)	Draw	1497

Table E19: E3 - T1M5

ID	Army 1 - 4	Outcome	New Rank
6	Honour Guard	Loss	1494
6	Sternguard Veteran Squad (102)	Loss	1494
ID	Army 2 - 2 *	Outcome	New Rank
6	Pedro Kantor	Win	1506

Table E20: E3 - T1M6

ID	Army 1 - 2	Outcome	New Rank
7	Sternguard Veteran Squad 2 (102, 106, 120)	Loss	1486
7	Bike Squad (176, 180)	Loss	1493
ID	Army 2 - 1 *	Outcome	New Rank
7	Pedro Kantor	Win	1513

Table E21: E3 - T1M7

ID	Army 1 - 2 *	Outcome	New Rank
8	Scout Bike Squad (173)	Draw	1498
8	Dreadnoughts	Win	1508
8	Rhino	Win	1503
ID	Army 2 - 4	Outcome	New Rank
8	The Emperor's Champion	Loss	1500
8	Razorback	Loss	1497

Table E22: E3 - T1M8

ID	Army 1 - 3 *	Outcome	New Rank
9	Devastator Squad	Win	1506
9	Chaplain Cassius	Win	1506
ID	Army 2 - 2	Outcome	New Rank
9	Vanguard Veteran Squad (93)	Loss	1493
9	Rhino	Loss	1496
9	Razorback	Loss	1490

Table E23: E3 - T1M9

ID	Army 1 - 4	Outcome	New Rank
10	Command Squad	Win	1507
10	Scout Squad (45, 46)	Draw	1501
10	Sergeant Chronus	Win	1507
ID	Army 2 - 3 *	Outcome	New Rank
10	Captain	Loss	1493
10	Attack Bike Squad	Loss	1493
10	Land Speeders 2 (189,195)	Loss	1488

Table E24: E3 - T1M10

ID	Army 1 - 3	Outcome	New Rank
11	Razorback (201)	Loss	1485
11	Sternguard Veteran Squad (102, 106)	Loss	1494
ID	Army 2 - 4 *	Outcome	New Rank
11	Terminator Assault Squad 2 (156)	Win	1504

Table E25: E3 - T2M11

ID	Army 1 - 3 *	Outcome	New Rank
12	Scout Bike Squad	Loss	1496
12	Honour Guard 2 (75)	Loss	1490
12	Sergeant Telion	Loss	1493
ID	Army 2 - 1	Outcome	New Rank
12	Honour Guard 2 (75)	Win	1490
12	Bike Squad (176, 181)	Win	1504

Table E26: E3 - T2M12

ID	Army 1 - 1	Outcome	New Rank
13	Bike Squad 7 (176,180,181)	Draw	1502
13	Sergeant Chronus	Win	1510
13	Attack Bike Squad	Draw	1491
ID	Army 2 - 3 *	Outcome	New Rank
13	Tactical Squad (40)	Loss	1496
13	Bike Squad 5 (176, 180)	Loss	1489
13	Scout Squad (45, 46)	Loss	1496

Table E27: E3 - T2M13

ID	Army 1 - 3	Outcome	New Rank
14	Tactical Squad 3 2 (35, 40)	Win	1502
14	Land Speeders (190,192)	Draw	1500
14	Sergeant Chronus	Win	1516
ID	Army 2 - 1 *	Outcome	New Rank
14	Chaplain	Loss	1492
14	Scout Bike Squad (168,168,171,173)	Loss	1491

Table E28: E3 - T2M14

ID	Army 1 - 4	Outcome	New Rank
15	Tactical Squad 3 2 (35, 40)	Win	1506
15	Tactical Squad 3 (40)	Draw	1495
15	Rhino	Win	1501
ID	Army 2 - 2 *	Outcome	New Rank
15	Assault Squad (157,159,160,166)	Loss	1496
15	Sergeant Telion	Loss	1488
15	Drop Pod	Loss	1501

Table E29: E3 - T2M15

ID	Army 1 - 2	Outcome	New Rank
16	Legion of the Damned	Win	1506
16	Predators	Win	1506
ID	Army 2 - 3 *	Outcome	New Rank
16	Venerable Dreadnoughts	Loss	1495
16	Scout Bike Squad 2 (168)	Loss	1490

Table E30: E3 - T2M16

ID	Army 1 - 1 *	Outcome	New Rank
17	The Emperor's Champion	Loss	1498
17	Scout Squad (45, 53)	Loss	1498
ID	Army 2 - 3	Outcome	New Rank
17	Devastator Squad	Draw	1504
17	Drop Pod	Win	1504
17	Honour Guard 2 (75)	Win	1493

Table E31: E3 - T2M17

ID	Army 1 - 3 *	Outcome	New Rank
18	Command Squad	Win	1513
18	Scout Squad 3 (43,45,46)	Win	1502
18	Attack Bike Squad	Win	1497
ID	Army 2 - 2	Outcome	New Rank
18	Bike Squad 7 (176,178,180,181)	Loss	1498
18	Tactical Squad 3 (40)	Loss	1490

Table E32: E3 - T2M18

ID	Army 1 - 1 *	Outcome	New Rank
19	Kor'sarro Khan	Win	1505
19	Librarian	Win	1502
ID	Army 2 - 3	Outcome	New Rank
19	Shadow Captain Shrike	Loss	1495

Table E33: E3 - T2M19

ID	Army 1 - 1	Outcome	New Rank
20	Scout Bike Squad 4 (168, 168, 168, 171, 173)	Loss	1485
20	Razorback 2 (201)	Loss	1482
ID	Army 2 - 2 *	Outcome	New Rank
20	Rhino	Win	1506
20	Razorback 2 (201)	Draw	1482
20	Techmarine (23, 25)	Win	1505

Table E34: E3 - T2M20

ID	Army 1 - 1 *	Outcome	New Rank
21	Scout Bike Squad 4 2 (168, 168, 171, 173)	Draw	1489
21	Drop Pod	Win	1507
21	Scout Squad (45, 53)	Draw	1499
ID	Army 2 - 2	Outcome	New Rank
21	Tactical Squad 2 3 (35,35,40)	Loss	1503
21	Techmarine 2 (23, 26)	Loss	1504

Table E35: E3 - T3M21

ID	Army 1 - 2 *	Outcome	New Rank
22	Terminator Assault Squad 2 (156)	Loss	1496
ID	Army 2 - 1	Outcome	New Rank
22	The Emperor's Champion	Win	1502
22	Sergeant Chronus	Win	1520

Table E36: E3 - T3M22

ID	Army 1 - 2 *	Outcome	New Rank
23	Scout Bike Squad 4 (168, 168, 168, 171, 173)	Loss	1494
23	Whirlwinds	Loss	1483
ID	Army 2 - 1	Outcome	New Rank
23	Scout Bike Squad	Win	1502
23	Command Squad	Draw	1513
23	Scout Squad (45, 45, 46, 53)	Win	1505

Table E37: E3 - T3M23

ID	Army 1 - 4 *	Outcome	New Rank
24	Vulkan He'stan	Win	1506
ID	Army 2 - 2	Outcome	New Rank
24	Scout Squad 3 2 (43,45,46)	Loss	1496
24	Sergeant Chronus	Loss	1514
24	Scout Bike Squad (168, 171)	Loss	1496

Table E38: E3 - T3M24

ID	Army 1 - 1 *	Outcome	New Rank
25	Captain Sicarius	Win	1503
ID	Army 2 - 3	Outcome	New Rank
25	Shadow Captain Shrike	Loss	1492

Table E39: E3 - T3M25

ID	Army 1 - 3	Outcome	New Rank
26	Thunderfire Cannons	Win	1506
26	Honour Guard 2 (75)	Draw	1494
ID	Army 2 - 1 *	Outcome	New Rank
26	Centurion Assault Squad (78)	Loss	1494
26	Drop Pod	Loss	1502

Table E40: E3 - T3M26

ID	Army 1 - 1 *	Outcome	New Rank
27	Sternguard Veteran Squad 4 (102, 103, 106)	Loss	1489
27	Librarian	Loss	1497
ID	Army 2 - 3	Outcome	New Rank
27	Scout Squad (43, 45, 46)	Win	1505
27	Scout Squad 3 2 (43, 43, 45, 46)	Win	1501
27	Razorback	Win	1496

Table E41: E3 - T3M27

ID	Army 1 - 2	Outcome	New Rank
28	Terminator Squad (151, 152)	Win	1505
ID	Army 2 - 1 *	Outcome	New Rank
28	Tactical Squad (40)	Loss	1494
28	Scout Squad 6 (43, 45, 46, 47, 51)	Loss	1499
28	Land Speeder Storm	Loss	1494

Table E42: E3 - T3M28

ID	Army 1 - 2 *	Outcome	New Rank
29	Scout Bike Squad 4 (168, 168, 168, 171, 173)	Win	1494
29	Librarian	Win	1502
ID	Army 2 - 3	Outcome	New Rank
29	Terminator Squad 2 (151, 152)	Loss	1500

Table E43: E3 - T3M29

ID	Army 1 - 2 *	Outcome	New Rank
30	Scout Bike Squad (173)	Loss	1497
30	Dreadnoughts	Loss	1503
30	Land Speeder Storm	Loss	1489
ID	Army 2 - 4	Outcome	New Rank
30	Vulkan He'stan	Win	1513

Table E44: E3 - T3M30

Appendix F - Personal Reflection

While I would not say that this thesis was *too much* for one person, it is my opinion that it would be greatly improved by the addition of one or two people. My supervisor (Anders Kofod-Petersen) had initially stated that the project (Specialization Project, 2015) and thesis are more suited for two people and at the end of the thesis I wholeheartedly agree with him. I believe that many of the limitations that were caused by complexity and time constraints could be avoided. I can not say if this would lead to more problems or limitations, but the implementation itself, as well as performing the experiments, did take a significant amount of time. A major part of the work on the implementation could have been performed in parallel, and a more creative case base could have definitively been made. Furthermore, experimentation would have been more robust and easier to perform. I do not regret working on the thesis, however, as I feel I have expanded my knowledge substantially and I have been very motivated throughout the length of the project and the thesis.

Another important point to reflect on is the experimentation. I believe that I should have discussed the experimentation plan with those involved at the beginning of the project, and begun my search for participants much sooner than I have. I had looked for participants at the start of the Master Thesis, and I feel that the few extra months that I had to work on the Specialization Project would have helped me get acquainted with a few more participants, and have a more varied set of result data. Any external resources (such as money), would have been a great help at securing more participants for experimentation and I encourage those that attempt to reproduce the results or attempt a different approach to seek out participants earlier rather than later should the external resources not be available.

Finally, even though I was aware of this from before, setting milestones for the thesis was of the utmost importance. Being able to gauge whether or not the implementation would be finished and what kind of limitations needed to be placed on the implementation was most likely the sole reason why I was able to finish the thesis in good time. Planning out the project was, I believe, as important as writing and implementing it, with regards to the time constraints.

Appendix G - The Rating System

This section is mostly present in the Appendix C of the Specialization Project (Zikic, 2015), and is only included here for convenience and minor updates.

The rating attribute of armies and squads is calculated using the same method that is used for chess players and the calculation of elo ratings. The K number in this particular explanation is 40, but it usually varies between 20 and 40.

Armies and Squads are introduced into the system at an even rating. This rating is a number that equals 1500. The ratings are adjusted identically for the squads and the armies but with different K values, as described in the thesis.

The rating goes up if a win is recorded. Consequently, the rating goes down if a loss is recorded. If a draw is recorded the rating will either stay the same, go up or down, depending on the ratings. If an army draws with an army higher rated than it, it will go up in ratings.

Due to the fact that a squad is very likely to engage other, varied rated squads, the squads rating is taken as an average across all squads in an army. Therefore, there may be some individual miscalculations in the rating system, however as a whole and over time, these number of these miscalculations should approach zero as the system rates squads closer and closer to their actual worth.

The rating system does not increase linearly. The bigger the difference is in between the armies, the less points are gained for the winning army if that army was the higher rated army. At equal rating, a win is considered 20 rating points, a loss is considered -20 points, and a draw is considered 0 points. A rating that differs by 400 points is the maximum difference that the system calculates. Any larger rating differences are set back to 400. At this rating, a win for the higher rated army is only 3.2 points, while a loss is 36.8 points. This is done to prevent any army from reaching incredible ratings by fighting only weak rated armies.