DEVELOPMENT OF AN EFFICIENT DATA COVERAGE STRATEGY FOR TestMANAGER
QUOTE

"Testing is an infinite process of comparing the invisible to the ambiguous in order to avoid the unthinkable happening to the anonymous." - James Bach
ACKNOWLEDGMENTS

Firstly, I would like to thanks Axini for the opportunity of realizing this MSc thesis with them. Especially to my supervisors Machiel and Vincent who have guided me during the whole project making me feel as a part of Axini’s team.
Secondly, I really appreciate the hard time that Wan has spent trying to decipher some parts of my report due to my English style.
Finally, the unlimited support of my family, friends and my life partner Ada must be recognized since otherwise the success of this project would never have become a reality.
ABSTRACT

Model-based testing (MBT) emerged during the last years as a new approach to deal with the testing of complex software systems, generating sets of test cases automatically. MBT may be implemented following different approaches, the most common ones being: transition, data and state coverage. This MSc thesis focuses on the combination of transition and data coverage in order to achieve a more complete testing. This combination is approached by the use of two black box techniques: equivalence partitioning and boundary value analysis. GNU Prolog (Gprolog) software, a constraint solver, has been selected as a solution for both obtaining an immediate solution and achieving the domain of a constraint. Achieved results on a small but realistic case study show acceptable execution times in order to get 100% transition and boundary value coverage. Further results are a boundary value coverage/time graph, an application of boundary value analysis not only to numeric values but to most common data types, and a set of metrics in order to evaluate results. All these achievements were gathered in a boundary value strategy, named BVA, a data oriented strategy. As a conclusion, it is shown that merging MBT approaches results in more benefits that the single use of one of them.

Keywords: MBT, equivalence partitioning, boundary value analysis, Gprolog, BVA strategy
# Table of Contents

Quote .......................................................................................................................................................... 1  
Acknowledgments .................................................................................................................................. 2  
Abstract ................................................................................................................................................ 3  
Table of Contents ................................................................................................................................... 4  
Table of Figures ....................................................................................................................................... 6  
Index of Tables ......................................................................................................................................... 7  
Table of Definitions ............................................................................................................................... 8  

1. Introduction ............................................................................................................................................. 9  
   1.1 Objectives and motivation .................................................................................................................. 9  
   1.2 Justification ....................................................................................................................................... 10  
   1.3 Structure of the document ................................................................................................................. 11  

2. Background ............................................................................................................................................. 13  
   2.1 Previous concepts .............................................................................................................................. 13  
   2.2 Black box testing techniques ............................................................................................................ 13  
      2.2.1 Equivalence partitioning .......................................................................................................... 14  
      2.2.2 Boundary value analysis .......................................................................................................... 14  
   2.3 Ioco theory ....................................................................................................................................... 16  
   2.4 Constraint complexity ....................................................................................................................... 17  

3. Boundary value strategy ........................................................................................................................ 19  
   3.1 Application of Ioco theory ................................................................................................................. 19  
   3.2 Data driven coverage ......................................................................................................................... 20  
      3.2.1 Factors ...................................................................................................................................... 20  
      3.2.2 Strategy proposals .................................................................................................................... 21  
      3.2.3 Strategy selection, statistical test .............................................................................................. 23  
      3.2.4 Proposed/previous strategy comparison .................................................................................... 27  
   3.3 Value selection ................................................................................................................................... 28
TABLE OF FIGURES

Figure 1 Levene’s test ................................................................. 24
Figure 2 Normality test .............................................................. 25
Figure 3 One-way ANOVA .......................................................... 26
Figure 4 Tukey's test ................................................................. 26
Figure 5 Normality test .............................................................. 27
Figure 6 Student's T test ............................................................. 28
Figure 7 Value selection process ............................................... 29
Figure 8 BVA coverage graph .................................................... 35
Figure 9 Solver – time comparison ............................................. 38
Figure 10 Predictive model ......................................................... 39
Figure 11 Architecture ............................................................... 41
Figure 12 Constraint joining process .......................................... 43
**INDEX OF TABLES**

Table 1 Boundary values types ................................................................. 15
Table 2 Strategy comparison ...................................................................... 23
Table 3 GQM – Goals .............................................................................. 33
Table 4 GQM – Questions ......................................................................... 34
Table 5 GQM – Metrics ........................................................................... 34
# Table of Definitions

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Labeled Transition System</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>IOTS</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>After, Out(), Straces</td>
<td>16</td>
</tr>
<tr>
<td>2.4</td>
<td>IOCO</td>
<td>16</td>
</tr>
<tr>
<td>2.5</td>
<td>CSP</td>
<td>16</td>
</tr>
<tr>
<td>2.6</td>
<td>Equivalence relation</td>
<td>17</td>
</tr>
<tr>
<td>2.7</td>
<td>Relational language</td>
<td>17</td>
</tr>
<tr>
<td>2.8</td>
<td>Polymorphism</td>
<td>17</td>
</tr>
</tbody>
</table>
1. **INTRODUCTION**

This chapter will introduce both the objectives, motivation and justification which drove the realization of this MSc project. Straightaway, a section will explain the organization of this document, explaining each chapter and its content. As a result, a global vision of this document will be provided.

1.1 **OBJECTIVES AND MOTIVATION**

The software life cycle has changed over time. Software testing has evolved from being a last phase in software development – sometimes reduced, or simply skipped for time constraints - to being a cornerstone of it. Model-based testing (MBT) and test driven development (TDD) are two examples of such change. Accepting the importance of testing, the idea of testing as exhaustive and complete as possible, comes to mind. Nevertheless, complete testing of software is in general not possible since loops and infinite data sets can turn it into an infinite process [1] [2]. In order to solve that problem, several theories have been developed [1] [2]. loco theory [3], one of these theories, has been adopted as base of this research.

Axini, host and partner of this project, is a firm specialized in MBT. They perform MBT testing with regard to other organization’s products by the use of their own tool, called TestManager. TestManager allows Axini the automatic testing of software systems employing both the implementation of the system under test (SUT) and a specification of it. TestManager creates a test suite from the model derived from the specification. Thereupon, that test suite is automatically confronted with the provided implementation. As a result, it can be predicted which degree of conformation exist between the implementation and the specification [4], and therefore how successful the acceptance tests will be.

The present MSc thesis has as main objective the improvement of Axini’s data coverage utilizing one of the loco premises; specifically, the one claiming for a data characterization of large data sets. Black box testing techniques have been proved as an effective mechanism for such data characterizations [5]. Thus, two black box testing techniques boundary value analysis and equivalence partitioning have been the selected techniques to achieve that characterization. The preference of those techniques over the rest of techniques was made according to their proved usefulness [6] [7].
As a result, a new strategy had to be developed and provided to Axini. Performance is an issue for Axini, thus a metric system was demanded to be established in order to prove the behavior of the strategy. The Goal-Question-Metric framework (GQM) [8], a metric creation methodology, helped along the metric definition process. The idea of complete testing was mentioned at the beginning of this section. Providing a complete test set, as a proof of the strategy validity and correctness, was also part of the objectives proposed by this MSc thesis.

A remarkable challenge was to try and achieve similar performance figures as other strategies already developed by Axini. Since performance is an issue, the execution time must not be too high compared with current strategies adopted by Axini during its MBT process. This is a challenge since current strategies do not focus on data values and a considerable amount of processing is required to apply equivalence partitioning and boundary value analysis over all the transitions present in the SUT.

1.2 Justification

Axini, as a company specialized in MBT, is continuously looking for the improvement of its testing process. Different approaches are available for the implementation of MBT, such as data coverage, transition coverage or state coverage. Although those approaches are not incompatible, their integration had not been explored by Axini thus far. A combination of transition and state coverage of the model under test (MUT) is Axini’s approach nowadays. Data coverage was not yet an issue for them due to two reasons: their desire of performing a fast testing process and the idea that adding data coverage might enlarge this time. This MSc project looked for the addition of data coverage to Axini’s TestManager, which increases its coverage quality, since not only states and transitions are taken into account but interesting values are selected as well; this strengthens Axini’s competitiveness in an always difficult market.

To the best of our knowledge, no black box testing techniques such as boundary value analysis or equivalence partitioning have been fully automated within a MBT context using a constraint solver so far. Hence this MSc thesis will explore new paths for MBT which reports back benefits to the MBT community, and therefore provides a contribution not only to Axini but to MBT users in general.
1.3 STRUCTURE OF THE DOCUMENT

This subsection will explain the structure of the present document in order to have a global vision of it and, consequently, a better understanding. The contents of each chapter will be mentioned.

Chapter 1. Introduction

This brief chapter is in charge of contextualizing this MSc project. Objectives and motivation of the research are explained within this chapter too.

Chapter 2. Background

This chapter will expose the previous knowledge which had to be studied before the beginning of the project.

Firstly, a brief section will describe two basic concepts: labeled transition system and trace.

Secondly, Black box testing will be introduced. Straightaway, equivalence partitioning and boundary value analysis, black box techniques applied in this MSc thesis, will be described and explained with two examples.

Thirdly, the loco theory, cornerstone of this project, will be presented. The theory will be explained from its bases – LTS and quiescence - until its final establishment.

Finally, a formula for assessing constraint complexities will be introduced. This formula had a big impact on this research, since it was adopted both as core of prediction model creation and as a factor for the data oriented strategy.

Chapter 3. Boundary values strategy

This chapter will be in charge of presenting the main achievements accomplished during the realization of the present MSc thesis.

Firstly, it will be shown how loco theory was applied in the MBT context, in particular, to the boundary value analysis field.

Secondly, the data driven strategy will be presented. Its main factors will be discussed and it will be displayed how different combinations between them drive to the development of five strategy proposals. Upon that, a statistical test was performed in order to decide the best approach from the proposals. Another
statistical test was carried out to compare the best strategy from the proposals and the current data strategy of Axini.

Thirdly, the value selection section will show the process of applying equivalence partitioning and boundary value analysis by the data strategy for both numerical and non-numerical values.

Thereupon, metrics will be the next topic of this chapter. The GQM process for metric definition is introduced. Then its result, in the form of a metric, will be discussed.

Finally, a results section will be in charge of presenting both final results and problems found in the context of this MSc thesis. Whereas a discussion section will review the achievements of the project.

**Chapter 4. Related works**

Related works and researches which were taken into consideration before the start of the project will be presented in this chapter. In particular, the main ideas of six different studies will be mentioned.

**Chapter 5. Conclusions**

Conclusions, reflections and future works derived from this research will be mentioned and exposed during this final section.

Firstly, the main contributions that this MSc project was able to achieve, will be cited and emphasized.

Secondly, possible topics for future graduation projects will be proposed, considering several ideas which were coming out during the realization of the project. Ideas that unfortunately could not be addressed due to time limitations. Nevertheless, they will be described as a possible MSc thesis proposal.
2. **BACKGROUND**

This chapter will define all the concepts applied within this MSc thesis. The first subsection, Previous concepts, will introduce two basic concepts: labeled transition system and trace. Thereupon, the black box testing techniques section will describe the concepts themselves and the two black box techniques which were applied during this project: equivalence partitioning and boundary value analysis. Finally, the loco theory, cornerstone of this project, and constraint complexity will be presented.

### 2.1 PREVIOUS CONCEPTS

A labeled transition system consists of a collection of states and a collection of labeled transitions between those states, where the labels indicate what occurs during the transition. Labels belong to a global set \( L \). A special label \( \tau \notin L \) is used to denote an internal action. For an arbitrary \( L \subseteq L \), a stenography for \( L \cup \{ \tau \} \) is \( L \tau \).

**DEFINITION 2.1 LABELED TRANSITION SYSTEM** [9]. A labeled transition system (LTS) is a 5-tuple \( \{ Q, I, U, T, q_0 \} \) where \( Q \) is a non-empty countable set of states; \( I \subseteq L \) is the countable set of input labels; \( U \subseteq L \) is the countable set of output labels, which is disjoint from \( I \); \( T \subseteq Q \times (I \cup U \cup \{ \tau \}) \times Q \) is a set of triples, the transition relation; \( q_0 \in Q \) is the initial state.

A trace \( \sigma \) of an LTS is a finite sequence of observable actions. I.e. a possible combination of inputs and outputs, belong to the set \((I \cup U \cup \{ \tau \})\), linking states. The set of all traces over \( L (\subseteq L) \) is denoted by \( L^* \), ranged over by \( \sigma \), with \( \varepsilon \) denoting the empty sequence.

### 2.2 BLACK BOX TESTING TECHNIQUES

It is important to mention black box testing techniques since they were directly applied along this MSc project. Chapter 3 will show such application. According to the ISTQB - International Software Testing Qualifications Board -, black box can be define as: “Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure” [5]. Technically speaking, a tester will exclusively focus on the output of the behavior being tested, and not on the internal structure.
The use of black box testing techniques has a number of advantages: 1. There is no need for the tester to have precise functional knowledge of the system; 2. Test cases can be defined as soon as the functional specification is complete; 3. It provides an end user point of view on tests; 4. It helps to identify functional contradictions.

There are several black box techniques such as domain tests, decision tables, equivalence partitioning and boundary value analysis. The current MSc thesis will focus on the last two: equivalence partitioning and boundary value analysis. As it will be shown later, a combination of both techniques will be the chosen approach. The next sub-sections will briefly explain both black box techniques.

2.2.1 Equivalence Partitioning

The first technique being explained is equivalence partitioning. According to the ISTQB, “The idea behind this technique is to divide a set of test conditions into groups or sets that can be considered the same, hence only one condition from each partition will be tested. This is because we are assuming that all the conditions in one partition will be treated in the same way by the software” [5]. Finding primary functional defects where data is wrongly handled is the main goal of this testing technique. Indeed, handling in the same way the values of each equivalence partition is its main strength, since the number of test cases decreases. However, if the previous statement is not followed up and no single value is tested from some partition, the loss of important test values will occur.

We give a small example of how this technique is applied:

\[ (X > 0 \&\& X < 4) \lor (X > 5 \&\& X < 10) \]

In this case \( X \) can take values within the domain \([1, 2, 3, 6, 7, 8, 9]\). Applying equivalence partitioning, two sets would be obtained, one gathering \([1, 2, 3]\) and another with \([6, 7, 8, 9]\). Equivalence partitioning claims that by testing one element from each group, the whole domain is sufficiently tested.

2.2.2 Boundary Value Analysis

Boundary value is defined, by the ISTQB, as “testing at the boundaries between partitions, taking the minimum and maximum (boundary) values from each partition” [5]. A testing process over the boundary values makes sense because in practice the majority of data errors corresponds to boundary values [6] [7]. Moreover, researches on retrospective fault data demonstrated that boundary value analysis outperforms other testing techniques such as random testing, statement coverage or branch coverage [10] [11].
There are different boundary value methodologies, according to the exhaustiveness of their value testing, i.e. the number of values required to be checked. In ascending order regarding that number, they are: robustness testing, worst-case testing and robustness worst-case testing.

Table 1 shows a comparison of these different methods, according to the requested values to be tested, and therefore, the number of test cases needed. (Clarification of Table 1: ‘min’ stands for minimum, ‘max’ for maximum, ‘nom’ for normal, ‘-‘ for the immediate previous value, and ‘+’ for the immediate next value).

Such a number of test cases comes from the following reasoning: each test case must select a single variable boundary value, keeping the rest of the variables within a normal value – understanding normal as a non-boundary value. Hence the number of test cases in each methodology depends on the number of variables present and the number of values taken into account. E.g. a robustness type may have an extent of $6n + 1$ test cases, since a test case would involve the $min$- value of one variable and normal values for the rest of variables, then another test case with the $min$ value of the same variable and normal values for the rest, and so forth for the rest {$min+$, $max$-, $max$+, $max$}, having an extra test case where all variables take a normal value. As a result, six test cases would be necessary for each variable ($6 \times n$), plus that one involving normal values. Nevertheless, some of these methodologies are barely used in software testing, because they require a large number of test cases. This MSc project focused on its basic approach, mainly because min- and max+ are non-valid values for the constraint solver adopted – the solver does not return back values out of the domain and min- and max+ are both out of the solution’s domain [18] (see section 3.3.1).

<table>
<thead>
<tr>
<th>Method</th>
<th>Values being tested</th>
<th>Test cases per variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Value</td>
<td>{$min$, $min+$, $nom$, $max$-, $max$}</td>
<td>$4n + 1$</td>
</tr>
<tr>
<td>Robustness</td>
<td>{$min$-, $min+$, $nom$, $max$-, $max$+, $max$}</td>
<td>$6n + 1$</td>
</tr>
<tr>
<td>Worst Case</td>
<td>Cartesian product of {$min$, $min+$, $nom$, $max$-, $max$} for each variable</td>
<td>$5^n$</td>
</tr>
<tr>
<td>Robustness Worst Case</td>
<td>Cartesian product of {$min$-, $min$, $min+$, $nom$, $max$-, $max$, $max$+} for each variable</td>
<td>$7^n$</td>
</tr>
</tbody>
</table>

*Table 1 Boundary values types*
A simple case of that basic approach can be seen continuing with the example presented in the previous section. As was shown before, after applying equivalence partitioning, two sets of possible values are present: \([1, 2, 3]\) and \([6, 7, 8, 9]\). Boundary value analysis requires to test the values 1, 3, 6 and 9, the minimum and maximum of each partition.

### 2.3 LOCO THEORY

This chapter explains one of the main references used during this MSc thesis, the well-known loco theory [3]. loco stands for input/output conformance, understanding conformance as the judgment whether an implementation is correct with respect to a specification. loco is a theory fundamentally based on the notion of quiescence in the context of LTSs.

Quiescence is a concept which designates system states that will not produce any output response, for as long as the system remains in those states. In an input/output system, inputs are continuously allowed, thus no input action will be rejected. As a consequence no deadlock will be present, since the possibility of new input actions will be always available. An LTS is denoted as strongly responsive if it always eventually enters a quiescent state; in other words, if it does not have any infinite number of labeled outputs (\(U\)). Traces that may contain the quiescence action \(\delta\), are called suspension traces (\(Straces\)).

At this point, it is possible to join the concepts of LTS and input/output system in one single term, input/output transition system (IOTS).

**Definition 2.2 IOTS** [9]. An input-output transition system \(p = \{Q, I, U, T, q_0\}\) is an LTS where the label set \(L\) is partitioned into an input label set \(I\) and an output label set \(U\) and for which all inputs present on the system \(I\) are enabled in all states: \(\forall q \in Q, a \in I: q \xrightarrow{a} p\).

Given a specification \(s\) and an (assumed) model of the SUT \(i\), the relation \(i \text{ loco } s\) denotes that \(i\) conforms to \(s\) according to loco theory. The criterion that decides whether it holds is the set of \(Straces\) of \(s\): it must be the case that, after any such trace \(\sigma\), every output action that \(i\) is capable of, should be allowed according to \(s\). This is formalized by defining \(p\) after \(\sigma\) - the set of states that can be reached in \(p\) after the suspension trace \(\sigma\) - , \(\text{out}(p)\) - the set of outputs and \(\delta\)-actions (\(p \xrightarrow{\delta} p\)) of \(p\) - and \(Straces\) of \(p\) - the suspension traces of \(p\). Definition 2.3 expresses formally the ideas presented.
\textbf{DEFINITION 2.3 After, Out(), Straces} [9]. Let $p \in \text{LTS} (I, U)$, let $P \subseteq Q_p$ be a set of states in $p$, let $i \in \text{IOTS} (I, U)$, $s \in \text{LTS} (I, U)$ and let $\sigma \in \mathcal{L}$. 

- $p \text{ after } \sigma = \text{def} \{ p' \mid p \xrightarrow{\sigma} p' \}$
- $\text{out} (p) = \text{def} \{ x \in U \mid p \xrightarrow{\delta} \delta \} \cup \{ \delta \mid p \xrightarrow{\delta} \}$
- $\text{out} (P) = \text{def} \cup \{ \text{out} (p) \mid p \in P \}$
- $\text{Straces} (p) = \text{def} \{ \sigma \in \mathcal{L}_\delta \mid p \xrightarrow{\sigma} \}$

The following defines the implementation relation $\text{Ioco}$, modulo a function $\mathcal{F}$ that generates a set of test-traces from a specification. This fact had a big impact in the consecution of this MSc thesis, since $\mathcal{F}$ determines the coverage that the strategy will achieve. In the next definition, $2^X$ denotes the powerset of $X$, for an arbitrary set $X$.

\textbf{DEFINITION 2.4 IOCO} [9]. Given a function $\mathcal{F}: \text{LTS} (I, U) \rightarrow 2^{L_i}$, we define $\text{Ioco}_{\mathcal{F}} \subseteq \text{IOTS} (I, U) \times \text{LTS} (I, U)$ as follows:

$$i \text{ loco}_{\mathcal{F}} s \iff \forall \sigma \in \mathcal{F}(s): \text{out} (i \text{ after } \sigma) \subseteq \text{out} (s \text{ after } \sigma)$$

So $i \text{ loco}_{\text{Straces}} s$ means $\forall \sigma \in \text{Straces} (s): \text{out} (i \text{ after } \sigma) \subseteq \text{out} (s \text{ after } \sigma)$. loco was used as an abbreviation for $\text{Ioco}_{\text{Straces}}$. The definition specifies that for every trace in the specification, the implementation does not allow an output which is not in the specification after that trace.

\section*{2.4 CONSTRAINT COMPLEXITY}

This sub-section will introduce the concept of constrain complexity, a concept which will be used as a factor within the transition selection process. Constraint satisfaction problem (CSP), equivalence relation and relational language are some of the terms that will be defined throughout this chapter. Straightaway, a theorem will present the formula which will be used for assessing constraint complexities.

CSPs may be defined as mathematical problems where a set of objects must satisfy a number of constraints or limitations. CSPs are represented as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods. Formally, CSP is defined as in definition 2.5.

\textbf{DEFINITION 2.5 CSP} [12]. For any set $A$, and any constraint language $\Gamma$ over $A$, the constraint satisfaction problem CSP ($\Gamma$) is the combinatorial decision problem with:

\textbf{Instance}: A triple $(V, A, C)$, where:
• $V$ is a set of variables;
• $C$ is a set of constraints, $\{C_1, \ldots, C_q\}$.

Each constraint $C_i \in C$ is a pair $\{s_i, \rho_i\}$, where:

- $s_i$ is a tuple of variables of length $m_i$, called the constraint scope;
- $\rho_i \in \Gamma$ is an $m_i$-ary relation over $A$, called the constraint relation.

**Definition 2.6 Equivalence Relation** [13]. Equivalence relation can be defined as $\rho$ on the set $\{1, \ldots, k\}$ that contains those pairs $\{i, j\}$ where $s_i = s_j$. I.e. the equivalence relation of a constraint can be considered as the different number of tuples of variables present in the constraint. E.g. having a constraint defined as “$X > Y \&\& X < 7$”, two tuples may be defined: $\{X,Y\}$ and $\{X\}$. Therefore the equivalence relation for such constraint is two.

**Definition 2.7 Relational Language** [13]. A relational language $\tau$ is a set of relation symbols $R_i$, each associated with a finite arity $k_i$. A (relational) structure $\Gamma$ over the (relational) language $\tau$ is a countable set $D_\Gamma$ (the domain) together with a relation $R_i \subseteq D_\Gamma^{k_i}$ for each relation symbol of arity $k_i$ from $\tau$.

**Definition 2.8 Polymorphism** [13]. Let $D$ be a countable set, and $O$ be the set of finitary operations on $D$, i.e., functions from $D^k$ to $D$ for finite $k$. We say that a $k$-ary operation $f \in O$ preserves an $m$-ary relation $R \subseteq D^m$ if whenever $R (x_1^1, \ldots, x_m^i)$ holds in $\Gamma$ for all $1 \leq i \leq k$, then $R (f (x_1^1, \ldots, x_k^1), \ldots, f (x_m^1, \ldots, x_m^k))$ holds in $\Gamma$. If $f$ preserves all relations of a relational $\tau$-structure $\Gamma$, we say that $f$ is a polymorphism of $\Gamma$. I.e. a function $f$ may be considered as a polymorphism of $\Gamma$, if its application to the domain of that relational $\tau$-structure $\Gamma$ does not modify the relations of the domain.

**Theorem 1** [13]. Let $\Gamma$ be closed under a binary injective polymorphism, and let $S$ be an instance of CSP ($\Gamma$) with $n$ variables and $q$ constraints. Let $k$ be the maximal arity of the constraints, and let $m$ be the maximal number of equivalence relations in the representations for the constraints. Then there is an algorithm that decides the satisfiability of $S$ in time $O (qm(qmk^2 + n))$.

Theorem 1 provides an algorithm to determine the satisfiability of a CSP instance within a reasonable time. This was used in the context of this MSc thesis for two different purposes. Firstly, as a prediction measurement in order to establish prediction models before test execution – an issue discussed in section 3.5--; secondly, as a complexity measurement of constraints – regarding strategy factors in section 3.2.1.
3. **BOUNDARY VALUE STRATEGY**

This section will present the main contribution of this MSc project. Firstly it will be illustrated how loco theory was applied within this project context. Then the development of the BVA strategy, a data oriented strategy, will take place. Afterwards, the value selection process will be explained. In order to assess the BVA strategy, a set of metrics was developed, following the GQM methodology. Finally two sections showing the achieved results and a discussion of them will close this chapter.

### 3.1 APPLICATION OF IOCO THEORY

The implementation of all the traces of a specification is in general not possible since there may be an infinite number of Straces. That impossibility is due to the presence of loops or infinite datasets. This MSc thesis applies a characterization of the dataset in order to make the number of Straces finite. Four main steps are required in order to apply the loco theory:

1. Characterization of the infinite set of Straces.
2. Computation of that finite subset.
3. Generation of test cases for that finite subset.
4. Report coverage.

The first phase characterizes the infinite Straces in order to obtain a manageable and representative finite subset of them. Otherwise the computation of test cases would be infinite and therefore no complete testing could be performed. Aforesaid characterization of the dataset, was the approach chosen for reducing those infinite Straces. Two decisions were required to be made at that point: depth of paths and data selection criteria – basically, which values were taken into consideration. That decision was not immovable, since the tradeoff of those two values reported different subsets and, as a consequence, different coverage. In fact, it is a decision taken by the user according to the model being tested and its testing requirements. The two black-box testing techniques - boundary value analysis and equivalence partitioning – were going to be applied for such data selection.

The second phase computes the finite subset of Straces achieved in the previous phase. Once the computation is finished, the minimum number of test cases needed in order to cover those Straces turns up. At that moment, the tester may agree with such number of test cases and therefore continue with the testing; or he
may disagree, and come back to the previous phase modifying the selected depth and/or the data selection criteria. Moreover, at that instant, the set of symbolic paths which the strategy will attempt to cover, is known by the strategy.

The third phase, as its description indicates, generates the test cases defined in the second phase. This generation is accomplished according to several factors, e.g. constraint complexity, number of child-states or data coverage. Section 3.2, will explain in detail the approach chosen, but it can be mentioned here that several approaches were proposed and thereupon performance tests decided the most appropriate one.

The final step, report coverage, is the phase in charge of checking the results achieved by the strategy in terms of the metrics defined in section 3.4. Performance, effectiveness, and therefore efficiency were the main aspects this MSc project was looking to achieve.

### 3.2 DATA DRIVEN COVERAGE

No literature establishes a best approach to follow dealing with data coverage in MBT. Hence, previous knowledge of the model being tested seemed the best component to be considered. Performing a pre-analysis of the model, three main factors were taken into account, deciding which transitions were more convenient to choose in the first place. Constraint complexity, number of child-states, and coverage exhaustiveness were those three factors. This section introduces those factors and their application within the proposed strategy. A statistical test, so-called one-way ANOVA, will establish a ranking between the proposals. This statistical process will be displayed. Finishing this section, a subsection will show how the Student’s T test was applied in order to compare the best strategy from the aforesaid ranking to the DataSimple strategy, the current data strategy of Axini.

#### 3.2.1 FACTORS

It is important to differentiate a couple of terms used during the realization of this project. Constraints are composed of one or more sub-constraints separated by an OR operator (|| / ∨). Whereas a sub-constraint may be defined as a set of constraints gathered by an AND operator (&& / ∧). E.g. the constraint “(X > 2 && X < 5) || (X == 1)” is composed of the sub-constraints “X > 2 && X < 5” and “X == 1”, whereas “X > 2” and “X < 5” are the two sub-constraints of “X > 2 && X < 5”.

Section 2.4 presented a formula to obtain the complexity of a constraint. This MSc thesis aims to establish a relation between this complexity and the number of
interesting values from a testing perspective. The formula implies that the complexity of a constraint increases with its number of sub-constraints. Indeed the data domain of the constraint is modified with each constraint; thus the number of boundary values increases also. Applying the previous reasoning, constraints with a higher complexity have a larger number of boundary values, thus we first target constraints with a high complexity.

The number of child-states a state links to, was the second factor taken into account. A state cannot be reached unless its parent-state is reached, thus the sooner such a parent-state is covered, the more data will be available to be covered. That statement corresponds to the straightforward relation of data covered over time. Hence, the strategy attempts to cover as much data as possible within as little time as possible.

Coverage exhaustiveness is the third factor taken into account in the strategy. That factor stands for the number of test cases a transition is required to be involved in to be considered as fully covered – covered in the sense of boundary value analysis. The exhaustiveness was a factor selected by the user in section 3.1 according to its preferences. The integration of such functionality with the strategy was accomplished by providing a configuration item which allowed two possibilities: min and max. Min covers the model with the minimal number of test cases demanded for covering all the boundary values of the SUT, whereas max seeks to accomplish more test cases than necessary for boundary value analysis in order to have a larger data coverage. Hence within max, boundary values are tested first and other values are tested afterwards - if any.

As was mentioned at the beginning of the current chapter, no best approach has been specified of how to proceed a boundary value analysis in MBT, since different models may request different approaches. Therefore, this thesis carried out a performance comparison of different approaches, focusing on different aspects. Visiting uncovered states first or trying to fully cover model branches first, are examples of those aspects.

### 3.2.2 Strategy proposals

A total number of five approaches were studied. The first strategy’s goal is to discover as much data as possible in little time. For this purpose, strategy 1 bases its behavior on sorting possible transitions, at the current state of the test case, by number of child-states and complexity. Straightaway, from this set of transitions, the first transition which has not reached its coverage exhaustiveness yet is the one selected. Whether all set has already reached it, the transition which has been
involved in fewer test cases, is the one picked out.

Reducing the tree width is the main goal of strategy 2, which basically follows the same reasoning as strategy 1 - in terms of transition sorting. It attempts to close model branches as soon as possible. As a consequence, the model tree is reduced, and therefore the set of possible transitions is smaller. Dealing with this smaller set, the strategy has fewer transitions to process, hence the transition selection process takes less time. Thus, an increase in the performance is expected.

Strategy 3, similar to strategy 1, has as main goal the coverage of uncovered states first. As a result, the model is covered incrementally, following the next reasoning. The \textit{min} valuation method is applied to all transitions; upon that, the \textit{max} valuation method is applied to all transitions, and so forth.

Strategy 4 focuses its effort on applying the opposite of strategy 2, i.e. closing less complex branches first. Consequently, it reports a faster tree reduction than strategy 2.

Strategy 5 encourages a selective test case generation. A preliminary search of key transitions is performed in the first place – considering as key transitions the most complex transitions within the model, i.e. the transitions which require to involve in a large number of test cases. Thereon, the coverage of those key transitions is the target of the strategy. Broadmindedness or scope selection may be considered as the main strengths of this fifth strategy.

In order to improve the performance of the strategy, a pruning method was provided to all the strategies. That method is in charge of pruning the model tree in case no more test cases are necessary on a specific branch - i.e. the coverage exhaustiveness of all transitions of the branch has been reached.

Table 2 shows a comparison – focused on performance - between the five strategies. 22 states with 29 transitions are considered as a medium size model (see Annex 1). The time shown on Table 2, stands for the average time achieved by each strategy along 100 executions. Although the difference among these times may not be considered as significant, it is important to remember that that time responds to a model with 29 transitions. Hence even a small difference in that model will suppose a big difference in a model with thousands of transitions. Furthermore, the main advantages and disadvantages and the minimum number of test steps required to achieve 100% boundary value coverage are mentioned.
As was mentioned before, there is a lack of literature available in order to decide which of the proposed strategies is the most suitable for Axini’s purposes. Table 2 shows a small performance difference among the strategies. It slightly suggests that strategy 4, closing easiest branches first, has the best performance. Nevertheless it was impossible to determinate which strategy should have been used at this point. Thus, statistical tests comparing the performance of each strategy were carried out. Section 3.2.3 will discuss this statistical procedure.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Main advantage</th>
<th>Main disadvantage</th>
<th>Steps</th>
<th>Time for medium model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fast boundary value coverage</td>
<td>Larger transition selection time</td>
<td>115</td>
<td>6.0695</td>
</tr>
<tr>
<td>2</td>
<td>Model width reduction</td>
<td>Less boundary value oriented</td>
<td>112</td>
<td>5.8236</td>
</tr>
<tr>
<td>3</td>
<td>Fast transition coverage</td>
<td>Less boundary value oriented</td>
<td>115</td>
<td>5.8919</td>
</tr>
<tr>
<td>4</td>
<td>Model width reduction</td>
<td>Less boundary value oriented</td>
<td>113</td>
<td>5.7798</td>
</tr>
<tr>
<td>5</td>
<td>More complex constraints covered first</td>
<td>Larger transition selection time</td>
<td>116</td>
<td>6.7988</td>
</tr>
</tbody>
</table>

Table 2 Strategy comparison

3.2.3 STRATEGY SELECTION, STATISTICAL TEST

Although a small difference in the performance was found in the previous section, a statistical test was required to assure that. “Which proposed strategy performs the best?” was the research question proposed. The first employed test is so-called one-way ANOVA. It stands for one-way analysis of variance and it determines whether there is difference between the means of three or more samples - the test can be performed for all five proposed strategies, hence the first requirement of comparing more than two means was fulfilled.

Analysis of variance is a technique for analyzing the way in which the mean of a variable is affected by different types and combinations of factors. It is an extension of the independent samples t-test and can be used to compare any number of groups or treatments [15]. Three assumptions must have been satisfied in order to

---

1 SPSS [14] was the statistical tool used for the realization of all statistical tests.
consider reliable the results of one-way ANOVA: 1. Variances of populations are equal; 2. Response variables are normally distributed; 3. Responses for a given group are independent and identically distributed normal random variables.

A zero step, composed by hypothesis definition and $P$ value selection, starts the test. On one hand, the $P$ value is known in the statistical field as “the chance of incorrectly rejecting the null hypothesis” [15]. In the medical area a level of confidence of 99% - 0.01 $P$ value - is the required option, due to the risk of this domain. In other fields, a level of confidence of 95% - 0.05 $P$ value - is the common and accepted figure. Hence that last level of confidence, 95%, was the one selected for this statistical test. On the other hand, the hypothesis of the experiment was defined as:

- $H_0$: there is no statistical difference between strategies’ executions.
- $H_1$: there is a statistical difference between strategies’ executions.

Before starting with the statistical test, those three aforesaid assumptions should have been verified, thus that was the first step.

Two hypotheses were proposed in order to verify the first assumption, equality of variances:

- $H_0$: there is no difference between samples’ variances.
- $H_1$: there is a difference between samples’ variances.

Levene’s test, a test designed for checking equality of variances among samples, was the selected test. It performs an analysis of variances on the absolute deviations of values from the respective group means for each dependent variable. Figure 1 shows the results of this test. 0.498 was the achieved $P$ value. As 0.498 $\geq$ 0.05 ($P$ value of the test), the null hypothesis could not be rejected, hence equality of variances between samples could be assumed.

<table>
<thead>
<tr>
<th>Test of Homogeneity of Variances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Levene Statistic</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>0.855</td>
</tr>
</tbody>
</table>

*Figure 1 Levene's test*

The second assumption claimed that response variables must follow a normal distribution. In that case, the Shapiro-Wilk test was in charge of deciding whether the respond variable *Time* follows a normal distribution. This test performs a comparison among the actual data and how this same data should be distributed if
it would follow a normal distribution. The next formula represents how to achieve that result.

\[ W = \frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \]

where:

- \( x_i \) are the ordered sample values.
- \( a_i \) are constants generated from the co-variances, variances and means of the sample (size n) from a normally distributed sample.
- \( \bar{x} \) is the sample mean.

Pursuing a process similar to Levene’s test, two hypotheses were proposed:

- \( H_0 \): the sample follows a normal distribution.
- \( H_1 \): the sample does not follow a normal distribution.

<table>
<thead>
<tr>
<th>Tests of Normality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy</td>
</tr>
<tr>
<td>Time 1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

\( * \). This is a lower bound of the true significance.

\( a \). Lilliefors Significance Correction

Kolmogorov-Smirnov and Shapiro-Wilk were the two normality tests provided by SPSS. Best practices on statistics recommend the usage of Shapiro-Wilk for sub-populations smaller than thirty data points, hence, as was mentioned before, Shapiro-Wilk was used as normality test. Since 0.606, 0.158, 0.062, 0.648 and 0.931 are higher than 0.05, the null hypothesis could not be rejected in any of the cases and the normality of all samples could be assumed.

Finally, the third assumption could be assumed since no extra factor was considered during the experiment and therefore values were randomly and independently generated.
Results of the one-way ANOVA (see Figure 3) showed a $P$ value of 0.00. As $0.00 < 0.05$ the null hypothesis could be rejected and therefore a statistical difference can be assumed between the strategies.

### ANOVA

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between Groups</td>
<td>7,078</td>
<td>4</td>
<td>1,769</td>
<td>804,169</td>
<td>0.00</td>
</tr>
<tr>
<td>Within Groups</td>
<td>0.099</td>
<td>45</td>
<td>0.002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>7,177</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 3 One-way ANOVA*

At that moment, a difference across strategies was proved. However comparison between each strategy was not done yet, hence no possible performance ranking could be established. Thus, the next step was to carry out a post-hoc process of comparison among strategies’ means. Tukey’s test is the appropriate test for this purpose. “Tukey’s test determines the individual means which are significantly different from a set of means. Tukey’s test is a multiple comparison test and is applicable when there are more than two means being compared. Typically, Tukey’s test is utilized after an (Analysis of Variance) has shown that significant difference exists and determines where the difference exists.” [16].

### Tukey HSD

<table>
<thead>
<tr>
<th>Strategy</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5,779800</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>5,823600</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5,918000</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>6,069500</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>6,269500</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.243</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.
a. Uses Harmonic Mean Sample Size = 10,000.

*Figure 4 Tukey’s test*

Four main groups were defined by Tukey’s test (see Figure 4). A first group, composed by strategies 2 and 4, highlighted as the group with best strategies, then strategy 3, 1 and 5 took place respectively. A deeper analysis of these results will be done and presented in section 3.5.
3.2.4  Proposed/Previous Strategy Comparison

A comparison between the current data strategy of Axini and the proposed strategy took place and is presented in this section. Both strategy 2 and 4 were the candidates to face Axini’s DataSimple strategy, since there was no statistical difference between them - as was proved on the previous chapter. In this case strategy 2 was the final choice. The time both strategies – strategy 2 and DataSimple strategy – required to obtain a 100% transition coverage was the time used for realizing the test.

A similar process was adopted as in the case of the strategy selection. However, having two samples instead of five, one-way ANOVA could not be used. The Student’s T test is a suitable test. According to the literature, “The Student’s t-test determines whether two populations express a significant difference between population means. A significant difference is distinguished from a nonsignificant difference by the properties of the normal distributions characterized by the data.” [17]. As a one-way ANOVA test, the Student’s T test must satisfy a set of assumptions in order to achieve reliable results: normality of both sub-populations and independence of samples.

Does the new suggested strategy behave better than the DataSimple strategy? That was the proposed research question. With a 95% level of confidence, the first requirement to be fulfilled was to verify the normal distribution of the samples’ data. Both sub-populations with $P$ value 0.627 and 0.158 of respectively (see Figure 5), were greater than 0.05. Hence both null hypotheses ($H_0$: sample follows a normal distribution) were not rejected and normality of both samples was assumed.

The second assumption could be assumed indeed, since both samples followed independent processes.

<table>
<thead>
<tr>
<th>Tests of Normality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

* This is a lower bound of the true significance.
  a. Lilliefors Significance Correction

Figure 5 Normality test

Only after both assumptions were fulfilled, the Student’s T test took place. Figure 6 shows the results obtained. Previous to the test itself, it was important to analyze the results of the Levene’s test. As was explained during section 3.2.3, it was a test
in charge of accomplishing an equality of variance test among sub-populations. 0.054 was the retrieved \( P \) value. As \( 0.054 \geq 0.05 \), the null hypothesis (\( H_0 \): both sub-population have equal variances), could not be rejected and the equality of variances across samples was assumed.

<table>
<thead>
<tr>
<th>Independent Samples Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Levera's Test for Equality of Variances</strong></td>
</tr>
<tr>
<td>( F )</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Figure 6 Student's T test**

0.00 was the gleaned \( P \) value. As \( 0.00 < 0.05 \), the null hypothesis was rejected and the alternative one (\( H_1 \): there are differences between sub-populations), was assumed. As a consequence, statistical evidence was proved. The final step of the test was to decide which strategy performed better. Q-Q plot and a descriptive indicator like the mean were the factors taken into account in order to answer the research question. Strategy 2, with a mean of 5.8236 seconds, could be claimed to have a better performance than the DataSimple strategy with a mean of 6.3713 seconds.

### 3.3 Value selection

The transition selection process was defined along the lines of the previous chapter, but an explanation about value selection was not yet given. Constraint solving, equivalence partitioning and boundary value analysis have been the essence of that process. The followed process will be described and analyzed within this section.

Furthermore, the application of boundary value analysis to non-numerical objects as strings and enumerables will be explained. GNU prolog [18] was used as constraint solver along this project.

#### 3.3.1 Process

The Valuation method identifies what sort of value the strategy attempts to achieve. \( \text{Min} \), \( \text{max} \) and \( \text{random} \) are the main three valuation methods of Axini within its MBT context. If the strategy seeks for the minimum value that satisfies a constraint, :\( \text{min} \) is the adopted valuation method. Analogously, :\( \text{max} \) returns back the maximum value. If a random value is requested, :\( \text{random} \) is the chosen valuation method. As a new contribution of this MSc thesis, the :\( \text{remaining} \) valuation method was created. It is used when a boundary value is requested. E.g.
having a constraint defined as the one presented on section 2.2.1 - \((X > 0 \&\& X < 4) \| (X > 5 \&\& X < 10)\) – :\text{min} would return ‘1’, :\text{max} ‘9’, :\text{remaining} ‘3’ and ‘6’, whereas :\text{random} returns the rest of values: ‘2’, ‘6’, ‘7’ and ‘8’. It was specified that random must return a value which has not been used yet.

**Figure 7 Value selection process**

Figure 7 shows a chart of the process. A path for the :\text{min} and :\text{max} valuation methods, one for :\text{random}, and one for :\text{remaining} are the three main flows proposed. The shortest path is the one composed of :\text{min} and :\text{max}. If there is a demand of minimum or maximum value of the query, a solution is immediately provided by the solver. In such a case, there is no necessity to check the solution since the solver cannot provide impossible solutions.

On the other hand, :\text{remaining} is the longest path; the reason is that no call for a solution is made to the solver. Instead, a solver call for the domain of each variable present in the constraint is made. Then, it is time to apply equivalence partitioning and boundary value analysis. The domains provided by the solver are already in equivalence partition form; hence no extra treatment is needed. The boundary values of each variable are obtained by taking the first and last value of each partition.

Once the values which each variable may take are clear, it is time to form the solution. A set of solutions is built up selecting boundary values – or random if
there was no boundary value - and the one with the higher number of boundary values is the chosen and provided solution.

An additional step was found to be necessary at this point. Certain value combinations are not allowed. Although the domains of the different variables of a constraint are correct, a problem defined as spatial constraints shows up. That term stands for the impossibility of some value combinations according to the combination of all the sub-constraints present in the constraint, i.e. the domains of the variables are dependent between them. Therefore the selection of one of the values modifies the other variables domains. E.g. a constraint defined as \((X > 0 \&\& X < 4 \&\& Y > 2) \&\& !(X == 2 \&\& Y == 3)\) has a dependent domain. Considering, for instance, ‘2’ the value for \(X\), then \(Y\) cannot be ‘3’, whereas for other values of \(X\) it would be a valid solution.

Since that case of running into impossible combination of values was not found commonly, the solution adopted was to start the value selection process again. Such impossible combinations are stored in order to avoid them when selecting the most appropriated solution. Section 3.6 will further discuss about this issue.

The third approach employs the random valuation method. It is similar to the remaining one, but excludes the equivalence partitioning and boundary value analysis steps (Figure 7). I.e. when a query is sent to the solver in order to get the variable domains, a solution is formed, and then a final step of checking the formed solution is performed immediately. If the solution is valid, it is sent back to the strategy; if not, the impossible combination of values is stored (not visible in Figure 7 for the sake of readability) and the process is restarted.

Thus far, all constraints have been composed of variables being restricted by values. Nevertheless, state variables might restrict variables in the same way. E.g. having two states variables \(foo\) and \(var\) updated at some point in the model, a constraint may be defined as \((X > foo \&\& X < var)\). As a consequence, a new type of constraint came out, named “dynamic constraint” - along this MSc project -, since its solution depends on the state variable’s values at each point of the testing procedure. However, no extra work is required in order to deal with those constraints, hence they are solved as was previously explained (see Figure 7).

3.3.2 Non-numerical elements

Boundary value analysis and equivalence partitioning are typically applied to numerical elements. These techniques can also be used with regard to for example the non-numerical elements Boolean, String or Enumerable. Length or size of such an element may be considered as a value for applying equivalence partitioning and
boundary value analysis. Nevertheless it did not seem the best approach in some cases, since no equivalence partitions could be defined, thus a loss of values may occur.

Booleans were not taken into account since the solver can deal with them by making a match ‘0’ standing for false and ‘1’ for true, and no more boundary value is possible [18].

Strings were the first elements covered. Internal boundary conditions or sub-boundary were concepts used as reference for [19]. It specified a set of boundary values based on ASCII, i.e. chars ‘A’ and ‘Z’ represent the boundary values; then ‘A’, ‘B’ (min+1), ‘M’ (normal elements), ‘Y’ (max-1) and ‘Z’ represent the elements which should be tested in Strings, allowing only upper case letters. However, this idea does not seem the most appropriated approach since it highly differs from Axini’s normal use of strings. On the other hand, Gprolog [18] has the ability to handle strings as numbers (in the same way as Booleans), but there is no plausible way to check all possible values a string may adopt.

Axini’s normal use of string consists of assignments and comparisons of the form: ‘string = “foo”’ or ‘string == “foo”’. Hence each of them is considered as a single boundary value. Thus no internal boundary values or solver solution have been contemplated and taking directly those values from the constraint is the elected approach. E.g. in case of the constraint ‘string == “foo” || string == “bar”’ the boundary values attempted to be covered would be “foo” and “bar”. Although one might expect a possible performance increase due to the absence of solver usage, the performance remained the same. (see section 3.5)

The second kind of element researched is enumerable: Array and Hash. As mentioned before, size might not be the best path to follow, thus another approach was tried. Looking at the contents of these elements, both integer and string may be their values. Hence it was decided in some case to follow the same procedure, explained in section 3.3.1, using the solver, and in other cases the solution without solver was adopted. If the values are integers, the pursued value selection process was already presented in section 3.3.1. E.g. ‘s[0] > 0 && s[0] < 5’ or ‘foo[“length”] > 0 && foo[“length”] < 5’ are both handled as normal integers. Contrary, if the values are strings, two possible processes may be adopted. When the constraint has only string variables, the operation is accomplished as mentioned before, not applying the solver. On the other hand, if the constraint has numerical or Booleans variables together with the string variable, the solver was applied. Mainly because dealing
with all constraint at once is faster than solving first the strings and then the rest of the variables.

3.3.3 Problems found

In order to get the most accurate solutions – in terms of domains -, the value selection process was individually applied to each sub-constraint of the constraint. Handling domains easily and having multiple solutions per constraint are the advantages reported by this method. Nevertheless, it has a drawback: when a sub-constraint has not all variables present in the constraint, no domain for that missing variable is returned. Thus, no value can be assigned to that variable. This is a problem since all variables are part of the solution, and not having one turns that possible solution into an uncomplete solution, and therefore an error occurs. Assigning interesting values to those missing variables was the elected solution. Hence, not only the problem is overcome, but a better solution is composed finally. Since boundary values from that variable are selected in first place, and only when no boundary value is remained to be tested, a random value is taken.

The solver getting back wrong domains due to the presence of multiple OR operators ‘||’ in a constraint, was another problem this MSc thesis had to deal with. This problem appears when not all variables are present at all sides of an OR-constraint, considering that an OR-constraint has n+1 sides, ‘n’ being the number of ‘||’ symbols. Although the boundary values are perfectly recognized, the domains of that missing variables are retrieved as the biggest domain possible, i.e. all values between 0 and 268435455, the minimum and maximum value allowed by the solver. As a result, two new boundary values appeared, due to solver limitations instead of the constraints themselves. Those new boundary values are responsible for extra test steps attempting to cover the real boundary values of the constraint. Since those two extra boundary values ‘0’ and ‘268435455’ need to be covered also. The proposed solution is to execute the :remaining valuation method as long as boundary values are still remaining. And as soon as all boundary values have been covered, :random is the applied valuation method. Otherwise, the full coverage of boundary values may take a longer execution time because :random does not focus on returning boundary values. I.e. only probability could make possible to achieve a similar time than the one executing the :remaining valuation method.
3.4 DATA COVERAGE METRICS

Firstly, this chapter will show the application of Goal-Question-Metric (GQM) within this MSc thesis’ main concern: performance. Then the set of developed metrics will be presented and discussed. Important to mention is the fact that the GQM methodology was used concerning the creation of proper metrics for the boundary value analysis context.

3.4.1 GQM

GQM [8] is a framework for developing and maintaining a meaningful metrics program. Goal, Question and Metric are the three phases GQM is composed of. Throughout those steps, this thesis developed a set of metrics which were adopted to measure the strategy’s aspects as performance or effectiveness. This section will illustrate the followed process.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Purpose</th>
<th>Perspective</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>To evaluate the process in order to assess it.</td>
<td>Examine the effectiveness from the point of view of MBT.</td>
<td>The environment consists of boundary value analysis context.</td>
</tr>
<tr>
<td>G2</td>
<td>To characterize the process in order to improve it.</td>
<td>Examine the correctness from the point of view of the organization.</td>
<td>The environment consists of previous data coverage implementation.</td>
</tr>
<tr>
<td>G3</td>
<td>To predict the execution time in order understand it.</td>
<td>Examine the effectiveness from the point of view of the implementation.</td>
<td>The environment consists of computational factors.</td>
</tr>
</tbody>
</table>

Table 3 GQM – Goals

Goal is the starting point of GQM. Goals may be defined for any object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment [8]. In order to carry out correctly that first step, the goal definition template, provided by the GQM methodology, was used. Table 3 shows the goal definition accomplished. Purpose, perspective and environment are the three necessary elements to be defined for each goal.

After the goals have been defined in the second phase, questions related to the proposed goals are suggested. Table 4 shows a relation between such goals and the proposed questions.
During the third and last step, metric definition, the suggested questions during phase two are turned into metrics. Table 5 shows how after gathering questions, metrics were being defined. The number of boundary values, time and constraint complexity were selected as the three main aspects to take into consideration.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Question ID</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>Q1.1</td>
<td>Have all boundary values been covered?</td>
</tr>
<tr>
<td>G1</td>
<td>Q1.2</td>
<td>What time does the strategy need to cover all model?</td>
</tr>
<tr>
<td>G2</td>
<td>Q2.1</td>
<td>Has the new implementation improved the previous data coverage approach?</td>
</tr>
<tr>
<td>G2</td>
<td>Q2.2</td>
<td>Is the new data coverage approach valid for organizational purposes?</td>
</tr>
<tr>
<td>G3</td>
<td>Q3.1</td>
<td>Is the execution time acceptable according to the prediction model?</td>
</tr>
<tr>
<td>G3</td>
<td>Q3.2</td>
<td>Is the strategy solving constraints within the expected time?</td>
</tr>
</tbody>
</table>

Table 4 GQM – Questions

<table>
<thead>
<tr>
<th>Question</th>
<th>Metric ID</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1.1</td>
<td>M1</td>
<td>(\frac{\text{Boundary values covered}}{\text{Total number of boundary values}})</td>
</tr>
<tr>
<td>Q1.2, Q2.1, Q2.2</td>
<td>M2</td>
<td>(\frac{\text{Boundary values covered}}{\text{time spent}} / \text{total time})</td>
</tr>
<tr>
<td>Q3.1, Q3.2</td>
<td>M3</td>
<td>(\frac{\sum \text{constraint complexities}}{\text{time}} / \text{total time})</td>
</tr>
</tbody>
</table>

Table 5 GQM – Metrics

3.4.2 Metrics

Three metrics were defined after the application of GQM methodology. M1 provides information whether all boundary values had been covered after the test execution. That metric responds to effectiveness issues, i.e. the correctness of the testing is guaranteed retrieving 100% at M1. Performance, a key factor of this MSc project, is established by M2 and M3. M2 shows the number of covered boundary values throughout the execution. Additionally, in order to clarify that figure, a graph is provided (see Figure 8). The time specified in seconds, and boundary value coverage in percentage, make the interpretation of the strategy results easier for
users. Moreover, a visual representation of the strategy behavior may allow users to perform further research on the strategy behavior.

![Figure 8 BVA coverage graph](image)

Plain lines in the graph of Figure 8 represent transitions where all boundary values have already been covered but they must be solved again to reach other constraints with uncovered boundary values.

The last metric provided, M3, allows a comparison between the expected execution time of the test and the real time needed. As was mentioned in section 2.4, the formula provides not only a complexity figure, but also an expected solving time. Gathering expected times from all model transitions, a predictability model is proposed; its development will be explained in section 3.5.

### 3.5 RESULTS

This section gathers and presents all accomplished results during the realization of the current MSc thesis. A comparison of the BVA/DataSimple strategies, predictive model development and the proposed architecture are some of those results. An additional section mentions the problems found and how they were handled.

Regarding strategy creation, strategy 2 and 4 were proved as the best proposals along section 3.2.3. On the one hand, strategy 2 bases its coverage on the number of child-states and on the constraint complexities. On the other hand, strategy 4 focuses on less complex constraints. However, trying to reduce the width of the model was the main goal of both of them, hence no statistical difference between their performances was expected. Moreover, the necessity of fewer test steps to achieve 100% boundary value coverage than the other strategies (see Table 2) was an indicator of their possible better behavior.
Strategy 3 and strategy 1 came out as the third and fourth strategies respectively. Although both required 115 steps to achieve 100% boundary value coverage, the extra step done by strategy 1 checking whether all boundary values of a transition have been covered showed up as a drawback for its performance. Finally strategy 5 appeared as the worst strategy. Its initial idea of selective transition choice along all SUT, seemed a good idea, but its high transition processing time turned out to be too large.

It is important to mention that a complete test set has been developed for each strategy in order to prove its correctness and validity. On the one hand, the correctness has been tested providing an exhaustive collection of Rspec unit tests. Rspec [20] is a behavior driven development (BDD) tool which allows to check the correctness of the software developed. On the other hand, the validity has been demonstrated by means of several Cucumber features. Cucumber [21] is a BDD framework that runs automated acceptance tests, i.e.: the functionality of the five strategies is specified with different Cucumber features which will be run against the implementation. The use of these two tool is motivated by the fact that they are the tools employ by Axini.

Once the ranking of strategies, regarding performance, had been established, it was time to confront the best strategy with the current data strategy of Axini, DataSimple. In Section 3.2.4 it was shown that strategy 2 has a better performance than the DataSimple strategy. In addition, the minimum number of test steps required for reaching 100% transition coverage and boundary value coverage is a factor that increases this difference. DataSimple needed a minimum number of 180 steps to retrieve a 100% transition coverage, whereas the BVA strategy – the name given to strategy 2 – required only 115 steps. This is 36% less than the steps needed by the DataSimple strategy to achieve a 100% transition coverage. Taking a look at the boundary value coverage, the BVA strategy achieved 100%, whereas DataSimple only 70%. This difference is due to the fact that the BVA strategy makes use of equivalence partitioning and boundary value analysis for its value selection, while DataSimple looks only for the minimum and maximum possible values and then returns back random values.

Using the Student’s T test to compare both strategies performances and the minimum number of steps required to achieve a 100% transition coverage and boundary value coverage reached by both strategies, the BVA strategy appears to be a more appropriate data strategy than the DataSimple strategy.
After the good results of the BVA strategy, it was the moment to confront it with a real model used by Axini in one of their projects. This model describes the behavior of a microscope. Its behavior may be divided into two main sort of sub-behaviors: parameter checking and parameter updating. In order to be able to use that microscope, a certain combination of parameters must be activated following the next procedure: at any state of the model, if the set of required parameters is activated (parameters checking), perform a parameter update. Regarding the magnitude of the model, it has 510 states and 649 transitions (not annexed due to readability) where a large set of variables (parameters of the microscope) is modifying its value along the model execution. That model was considered as a good test for the BVA strategy since the transition’s constraints present in the model involve many variables which are related among them. Since Axini considers that model as one of the most complicated models they have, a good performance in that model would suggest a good performance in other models at Axini’s.

A similar experiment conditions that in the case study of the medium size model were proposed. A hundred executions for both the DataSimple and the BVA strategy would indicate the average time they need to obtain 100% in both boundary value and transition coverage. Afterwards, a Student T test would be performed in order to compare both times – as in the previous comparison between these two strategies. Although the performance of the BVA strategy solving these large constraints and covering the model is as good as in the medium size model (Annex 1) and it seems better than the performance of the DataSimple strategy, a reachability problem showed up - section 3.5.1 will discuss it. At that point, it was decided to stop the case study and research the cause of such a low transition coverage achieved by both strategies. Despite that reachability issue, the behavior of the BVA strategy remains the same dealing with such a complex model and only a specific problem of that model, makes the test incomplete.

Performance, as key point of this MSc project, was taken into account from different points of view. The time that the BVA strategy required to solve easy and difficult constraints was one of those. Both times were explored, since a large solver time might have led this project to propose a parallel architecture. Two constraints were compared (see Figure 9): a first constraint with a single variable and short domain was confronted with a five multi-dependent unbounded variables constraint. The point was to compare the time that the BVA strategy required to return boundary values back calling directly to the solver with min and max
valuation methods – case A – and the time required in order to pick boundary values applying the process described in section 3.3.1 with the remaining valuation method – case B. After a hundred executions, the first constraint returned back an average execution time of 0.047 seconds, whereas the second constraint handed back a time of 0.050 seconds.

\[\text{A)} \quad i > 0 \land i < 5\]
\[\text{B)} \quad i > 0 \land i < 5 \land j < i \land j < 5 \land k = j \land l > k \land i > m \land k > 2 \land k < 5 \land l > 2 \land i < l \land m \leq j \land m < k\]

Figure 9 Solver – time comparison

No statistical test was required to be able to claim that no important difference was appreciated among both solving times. Since the difference was only 0.003 seconds, a parallel architecture would have probably increased its execution time due to communications and protocol issues. As a consequence, no parallel solution was adopted for this purpose and the only asynchronous part of the system was the one in charge of the metric calculation and drawing the BVA coverage/time graph (see section 3.4.2).

Predictive model viability was another point this MSc thesis researched. I.e. predict the required execution time for covering the SUT. As was mentioned before along sections 2.4 and 3.4.2 with the M3 metric, the complexity achieved was intended to be used not only for constraint complexity classification but for the development of predictive models. Such a predictive model is supposed to give back a figure of the expected execution time that the BVA strategy may need. Figure 10 shows an example of how such matter was handled. A first step analyzing the constraint complexity of the model’s transitions, returned back the expected time. Since no hardware features were specified together with the constraint complexity formula, the expected time to solve a constraint depended of the hardware resources the strategy was running on. Hence, an adjustment of the expected time to cover all MUT was needed. Steps two, three and four (see Figure 10) were responsible for that adjustment.
Detailing this process, it can be said that a comparison was made between the expected time for solving the first transition selected by the strategy using the formula presented in the section 2.4, with the time it needed overall. That comparison was visible through the creation of a coefficient. I.e. expected time/real time. Thereupon, this coefficient was applied to all the transitions of the SUT in order to get a corrected expected time. Nevertheless, even with that adjustment, the figures returned back were too different. Deeper study of the situation revealed that simple constraints seemed to be quite accurately predicted, but due to randomness. Whereas complex constraints were completely wrongly predicted. This fact made all predictive models to be wrong. The point was clear looking at the paper [13], no solver was specified, thus pure mathematics and algorithms solved the constraints by hand. The decision to use a constraint solver was taken at the beginning of this MSc project since it provides faster solutions than solving them by hand. Therefore, the chosen solver was faster than applying mathematics, and afterwards, it could be claimed that such a decision increased the performance of the execution. However, no predictive model could be formed following the intended initial approach.

A new architecture able to deal with parallel computing, was the next point to be investigated. As a first step, already mentioned in this chapter, a study checking the required time by the solver to retrieve a solution was considered. Results showed that the time required by the solver to directly return back a solution and the time needed to get a solution following the remaining or random paths at Figure 7 are similar. Secondly, the cost of calling to the solver was investigated. For that purpose, the previous example of time comparison at Figure 9 was also valid, since the case B is calling more times to the solver asking for the domains of all variables than the case A that is only calling one time for asking for a solution. As in the previous study, the small difference between the times may claim that the
solver call is not expensive in terms of execution. This outcome raised the belief that the implementation of a parallel architecture for the value selection process might report worse figures than a normal architecture. The initial idea to consider a parallel implementation was confronted with different researches, e.g. Sutter and Larus at [22]: “A typical client application executes a relatively small computation on behalf of a single user, so concurrency is found by dividing a computation into finer pieces. These pieces, say the user interface and program’s computation, interact and share data in a myriad of ways. Non-homogeneous code; fine-grain, complicated interactions; and pointer-based data structures make this type of program difficult to execute concurrently”. Hence experimenting acceptable figures both in complex and non-complex constraints, it was decided to keep that part asynchronous and therefore have as the only parallel part of the architecture the one responsible for metrics calculation and graph plotting (see Figure 11) since the operation of drawing the graph is an expensive action.

Figure 11\(^2\) shows a schematic representation of the components present on the architecture. TestManager, Axini product previously mentioned in the Introduction, may be considered as the core of the architecture. Above it, different systems are waiting to be tested; below it, the strategy is depicted how the test is aimed to be performed. The solver and the “Metrics & Graph” components aim to return solutions (Solver) or generate metrics and graphs (“Metric & Graph”). The “Metric & Graph” component is triggered as one of the classes of TestManager every time the test chooses a model’s transition; then its behavior runs in a parallel flow apart from the normal workflow.

The use of such parallel behavior was found extremely useful in terms of performance, since its implementation reduces the whole execution time, especially at the end of the execution when the graph needs to be plotted.

\(^2\) Due to confidentiality issues, it does not show the architecture of the system in full detail.
Before moving on to the next contribution of this MSc thesis, the Fringe strategy needs to be presented. The Fringe strategy - or only Fringe, as it is called by Axini – is probably the smartest strategy that Axini makes use of nowadays. Being able to remember paths where there are states with uncovered transitions – called Fringe paths - is its main strength. Moreover, Fringe is also capable of remembering the specific values which end in that state with an uncovered transition. Long execution time may be considered as its single drawback, since computation of Fringe paths may be a quite protracted process, especially when dealing with large SUTs.

In view of the achievements with the BVA strategy, it was natural to integrate the Fringe and BVA strategies in order to obtain a strategy which may have the benefits of both of them. The first modification simply provides Fringe with boundary values skills when solving transitions. I.e. when there are no Fringe paths, the transition selection process of the BVA strategy takes place. Hence, no more random values are taken at that point and only boundary values are obtained.

The creation of Fringe paths remembering states with uncovered boundary values instead of uncovered transitions may result in a smarter way of achieving 100% boundary value coverage. Thus, that was the second modification. In order to keep the initial behavior of Fringe, Fringe paths with uncovered transitions have preference over Fringe path with uncovered boundary values. Detailing the process, it follows the next steps: 1. Find uncovered transitions and transitions with uncovered boundary values; 2. Compute Fringe paths for uncovered transitions; 3.
If there are Fringe paths from the previous step, deal with them, if not, compute Fringe paths for uncovered boundary values; 4. Deal with the previous set of Fringe paths, if any, otherwise let the BVA strategy select the most appropriated transition. Despite the modifications, Fringe still keeps its main goal of tracking uncovered transitions, hence the adaptation of Fringe to BVA only increases its features and it does not change its original behavior.

As a third adjustment, also trying to keep its original behavior, Fringe paths to uncovered transitions with boundary values are prioritized over Fringe paths to uncovered transitions without boundary values. As a consequence, uncovered transitions will be covered in the first place, as it was initially intended, but a faster boundary value coverage is achieved.

### 3.5.1 Problems found

Transition coverage was the first problem found. Dealing with a model used by Axini during one of its projects and using the BVA strategy, whereas the BVA coverage was 100% (over the transitions already covered), the transition coverage was only 30.02%. At that point, the reaction was trying to find out the cause of such a low transition coverage. The first proposed idea was to apply one of the strategies of Axini: Fringe. The result was the same 30.02%. The second idea was change the one test case with 1000 steps which had been used thus far, for 100 test cases with 100 steps each. The coverage increased to 40.06%. Another change in the parameters was made, 500 test cases with 500 steps each. That time the coverage increased by 1.93%, to 42.53%. No better coverage was obtained with this model, due to reachability issues. It appeared that in order to reach certain states, a strict combination of several transition choices is needed, as they are impossible to reach otherwise. That is due to the fact that the parameters of the microscope are being modified all the time and not having the required values in a certain state the BVA strategy cannot take an uncovered transition. Therefore the BVA strategy chooses another transition which modifies the parameters again. Thus only randomness can turn this process into a finite process, but due to the large number of variables, that probability is quite low.

Regarding this interesting problem of test cases, test steps and boundary values coverage relationship, and extra effort was made in order to have a better understanding of it. A pre-analysis of the SUT, taking into account the model’s transitions and their boundary values, was performed so that the required number of test cases and test steps to achieve a 100% boundary value coverage was known. Furthermore, another functionality was provided to the user, giving him the possibility to choose limit values for one or more of the previous variables and then
automatically receive the relationship among them. This feature increases user freedom in terms of execution time and test exhaustiveness according to his needs.

Specific values allowing to reach certain states, is a problem related to the one previously mentioned. In other words, not selecting those values makes a complete coverage of all model states impossible. Although the BVA strategy may be responsible for that, keeping track of all demanded values for each single transition was perceived as an unviable option due to its complexity - especially handling large models. The use of the solver came out as a valid solution. The idea was to identify transitions where states variables are assigned, so that the value selection may be customized. Figure 12 exemplifies the followed process. First, it is checked whether the transition being processed has any kind of impact on the rest of the model transitions. If yes, the affected transitions are found and retrieved. Straighway, the initial transition and the transitions affected by that initial transition are joined. Afterwards, this combined transition follows the aforesaid value selection process in section 3.3.1. The solution which is obtained by the solver may trigger thus far unreached states. In the case that no transition is affected by the initial transition, it follows the value selection process already mentioned and the process finishes.

An increase of the strategy reachability was obtained thanks to the process already explained. Nevertheless, reachability issues derived from the execution of a specific transition combination, as the first problem mentioned during this same section, were still unsolved. However the practical applicability of the BVA strategy
is not compromised since that reachability issue is not commonly found, being a special case from that model.

### 3.6 Discussion

The _min_ and _max_ valuation methods, previously mentioned in Section 3.3.1, were used to achieve both the minimum and maximum values that satisfy a constraint. On the one hand, _min_ and _max_ are interesting valuation methods, since they retrieve the domain boundaries. On the other hand, the _remaining_ valuation method is able not only of returning those minimum and maximum values but all boundary values present in the constraint. Hence it might be considered that _min_ and _max_ base their utility on the clarity of their results, since using _remaining_ it is more difficult to figure out when the maximum value has been reached. Finally, the first option was selected. Although _remaining_ was able to perform the same results as _min_ and _max_ and a final user would not notice the difference, keeping them made debugging easier, because it was simpler to recognize when an error was caused by the minimum or the maximum value. Thus it was preferred to keep the four valuation methods: _min, max, remaining_ and _random_.

The value selection process discussed in Section 3.3.1 was implemented having in mind the time constraint this MSc thesis had. Hence the process did not follow the most efficient path. Figure 7 (in Section 3.3.1) shows a quite inefficient way of reforming an impossible solution, coming back to the “Get domain” state instead of going straightly to “Form solution”, which would be more efficient. However, the execution time was not highly affected since the case of forming an impossible solution was unusual. In order to solve this problem, a refactoring process may be proposed. Such refactoring was considered to be out of this MSc project scope since several files may be involved; in any case, it might be part of a related future work.

Development of a predictive model was another point explored during the realization of this thesis. With an initial idea of using the formula presented in Section 2.4, a predictive model able to suggest the time BVA strategy would require to cover the model, was implemented. However, the aforementioned results showed the incorrectness of the initial assumption. In fact, the idea of developing a predictive model would have required a more extensive study, since not only software is involved. Hardware aspects as processor or memory workload have a high impact on performance figures, hence a preliminary analysis of hardware capabilities should have been done, as the coefficient proposal is not
adequate enough. Certainly, the elected approach handling the creation of a predictive model might be considered as a good starting point, but this topic definitely requires of a more extensive research.

Parallel computing seemed to be an exceptional point to be covered along this MSc thesis, since a performance increase may be reported. However, in some cases that previous statement is not entirely true. When the problem is simple enough and the retrieved execution time is acceptable, such a parallel implementation may not be necessary. Sutter and Larus at [20] explicitly mentioned data sharing as one of the causes of bad performance using parallel computing, and in fact, the architectural proposal was based on data sharing among different modules. Thus taking into consideration literature, it was agreed to discard such an architecture and focus effort on other areas of the thesis.

The last discussion point which needs to be mentioned is the usage of the GNU Prolog solver [18]. Although it has been shown how Gprolog has been used along this MSc project in order to solve constraints efficiently (see Section 3.3), it also has been mentioned how its returned domain is not accurate in some cases (see section 3.3.3). This fact decreased the global efficiency of the BVA strategy. At that moment, a discussion about the use of Gprolog might have arisen, nevertheless it did not come out since its constraint solver feature retrieved very good results. Moreover, such imprecise domains were normally returned back due to unusual constraint forms, e.g. incomplete constraints where not all variables appear in all sub-constraints. Hence its appearance might be limited to rewriting constraints to a form which it is better understood by Gprolog.
4. RELATED WORKS

This section will present five different works which have been taken as a reference for the realization of this MSc thesis. “Test Models and Coverage Criteria for Automatic Model-Based Test Generation with UML State Machines” [23], and “Combining combinatorial and model-based test approaches for highly configurable safety-critical systems” [24] are examples of these six researches.

“Test Models and Coverage Criteria for Automatic Model-Based Test Generation with UML State Machines” [23] appeared as a first related work. In that study, a new theoretical approach to test suite generation combining not only boundary value analysis but also data-flow-based, control-flow-based, and transition based coverage criteria is proposed. The research methodology may be expressed as a model transformation from an original model to an UML state-machine. As a consequence, several benefits are achieved, for instance a higher rate of fault detection or a guideline to combine models in order to reduce test costs. The main difference between this MSc project and the previous research is how to address constraint solving, and therefore how to handle domains. Meanwhile our research decided to make use of a constraint solver such as Gprolog, they decided to use a mathematical solution deriving constraints.

A second relevant research titled “Combining combinatorial and model-based test approaches for highly configurable safety-critical systems” [24] presents a theoretical approach combining MBT with combinatorial techniques such as equivalence partitioning, boundary analysis, or n-wise parameter coverage. It introduces a systematic model-based test approach for parameterized systems with a large configuration space, emphasizing the complexity of the testing process dealing with that large number of configurations. Accomplished results provided a test automatization of relevant test cases reusing the similarities between configurations. They applied equivalence partitioning and boundary value analysis with the purpose of reducing the domains of the different configuration items in order to be able to reuse as many test cases as possible.

Another related study is the one named as “Model-Based Testing in Practice” [25]. It focuses on the application of combinatorial test generation techniques to large projects, with the goal of checking what MBT aspects work in practice and what may report a problem for testing organizations. Their results show how the use of boundary value analysis resulted in more defect detection. Regarding their
boundary value analysis, they preferred either the manual calculation of it or obtain it from the specification of each value. Hence no domain was implied.

The usage of equivalence partitioning and boundary value analysis within classification tree techniques in order to automatize some phases of MBT, is the main aspect discussed at [26]. A raw classification tree was turned into a classification tree, where almost all input data is gathered in equivalence partitions, by a set of transformation rules. I.e. the classification tree is being built up as a result of using equivalence partitioning and boundary value analysis on input values. As a consequence, a reduction of human activities along the MBT process was achieved.

Condition & Decision Coverage (MCDC), Classification tree and Exploratory methods are the tree unit testing methodologies – in the model-based development context - confronted at [27]. The three methodologies were employed during the realization of three different projects with the goal of comparing their effectiveness. Since it used unit testing, statement, decision and condition coverage were the three aspect taken into account for such comparison. Results showed that MCDC with boundary value analysis was the most productive methodology, whereas exploratory methods the least.
5. CONCLUSIONS

Along this last chapter, contribution and future work sections will be discussed. Firstly, important achievements of this MSc thesis will be summarized; afterwards, possible derived projects will be proposed. Summing up, this chapter will help to close the current document having a clear idea of the covered points during its realization.

5.1 CONTRIBUTION

An increase of the MBT quality performed by Axini, improving its data coverage is the main contribution of this MSc project. Thus far, Axini is testing only the minimum and maximum values which satisfy the model constraints, and then random values are taken. This situation was derived from the combination of transition and state coverage approaches for MBT applied by Axini. Nevertheless, the data coverage approach is not incompatible with them, as was shown along the realization of this project. Equivalence partitioning and boundary value analysis, have been successfully applied and integrated into the Axini MBT context. As a consequence, all model’s boundary values are tested and a good performance in both transition and boundary value coverage has been achieved. Moreover, the applicability of the BVA strategy has been proved thanks to the two case studies accomplished – medium size and real models. As a result, the utilization of the BVA strategy for Axini’s testing process is only a matter of adapting the code to Axini’s style.

Although a worse performance might be expected from the beginning of the project due to the possible necessity of covering a transition more than one time to test all its boundary values, statistical tests showed the opposite. The BVA strategy reached 100% of both boundary value and transition coverage even faster than the DataSimple strategy – the current data strategy of Axini. That improvement in the transition coverage is due to the fact that normally boundary values are the values which trigger uncovered states and transitions. Moreover, results of the BVA strategy have been combined with the Fringe strategy, a fact which resulted in an excellent combination between transition and data coverage within a single strategy.

Apart from the BVA strategy, more contributions have been provided. A set of metrics have been developed using the GQM methodology in order to be able to assess the BVA strategy’s performance. Furthermore, a graph showing the
boundary value coverage evolution along execution has been added, increasing its readability and understandability. In addition, the graph itself provides an extra tool to check the BVA strategy behavior, since analyzing its trends, it was possible to understand how the strategy is covering both boundary values and model. Moreover, performance is not compromised by that fact, since metric calculation and graph plotting are allocated outside from the main process in a parallel thread.

Having performance as an important issue, the possibility of applying restriction values for the testing, is provided to the user. Boundary value coverage and number of test steps and test cases, are these limit variables. As a consequence of adopting one or more restriction values, users may customize MBT according to their requirements and constrictions.

Typically, both equivalence partitioning and boundary value analysis are techniques applied to numerical values. Nevertheless, this project researched its utilization – in Axini context - to other elements as Booleans, Strings and Enumerables (Arrays and Hashes) with regard to the BVA strategy.

Summarizing, the BVA strategy increases Axini’s MBT quality, offering a better data coverage with similar execution time. A fact which is important for clients of Axini that employ data testing.

5.2 Future Work

Reachability was an issue out of scope for this MSc thesis since it is an MSc proposal by itself. Therefore the only reachability issue considered was merging constraints in order to know the values which enable certain transitions (see 3.5 Results). It turned up as a valid solution, but only for cases where values triggered new paths. Nevertheless, such a solution was ineffective when a specific order of transitions was required (see 3.5.1 on problems found). That situation demands for its own research since there should be a better way to confront such problem than trying all possible combinations of transitions. A theoretical proposition taking into consideration the aforesaid loco theory – working with symbolic paths - might be a good approach for it.

The BVA strategy may be considered as the main contribution of this MSc thesis, a strategy capable of obtaining boundary values from constraints. A similar approach may be taken for further researches e.g. using another data testing procedures as, for instance, pair-wise testing. This testing methodology test not only interesting values, but also smart combinations of them. Currently, the BVA strategy returns
back boundary values whenever one of them is available. Nevertheless it may be the case that a certain combination of values triggers unvisited model paths – an aforementioned problem but with a set of values this time -, a Pair-Wise strategy might be the solution. On the one hand, that strategy might result in longer execution times due to a large number of value combinations, but on the other hand, it would perform a really extensive data coverage. Hence depending of the MBT purpose of each client, the viability of its development might be considered as a future option.

Developing a new collection of metrics might be considered as another point which may need further research. As previously mentioned, the achievement of an acceptable performance was one of the important points this MSc project required to fulfill, because a poor performance would have become BVA strategy a useless strategy. The GQM methodology was applied having in mind such a performance scenario. However, further effort working with BVA strategy might create a new scenario where performance was not the main issue. In such case, a new set of metrics should be developed. In addition, the GQM would be considered as a good methodology to use for that future situation as it was for this project.
ACRONYMS

BBD – Behavior driven development
BVA – Boundary value analysis
CSP - Constraint satisfaction problem
GQM – Goal Question Metric
IOCO – Input/output conformance
IOTS – Input/Output transition system
LTS – Labelled transition system
MBT – Model-based testing
MCDC - Condition & Decision Coverage
MUT – Model under test
SUT – System under test
TDD – Test driven development
REFERENCES


ANNEX

1. MEDIUM SIZE MODEL