Márcio Filipe Caetano Mateus

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Measuring Data Transfer in Heterogeneous IoT Environments

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientadores : Adolfo Sanchez Steiger Garção, Professor Catedrático, FCT-UNL
Pedro Miguel Negrão Maló, Professor Assistente, FCT-UNL

Júri:

Presidente: Doutor Rodolfo Alexandre Duarte Oliveira - FCT/UNL

Arguente: Doutor Manuel Martins Barata - ISEL/IPL

Vogais: Doutor Adolfo Sanchez Steiger Garção - FCT/UNL
Mestre Pedro Miguel Negrão Maló - FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Junho, 2012

Measuring Data Transfer in Heterogeneous IoT Environments

Aos meus pais e à minha irmã

Acknowledgements

Diversas pessoas contribuíram e suportaram o desenvolvimento da dissertação. De entre estas pessoas destacam-se:

O mestre Pedro Maló, pela oportunidade e suporte dados, bem como pelos conhecimentos partilhados.

O mestre Bruno Almeida, pelo tempo dedicado, pelas sugestões oferecidas e pelo rigor requerido, contribuindo para uma melhoria da qualidade do trabalho. O Tiago Teixeira pelo companheirismo, pelas sugestões que contribuíram para o desenvolvimento das ideias apresentadas, pelos momentos de descompressão.

Todo o pessoal do Gris, nomeadamente o Henrique Baeta, Jorge Casanova, e Hugo Pereira, Raquel Melo e Edgar Silva pelo apoio e conselhos.

Todos os amigos da FCT pelos momentos de diversão que facilitaram nos momentos mais difíceis.

Os meus pais e irmã pelo suporte e apoio condicional, e à minha sobrinha pelos momentos de distracção que me ajudaram a descontrair.

Abstract

Internet of Things (IoT) is characterised by the heterogeneity of the used devices, which leads to information exchange problems. To address these problems, the Plug'n'Interoperate approach is used, where the steps needed to perform the information exchange between devices are described by interoperability specifications (IS) and are operated by the devices. However, more than one IS can exist to describe the information exchange between each pair of devices, so to choose the suitable IS, there is the need to measure the information exchange described by each one. To do this, there already exist some methods. But, they rely on a deep understanding of the IS and the data formats involved. To overcome this, an advanced measurement method is presented. This method advances by measuring the data transfer provided by an IS, without the need of specific knowledge about it. This measurement does that, by relying only on an abstract view of the data transfer and providing results that allows the benchmarking of the entire interoperability performance of the IoT environment. Thus allowing the comparison of different IS without the need of being specialized on them.

Keywords: Internet of Things, Interoperability, Measurement.

Resumo

A *Internet of Things* (IoT) é caracterizada pela heterogeneidade dos dispositivos utilizados, o que pode originar a ocorrência de problemas na troca de informação. A abordagem *Plug'n'Interoperate* é utilizada para lidar com estes problemas através da descrição dos passos necessários para realizar a troca de informação entre cada par de dispositivos. Estas descrições são chamadas Interoperability Specifications (IS) e são manipuladas pelos dispositivos. Mais de uma IS pode existir para descrever a troca de informação entre cada par de dispositivos. Assim, de modo a escolher a IS mais apropriada para ser utilizada, existe a necessidade de medir a troca de informação descrita por cada IS. Existem algumas abordagens que permitem analisar IS, contudo estas precisam de um conhecimento profundo sobre a IS e formatos de dados utilizados. Para superar estas limitações, é apresentado um método que difere por permitir a medição da troca de transferência de dados descrita num IS sem a necessidade de a conhecer. Esta medição baseia-se numa abstracção da transferência de dados onde são analisados os resultados da aplicação de IS, permitindo a análise da performance da interoperabilidade num ambiente IoT. Esta abordagem permite a comparação de diferentes IS sem a necessidade as interpretar.

Palavras-chave: Internet of Things, Interoperabilidade, Medição

Acronyms

AMR Abstract Model Representation

ASF Algebraic Specific Formalism

AST Abstract Syntax Tree

ATL Atlas Transformation Language

DIT Depth of an Inheritance Tree

DTr Data Transfer ratio

GRIS Group for Research in Interoperability of Systems

ICT Information and Communication Technologies

IEEE Institute of Electrical and Electronics Engineers

ISO International Organization for Standardization

IoT Internet of Things

ITU-T International Telecommunication Union Telecommunication Standardisation Sector

NOC Number Of Children

OCL Object Constraint Language

OSI Open System Interconnection

PhD Doctor of Philosophy

PnI Plug and Interoperate

SDC Source Data Count

SDF Syntax Definition Formalism

SMC Source Model Count

TDC Target Data Count

TMC Target Model Count

TTCN Tree and Tabular Combined Notation

WMC Weighted Methods per Class

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motivating Scenario: Plug and Play Interoperability

Internet of Things(IoT) consists in a network of objects, which can be connected to the Internet (CERP-IoT, 2010). The use of such interconnected objects can lead to improved situational awareness and enhanced control in target environments via comprehensive sensing and actuation. For instance, let us consider a food storage scenario. Food is an essential resource to all the living beings however, perishable food easily spoils if it is not stored under the appropriated conditions. If spoiled food is ingested then it can harm the living beings health.

A monitoring system is needed to identify spoiled food based on the record of the conditions to which the food was subject. For that, let us consider a food scenario composed by a warehouse and several boxes designed food storage, where each box is equipped with a sensor containing a thermometer and a hygrometer to measure the temperature and the humidity inside the box. Each sensor senses the conditions inside its box and sends it to the warehouse's monitoring system.

This scenario can be implemented using wireless sensor networks being each sensor a wireless sensor, not only capable of sensing the information, but also able to inject it in the wireless sensor network, where it will hop from one wireless sensor to another until it reaches the warehouse's monitoring system. This scenario is illustrated in Figure 1.1. The implementation of this scenario using wireless sensors provides some advantages as: the mobility of boxes since they need not to be connected to a fixed structure and easy configuration since only is needed the configuration of the new box to connect it into the network because then the network will configure itself. These advantages justify the increase of interest in wireless sensors development.

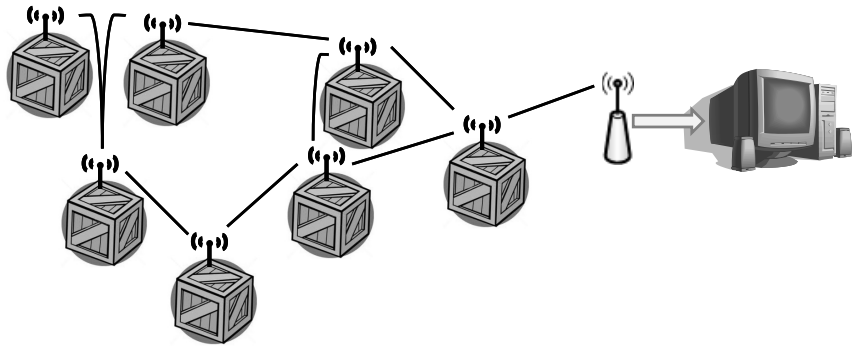

Figure 1.1: An example of an Internet of Things environment

This interest in wireless sensors development led to the production of many types of devices, implementing several communication protocols, and designed for different application scenarios (Atzori, Iera, & Morabito, 2010). The communication between these devices can occur at several levels such as: *a)* physical level which concerns with the physical connection between the devices, both by cable and by air; *b)* data level concerned with the exchange of data between the devices. This diversity of devices can raise communication issues at all levels if no standards are adopted. Regarding to the physical level, standards as the IEEE 802.15.4 standard were defined and accepted (Callaway et al., 2002). However, with respect to the data level, there is still missing a standard accepted by the community which leads to the occurrence of interoperability problems.

Interoperability is defined by the IEEE as: "*the ability of two (or more) systems or components to exchange information and to use the information that has been exchanged*" (IEEE, 1990). This definition implies that interoperability between systems consists in the composition of two process, which in the data level scope have the following interpretation: *1)* the exchange of information between systems consists in the agreement of the data format used to represent the information; and *2)* the use of the information exchange consists in the correct interpretation of the information exchanged.

The definition of interoperability can be better understood through an example, for this purpose, let us return to the example shown in Figure 1.1. The first interoperability process refers the data format chosen to support the exchange of information. Considering the information acquired by the thermometer, the data format can be used to specify the data type used to represent the temperature value (e.g. integer, float or string). Without this specification a a sensor could represent the temperature as a integer and the monitoring system could read as a string, obtaining a value different from the value sensed. The second process corresponds to the interpretation of the information correctly exchanged. For instance, one sensor reads the temperature in Fahrenheit degrees and the monitoring system interprets the it as a value in Celsius degrees. In this case the data value is corrected but the information interpreted is incorrect as it corresponds to a temperature value different from the sensed.

In order to address the interoperability problems that occur in the data level, the concept of Plug'n'Interoperate (PnI) has been defined in Uninova - GRIS, research group where this work

was developed. The PnI is based on the principle that the devices used in IoT are heterogeneous. Heterogeneous devices are devices that implements different standards and are designed for different purposes, which usually results in the use of different data formats. In order to allow the exchange of data in the system without the need to re-manufacture the devices, the manufacturers need to provide some specification that indicates how can the data be converted from the data format used by the device to another data format. The PnI aims to the management and execution of these specifications.

The scenario illustrated by Figure 1.1 can be used as an example of PnI. Consider that a new box enters in the warehouse that uses a specific data format. If this data format is unknown to the data exchange system, the data of this new box cannot used by nodes in the network of boxes or by the monitoring system. However, the existence of a specification that tells the system how to convert the data format of the new box into a data format known to the data exchange system, allows the use of the data. Therefore this specification assumes a special relevance in the concept of PnI and are called interoperability specification. Figure 1.2 illustrates the definition of interoperability specification. In this figure is represented an data transfer between two systems. The system that wants to send data is seen as the source system and the data receiver system is seen as the target system. The data transfer function is defined by a interoperability specification. Interoperability specifications can also be heterogeneous as they can be defined using different technologies/languages.

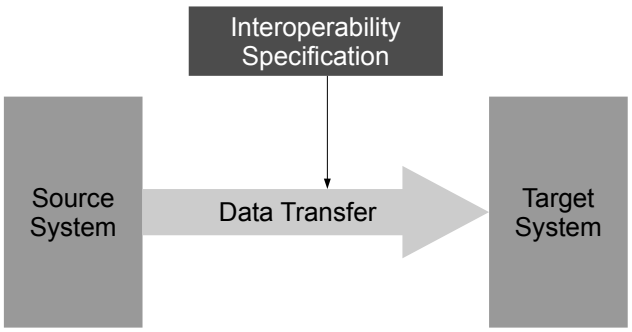

Figure 1.2: Data exchange between two systems provided by an interoperability specification

The addition of new devices to a system potentially implies the increase of the number of interoperability specifications that the system needs to manage. This fact can lead to the existence of several interoperability specifications for the same pair of data formats, where some may provide a better data exchange than the others. This diversity of interoperability specifications can be used to improve the interoperability potential.

The improvement of the interoperability potential within the environment can be achieved through the maximisation of the data exchange. The improvement of interoperability potential achieves its maximum when there are, within the environment, interoperability specifications between all known data formats, and each one of these interoperability specifications maximises the data exchange between the source and target data formats.

Several options exist to improve the interoperability potential of a system, as for example: *a)* the selection of the best interoperability specification, i.e. the interoperability specification that provides the best data exchange, between each pair of data formats; *b)* the comparison of all the interoperability specifications known by the system, regardless to the pair of data formats that they correspond, in order to improve the most suitable interoperability specifications.

Lord Kelvin once stated that: "*if one can not measure it, one can not improve it*". This conclusion can also be achieved through the analysis of the previously listed options where there is a common need: the need to determine/measure the actual data exchange provided by an interoperability specification.

1.2 Problem: Measure Data Exchange

In IoT, or more specifically in PnI scenarios, the measurement of the data exchange requires the establishment of relations between the concepts used by each device involved in the data exchange, in order to identify the correspondence between concepts. However, the lack of globally accepted standards for the definition of these concepts can hinder this approach. The measurement also needs to know the steps required to perform the data exchange, which are described in an interoperability specification. However, the heterogeneity characteristic of interoperability specifications can become an hurdle. In order to allow the improvement of data exchange within the environment, the best interoperability specifications must be selected to be used and the more suitable to be improved must be identified. This need implies that the measurement result must be comparable. These facts leads to the following research question:

How to measure the data exchange provided by an interoperability specification in an Internet of Things scenario?

The goal of this work is the answer to this question. To accomplish that, the characteristics of the measurement problem in an IoT environment, namely in a PnI scenario, need to be clearly identified and studied. The characteristics are: *a)* the lack of semantic definitions, *b)* the heterogeneity of the interoperability specifications, and *c)* the need of generation of comparable results.

Lack of Semantic Definitions

According to the experience of the research group in relation with the devices used in IoT environments, usually the manufactures of this kind of devices only provide the data formats used by the devices, and do not specify the semantic classifications between the concepts related to sensors. This lack of semantic definitions is mainly due to the fact that does not exist a globally accepted standard for the definition of the semantic concepts used in IoT applications(Katasonov, Kaykova, Khriyenko, Nikitin, & Terziyan, 2008). This characteristic makes difficult the measurement of the data exchange based on semantic relations, being preferential the use of another approach.

Heterogeneity of Interoperability Specifications

In order to measure the data exchange, the steps needed to perform this exchange must be assessed, being this information described in interoperability specifications. However, as interoperability specifications artefacts can be implemented using different technologies, which hinders the development of a general assessment of these steps using these artefacts. Therefore, a measurement approach must be defined that overcomes the obstacle imposed by this characteristic.

Comparison between Different Pairs of Data Formats

The measurement process must produce an output able to be comparable in order to allow the draw of conclusions about the data exchange provided by several interoperability specifications. This comparison must, not only, be performed between interoperability specifications for the same pair of data formats, but specially between interoperability specification defined to different pairs of data formats. If no relation can be identified or established between the results of the measurement process for different interoperability specifications, then no conclusion can be reached and therefore, no improvement can be performed. While the first comparison does not poses a big problem since the data formats used are the same, the second kind of comparison corresponds to a harder task since the data formats are different, changing the measurement context. The measurement output must allow the sorting of interoperability specifications based on the data exchange performed by each one. This requirement results in the need of establishing relations "higher than", "lower than", and "equals to" between the outputs of the measurement approach, forcing the result of the measurement approach to be comparable.

1.3 Work Methodology

The work methodology followed by this thesis is based on the basic principles of scientific method described in (Schafersman, 1997). The used methodology is illustrated in Figure 1.3, and is composed by the following seven steps:

1. Characterise the problem;
2. Do a background research;
3. Formulate hypothesis;
4. Set up an experiment;
5. Test hypothesis through experimentation;
6. Draw conclusions;
7. Publication of results.

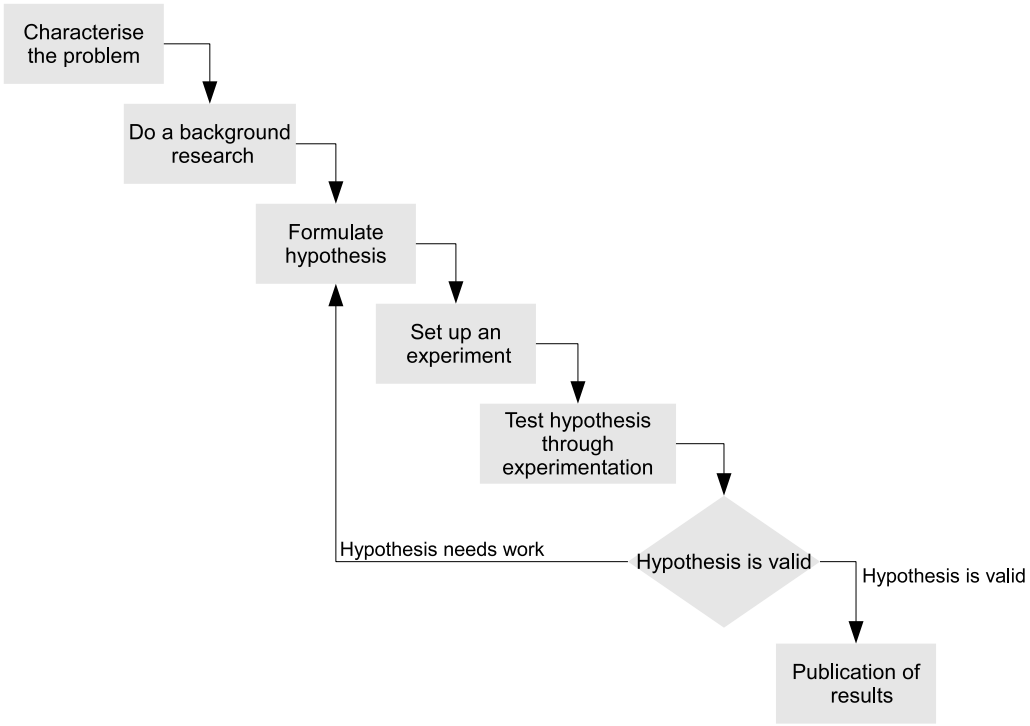

Figure 1.3: Research methodology used in this thesis

1. Characterise the problem

In this step the problem is identified and characterised, through the study of the established characteristics. In this step is also formulated the research question that will be the basis of the research work. The identified problem in this work is to measure the data exchange provided by an interoperability specification in an IoT scenario.

2. Do a Background Research

In this step is performed the study of prior work , that is related with the research question formulated in the first step. In this study the characteristics of the problem are taken into consideration in order to perform the analysis of the prior work. Using this analysis is identified the contribution of prior work for the solution of the research question as well as the advancement that this work aims to introduce. In this work is performed the study of approaches designed to measure the quality of model transformations, since model transformations can be used to represent interoperability specifications.

3. Formulate Hypothesis

Based on the background research, it follows for a conceptual achievement that serves as the research hypothesis. The hypothesis should enable a conceptual approach and define a specification in order to allow the elaboration of an experiment. In this work the hypothesis consists in a measurement method the uses the number of data elements in the source and target data models, related by a mathematical expression , to perform the measurement.

4. Set up an Experiment

This step consists in the technological realisation of the hypothesis through the implementation of the specifications defined in it. This implementation is designed to be used as a proof-of-concept, built only to test the validity of the hypothesis.

5. Test Hypothesis through Experimentation

In this step are defined the tests which the implementation of the hypothesis will be submitted to. These tests are designed in order to gather results that allow the evaluation facing the characteristics of the problem. The tests are performed using the implementation. All the tests must be executed in a controlled environment in order to control all the results of the experiment and ensures that these testing can be reproduced.

6. Draw of Conclusions

To assess the proposed solution, the results of the tests performed in the previous step are checked confirming if the hypothesis complies with the characteristics identified in the problem. If the tests fail, back to step 3, where the hypotheses is subject to work, until there is a new hypothesis that successfully answers the research question.

7. Publication of Results

The last step consists in the publication of the results and experience obtained in the research work. This publication can assume the form of a final report and / or published in a scientific publication. The publication of this thesis document is included in this step.

1.4 Dissertation Outline

This dissertation is composed of five chapters, where the first is the present one:

Chapter 2 presents the background research conducted in this work. This research focuses in the identification of approaches designed to evaluate the performance of model transformations. Four approaches are identified and analysed, being elicited the contribution of prior research background to the development of the hypothesis.

In Chapter 3 is presented the measurement method proposed to solve the problem identified in the Introduction. In this chapter is defined the theoretical concepts associated to the measurement method and all the steps of the measurement method are specified and presented as components.

In Chapter 4, which corresponds to the Testing and Validation, is described the adopted testing methodology and notations. Using these approaches, an abstract test is defined and tests are executed having in mind the validation of the hypothesis against the characteristics of the problematic identified in the Introduction. After the execution of each test, the results obtained are analysed and a verdict is drawn.

Chapter 5 corresponds to Conclusions and Future Work. In this chapter is summarised the content of each one of the previous chapters, in order to support the drawn of conclusions about the developed work. In this chapter the publications carried out during the thesis time are listed and future developments proposed and exposed.

2

State of the Art

2.1 State of the Art Review

An extensive research was made in order to identify technologies which use the concept of interoperability specification, and for each technology, approaches to measure it. In this research work four approaches were identified and all of them use model driven concepts. This fact creates the need to perform a brief study of these concepts before the starting with the study of each one of the identified approaches. Therefore, the concepts of metamodels, models, and model transformations are addressed, being also referred the concept of element mapping.

A model describes the characteristics of a system, in a concrete viewpoint, using a well-defined language called modelling language which have well-defined syntax and semantics (Singh & Sood, 2009). This modelling language is in its turn defined by a higher level model called metamodel which consists in a specification model for a class of the system, making statements about what can be expressed in the valid models of a certain modelling language (Seidewitz, 2003). A metamodel is described by a specialized language called metalanguage. This generic modelling approach is shown in Figure 2.1.

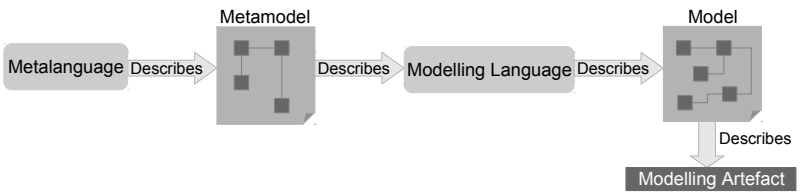

Figure 2.1: Generic model approach

In the domain of model driven approaches model transformations are used to exchange information between models (Sendall & Kozaczynski, 2003). Models transformations are composed by a set of transformation rules that allows the mapping between the elements of the source models and the elements of the target models. These rules are well-defined by a model transformation language (Jouault & Kurtev, 2006; Czarnecki & Helsen, 2006). In Figure 2.2 is shown a mapping example of a model transformation where elements A and B of the source model will be represented in element X of the target model through the execution of the transformation rule f and the data in element C will be represented in element Z through the execution of the transformation rule h.

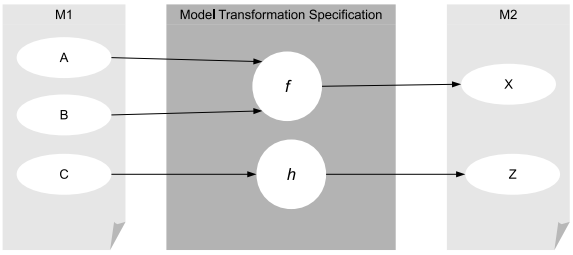

Figure 2.2: Example of a Model Transformation mapping

The execution of a model transformation is performed at the data elements level, transforming the data values in accordance to the element mapping defined. Therefore, to perform a model transformation some specific model artefacts are needed: one or more source data containing the information to be transferred, a source model for each source data to identify the elements in each source model, one or more target data to receive the information transferred, a target model for each target data identify the elements in each target model, a model transformation artefact to describe the element mapping and a model transformation execution engine to execute the information transfer function in accordance to the model transformation artefact. The organisation of these artefacts is represented in Figure 2.3.

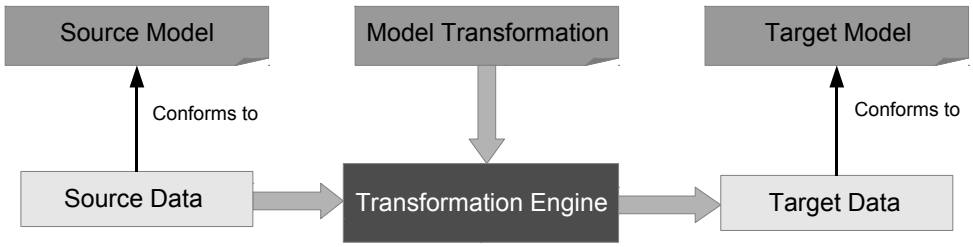

Figure 2.3: Diagram of a model transformation execution

2.1.1 Individual Analysis of Measurement Approaches

Approach 1: Measurement of Metrics of the Model Transformation

Marinus van Amstel proposes in his PhD Thesis (Amstel, 2011) an approach to assess and improve the quality of model transformations in order to support the development and maintenance of

model transformations. To achieve this objective Amstel defines seven quality attributes that can characterise a model transformation artefact. These quality attributes are:

- Understandability refers to the effort needed to understand the purpose of a given model transformation;
- Modifiability represents the difficulty of modifying a model transformation in order to provide different or additional functionality;
- Reusability expresses the ability of parts of a model transformation being reused by others model transformations
- Modularity reflects the extend in which a model transformation is systematically separated and structured, e.g. grouping rules in modules;
- Completeness expresses the degree in which a model transformation correctly transforms models conforming to its source metamodel to models conforming to its target metamodel;
- Consistency represents the uniformity used in the implementation of a model transformation, namely in the programming language used;
- Conciseness refers to the quantity of superfluous element defined in a model transformation, e.g. declaration of unused variables.

In order to assess each quality attribute, quality metrics are defined. These quality metrics can vary with the technology used to implement the model transformation, however Amstel states that conceptual similar metrics can be defined for different model transformation languages. Amstel, in his work defines the metrics for ASF+SDF and ATL model transformations and build tools to measure these metrics.

With the objective of establish a relation between the quality metrics and the quality attributes, surveys are made to experts about a set of model transformations where were classified each quality attribute for each model transformation. In this classification a number between 1 and 7 is assigned to each quality attribute, where a higher number corresponds to a greater quality. The results obtains are then compared with the quality metrics of the corresponding model transformation in order to understand which quality metrics have influence in each quality attribute.

The process of extraction of the metrics is depicted in Figure 2.4. In this process, the metrics are obtained from the model transformation artefact through the use of a metrics extractor which produces a model that contains the metrics data. The data in this model is then presented as a report. This report is generated by a metric processor that can perform operations over the metrics in order to present the data in a specific form.

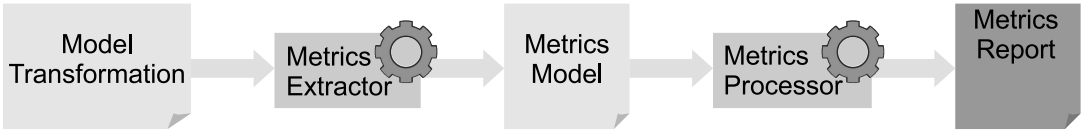

Figure 2.4: Metrics extraction process

ANALYSIS

This approach produces quality metrics and attributes. Quality metrics are dependent from the transformation language which means that they can only be used to compare model transformations described in the same transformation language and with the same purpose. On the other hand, quality attributes can be used to compare any model transformation as quality attributes are independent of the transformation language used are expressed as a number between a well-defined range of numbers (1-7).

Despite the fact that this approach is designed to assess and improve the quality of model transformations, it presents the quality attribute Completeness that can be associated with the data exchange. However, as this approach performs the measurement using only parameters of the model transformation artefact, there is no guarantee that all the data within the models is handled by the model transformation.

Another issue arises by the use of parameters of the model transformation artefact to perform the measurement is that this operation requires a deep understanding of the technology in which the model transformation is described. As model transformations can be described using different transformation languages, with this approach, there is the need to choose the more suitable metrics and to produce a metrics extractor for each transformation language since the metrics supported can differ from transformation language to transformation language.

Approach 2: Measurement of Metrics of the Models

Motoshi Saeki and Haruhiki Kaya propose in (Saeki & Kaiya, 2007) an approach to identify the model transformations that can improve the quality of models. To achieve this objective the authors resort to quality metrics. They state that if the values of the metrics increase with the execution of the transformation, then that transformation improves the model quality. As result, the authors propose the introduction of model-specific metrics in the models since the metrics of a model transformation can be defined from the metrics of the models used in the transformation.

The model-specific metrics are introduced in the model through the extension of the metamodel. In this extension metrics, as WMC (Weighted Methods per Class), DIT (Depth of an Inheritance Tree) and NOC (Number of Children) are defined as classes, and their calculations methods can be defined as a constraint written in OCL, being both embedded into the metamodel.

In relation to the model transformation, it can also be modified in order to use the metrics of the source model. These metrics can be used, as example, in conditions that unable the transformation if the values of the metrics are lower than a minimum value, allowing the execution of the transformation only when the model has a quality higher than a certain standard. After the execution of the transformation, the constraints defined in the target metamodel will be responsible for the calculation of the values of the metrics in the target model. Comparing the metrics in the source model with the ones in the target model is possible to calculate how much the quality was

improved, or degraded. This process is illustrated in Figure 2.5.

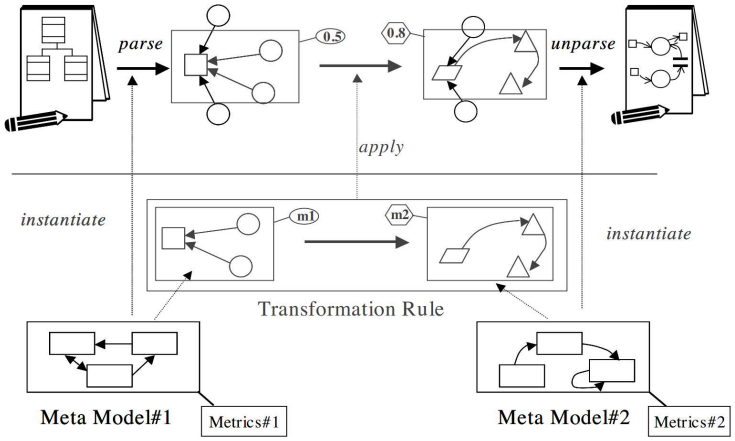

Figure 2.5: Model Transformation Process (Saeki & Kaiya, 2007)

ANALYSIS

This approach proposes the introduction of model-centric metrics in the source, target, and model transformation metamodels. This methodology requires the modification of the metamodels, option that is not always available.

The measurement is performed through the comparison between the metrics in the source model and the target model which makes the measurement methodology independent of the transformation language used.

Regarding to the result of the measurement, this approach does not propose any specific result and therefore does not allow the draw of a conclusion about the comparability of the results.

Approach 3: Model Transformations Verification using Assertions

In (Asztalos, Lengyel, & Levendovszky, 2010) is proposed an approach focused in the verification of model transformations, which consists in proving some functional and non-functional properties of model transformations, as well as properties of the models used in the transformation. The goal of the authors is to provide an automated verification framework to formally analyse model transformations.

The approach proposed is strongly based in the use of assertions. An assertion consists in a formal expression that can be used to state properties of the models used in the transformation, properties of the transformation rules, or the modifications produced by the transformation rules. Assertions can be classified as True or False depending if the properties respect the logic expressed in the formal expression.

Assertions are based on the first-order logic which allows to automatically generate new assertions through the use of a reasoning system by applying several deduction rules to an initial assertion set. The assertions are applied at several points of the control flow allowing to verify different

properties of models and model transformations at different stages of the transformation and to verify the modifications performed by the transformation rules at runtime.

ANALYSIS

This approach proposes a framework to perform the verification of properties of the artefacts used in the transformation process, as well as the assessment of all transformation steps in execution time. This verification is performed through the use of assertions. Assertions provide a platform independent tool to perform the verification, allowing the implementation of the framework in any model transformation framework.

This approach resorts to a reasoning system to produce new assertions from an initial set of assertions and allow the manual introduction of assertions in order to have a contribution from the knowledge of experts. This feature gives flexibility to the system, however the reasoning system has efficiency problems as it may take much time to perform the deduction of new assertions when there are many assertions and deduction rules in it.

Regarding to the results produced by this approach they are represented by boolean values since the results depend on the proprieties verified by assertions.

Approach 4: Model Transformation Analysis using Alloy

In (Anastasakis, Bordbar, & Küster, 2007) is proposed an approach that resorts to Alloy (Jackson, 2006) to perform the formal analysis of model transformations. Alloy consists in a textual and declarative modelling language based on first-order relational logic. Alloy uses a tool called Alloy Analyzer [1] which supports the automated analysis of the models defined in Alloy.

The Alloy Analyzer provides two functionalities: simulation and checking using assertions. The simulation functionality produces a random instance conform to a model. The successful generation of the instance guarantees the consistency of the model. The use of assertions enables to define constraints about the properties of a model and are the basis to the model and transformation verification.

The approach proposed is composed by two steps, as depicted in Figure 2.6. The first step consists in the representation of model-driven artefacts, both model and model transformations, using the Alloy language, allowing to the Alloy Analyzer to use these artefacts which consists in the second step. In this step the Alloy Analyzer can be used to simulate the transformation. To perform this, the Alloy Analyzer generates an instance of the source that conforms to the source metamodel and an instance of the mapping between elements. The execution of the transformation generates a target model conform to the target metamodel. During the simulation process the assertions can be used to check the properties of the source and target models.

[1] Available at `http://alloy.mit.edu/alloy/`

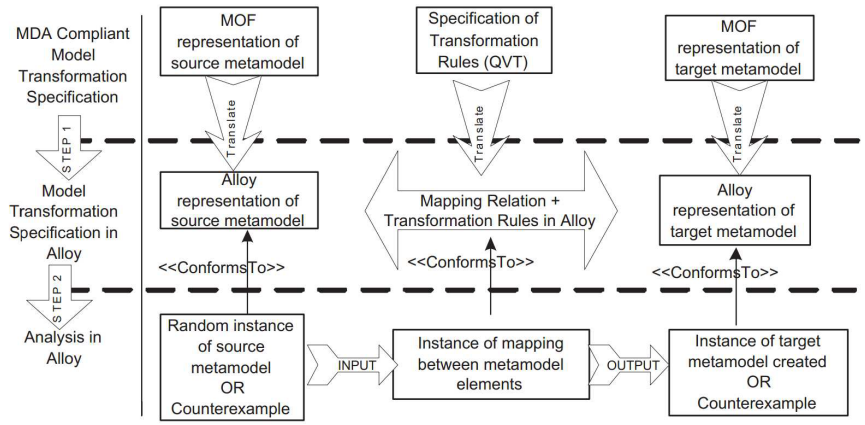

Figure 2.6: Execution of a model transformation using the Alloy Analyzer (Jackson, 2006)

ANALYSIS

In order to allow the measurement of the source and target metamodels need to be defined using Alloy as the Alloy Analyzer can only interpret this language. For the same reason the transformation rules of the model transformation must also de described using Alloy. This requirement can become a complex task since it is necessary of perform mapping between all the modelling languages and transformation languages the will be used to Alloy.

To perform the measurement, this approach generates the source model and the model transformation to be used in a simulation. With this methodology it is possible to verify the consistency of the metamodels and transformations rules described in Alloy. As the instances generated are random the measurement is not associated to a specific source model.

The measurement results produced by this approach are related with the consistence of the metamodels described in Alloy. This approach also supports the use of assertions to verify properties of the models, which produces boolean results.

2.1.2 Synthesis

Presented all the identified measurement approaches is now the time to highlight how these approaches address each one of the characteristics of the problem: Lack of Semantic Definitions, Heterogeneity of Interoperability Specifications, Comparison between Different Pairs of Data Formats.

Regarding to the first characteristic, Approaches 1 and 2 define metrics to be used as basis for the measurement. Approach 1 defines specific quality attributes related with the development and maintenance of model transformations which focus in the measurement to assess structural features of the model transformation artefact. Among these attributes there is the completeness attribute which can be used to define relations between data elements of the source and target

models. Approach 2 proposes the introduction of metrics in the source, target, and model transformation metamodels without specifying an application scenario, which results in the lack of specification of the measurement level.

Approaches 3 and 4 focus in the verification of properties of the model used in the transformation, including properties of the model transformation artefact, and in the verification of the modifications performed by each transformation rule. In both approaches, is not made a specification of which properties or modifications should be measured, and therefore the measurement level is not defined.

In relation to the second characteristic, Approach 1 assesses the quality metrics through the extraction of specific parameters from the model transformation artefact, which corresponds to a direct study of the interoperability specification. Approach 2 executes the model transformation to generate the metrics in the target model, proceeding then to the comparison between the metrics of the source and target models. This procedure allows to measure the model transformation without the need of study the model transformation artefact as the measurement can be performed by comparing the metrics before and after the transformation, which correspond to a indirect study of the interoperability specification.

Approaches 3 and 4 can verify the properties of the models and model transformations, as well as the actions performed by the transformations rules through the use of assertions, before, after, and during the execution of the model transformation. Approach 4 differs from Approach 3 by building instances of the source model and the model transformation to be used in the transformation. By generating these instances, this approach verifies the consistence of the metamodels, however to perform this verification the metamodels need to be expressed using the Alloy language which can prove to be a complex task due to the possible heterogeneity of the metamodels. The verification of the model transformation is performed, rule by rule, in execution time, which implies the direct manipulation and a deep understanding of the interoperability specification.

With respect to the third characteristic, Approach 1 produces quality metrics and attributes which are usually numbers. While quality metrics can only be used to compare model transformations implemented in the same technology and for the same purpose, quality attributes can be used to compare any model transformation since they are not technology dependent and assume values in a well-defined range of values (1-7). On the other hand, Approach 2 presents the comparison between the metrics of the source and the target models. This approach does not specifies how this comparison is expressed so no conclusion related with the comparability of the results can be made.

Approaches 3 and 4 produce boolean values as measurement results since assertions can only perform verifications, returning True or False values depending if the verification is valid of not. Using these results, the only conclusion that can be achieved is if the model transformation passed a certain verification or not. This conclusion can only be used to used to compare the number of verifications passed by each model transformation.

The most relevant conclusions of this analysis are summarised in Table 2.1.

Table 2.1: Approaches synthesis

	Semantic Relations	Heterogeneity of Interoperability Specifications	Comparison between Different Pairs of Data Formats
Approach 1	Defines the quality attribute completeness which can represent a relation between the data elements of the source and target model	Uses the model transformation artefact to determine the quality metrics	Using the quality attributes as they can only assume values in a well-defined range of values and are independent of the technology used to define the model transformation
Approach 2	Does not define the metrics to be used, the measurement level depends on the metrics selected by the analyst	Performs the comparison between the metrics in the source and target models to perform the measurement	Does not define how the results of the comparison are expressed
Approach 3	Does not define the properties or the modifications performed by the transformation rules to be verified	Performs the verification of each transformation rule in execution time, directly manipulating the model transformatio	The analysis of the verifications passed by each model transformation can be used as a mean of comparison
Approach 4	Does not define the properties or the modifications to be verified by the assertions	Requires the mapping of the transformation rules to Alloy	Uses the results of the assertions to perform comparisons

2.2 Advancement

According to the scientific method, the construction of a solution to a question should use the knowledge from past experiences and experiments as support (Schafersman, 1997). This approach allows the evolution of science and of the technology through the transmission of the previously acquired knowledge. Therefore, the analysis of the four measurement approaches can be used to bring experiences and ideas that will reflect in characteristics and/or behaviours of the solution proposed in this work. In order to determine how can each measurement approach contribute to the solution, the analysis of each approach against the characteristics of the problem, summarised in Table 2.1, is taken into consideration resulting in the following conclusions.

Regarding to the Lack of Semantic Definitions only Approach 1 defines an attribute that can be used instead of semantic relations: the Completeness. The other approaches do not define any metric or property to be used as support for the measurement, therefore none metric or property is define to measure the information exchange. Thus, the concept of Completeness, namely the relation between data elements will be addressed by the hypothesis, focusing the measurement in the data exchange. However, this attribute cannot be directly assessed from the model transformation artefact as performed by Approach 1 since it is limited by the heterogeneity of these artefacts.

Approach 2 performs the measurement through the comparison of the metrics of the source model with the metrics of the target model, being the only one that performs the measurement without looking onto the transformation rules defined in the model transformation. However, this approach requires the modification of the the source and target models, which is not possible in a PnI scenario. Approaches 3 and 4 propose the assessment of properties of the source and target models, but do not specify any property to be measured. However, for doing that they resort to the use of assertions whose logic can vary from model to model, being dependent from the source and target models used, and thus not recommended. Therefore, the hypothesis will measure the data exchange without looking to the model transformation, nor changing the source or target models. To achieve this, the hypothesis will compare properties the source data with the target data produced by the execution of the model transformation.

In relation to the measurement results, Approach 1 produces quality attributes, technological independent, that are represented by a number within a well-defined range of values. Therefore, the hypothesis will produce as result number within a well-defined range of values, in order to allow to comparison of data exchanges between the same pair of models and between data exchange defined between any models.

Approach 4 presents an interesting feature as it is capable of generate an instance of the source model to perform the simulation of the model transformation. This feature enables the execution of a generic measurement as it does not depends of a specific instance of the source model.

3

Data Transfer Measurement

3.1 Concept

The usage scenario of the measurement approach is composed by data formats and interoperability specifications, where a data format represents the data of a device and interoperability specifications provide the information on how to transfer data between the data formats. This scenario is depicted in Figure 3.1. In this scenario there is a measurement method that uses a set of data formats called Source Data Formats, a set of interoperability specifications, and a set of data formats called Target Data Formats. This measurement method performs the evaluation of each interoperability specification.

The Source Data Formats corresponds to the set of data formats where, for each data format, exists at least one interoperability specification that knows how to transfer data between that data format and another one. On the other hand, the Target Data Formats is the set composed by all the data formats where, for each data format, there is at least one interoperability specification that knows how to transfer the data of other data format to that data format. Notice that these two sets can have data formats in common as well as disparate data formats. The set of interoperability specifications is composed by all the interoperability specifications with information of how to transfer data from the data formats in the Source Data Formats set to data formats in the Target Data Formats set.

The hypothesis proposed in this work is a measurement method which aims to the classification of each interoperability specification according to the data transfer that it provides. This classification supports the comparison between interoperability specifications. This comparison of

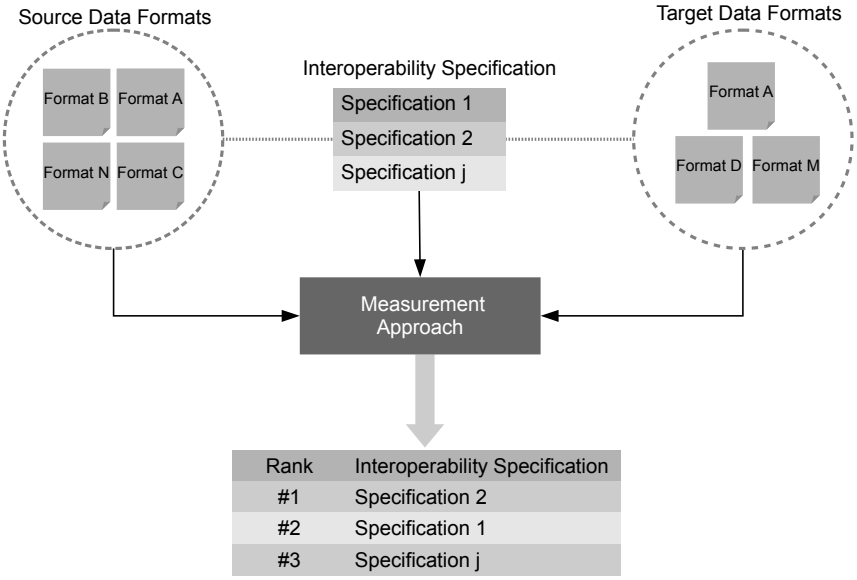

Figure 3.1: Concept of the Measurement Approach

classifications can be either for interoperability specifications designed for the same pair of data formats or for all interoperability specifications in the set, disregarding the pair of data formats involved. The first comparison can be used, for example, to infer which interoperability specification provides the best data transfer between a certain pair of data formats while the second comparison can be used to determine the interoperability specification that provides the best, or the worst, data transfer in the environment.

3.2 Method

The Measurement Method proposed in this work is illustrated in Figure 3.2. As inputs, the measurement method has a Source Model and a Target Model. The measurement method uses the data transfer to evaluate the mapping of elements. The data transfer is assessed by relating the coverage of source elements with the coverage of target elements.

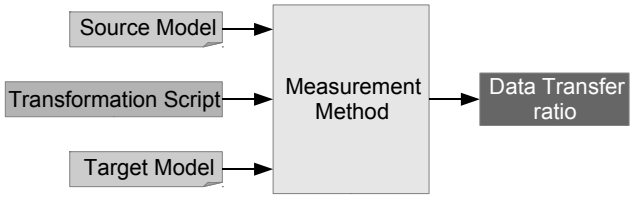

Figure 3.2: Representation of the Measurement Method

The coverage term is understood as the relation between the number of elements defined in the *model* and the number of elements used in the element mapping to produce a *data*, where the term *model* corresponds to the artefact that provides the description of the data elements and the relations between them. A *model* does not represents data as it only defines the structures that will support and represent it, i.e. corresponds the data format. The term *data* corresponds to the

artefact that represents data using the elements described in a *model*. Therefore a *data* must always be conform to a *model*.

This approach leads to the need to perform the data transfer, i.e. to execute the element mapping in order to perform the measurement, since it is required to determine the used elements. Therefore a Transformation Script is needed to provide the measurement method with a reference to an executable mapping of elements to be evaluated.

The output of the measurement method must discern each element mapping and enable the comparison based on the data transfer described in the element mapping. These requirements make Data Transfer ratio as a suitable name for this output, as it evaluates the data transfer based on the ratio of data elements used by the element mapping. There could be some cases where, due to semantic details, the data transfer of a mapping of elements cannot be improved even if the Data Transfer ratio has not achieve the maximum value. Cases like this can occur because the measurement method is only concerned with the study of the data elements, disregarding the semantics related to these elements.

The calculus of the Data Transfer ratio is performed in order to classify the data transfer performed by a transformation according to a ruler. This scale must allow the comparison between several Data Transfer ratios, each one produced by different transformation scripts with or without relations among them. The ruler defined for this purpose assumes values in a range between 0 and 1, as depicted in Figure 3.3.

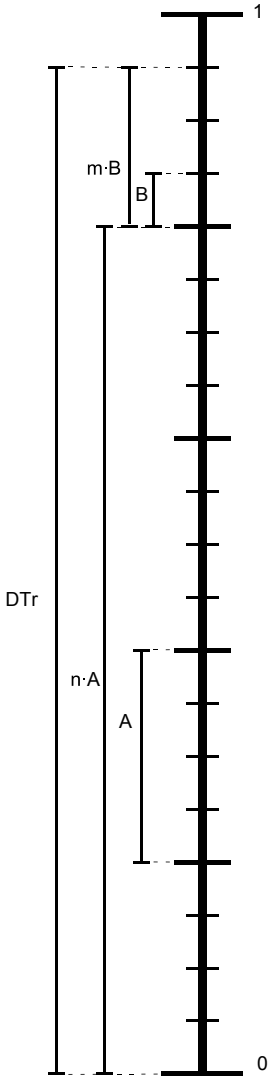

Figure 3.3: Measurement ruler

The measurement method is based on the data elements defined in a data format, i.e. the elements defined to store data values. The calculus of the Data Transfer ratio resorts to the number of data elements defined in the source and target *models*, and to the number of data elements used by the element mapping to define the data transfer. These elements are represented in the source and target *data*. To this number of elements is assigned the term *Count*. Thus, the measurement method resorts to four parameters to compute the Data Transfer ratio:

1. Source Model Count (SMC) - represents the number of data elements defined in the source *model*;
2. Source Data Count (SDC) - represents the number of data elements in the source *data* that are used by the element mapping;
3. Target Model Count (TMC) - represents the number of data elements defined in the target *model*;

4. Target Data Count (TDC) - represents the number of data elements in the target *data* that are produced by the execution of the element mapping.

As depicted in Figure 3.3, the reasoning to calculate the Data Transfer ratio (DTr) uses the concepts of the distances represented in the to defined the Data Transfer ratio. The first distance to be defined is represented by the letter "A". This distance corresponds to how much the Data Transfer ratio improves with the generation of one more data element in the target *data*. This distance defines the step of the main scale of the ruler, corresponding to the contribution of the target data elements.

It is imposed that the contribution of the utilisation of source data elements to the Data Transfer ratio can only be, at most, equal to the contribution resultant of the addition of one more data element to the target *data*. This imposition is made due to the consideration that the number of data elements produced by a transformations script is more important then the number of source elements used, based on the fact that the first ones are used by the receiver system to use the data exchanged. Therefore, the number of steps that compose the main scale of the ruler is determined by the number of elements defined in the target *model* plus one that represents the maximum contribution from the usage of the source data elements. Thus distance "A" is determined through the use of Equation 3.1.

$$A = \frac{1}{TMC + 1} \tag{3.1}$$

Other distance that needs to be defined is represented by the letter "B" in Figure 3.3. This distance determines the improvement of the Data Transfer ratio, caused by the utilisation of the one more source element by the transformation script. This distance defines the step of the secondary scale, corresponding to the contribution of the source data elements. This scale consists in a subdivision of the main scale allowing an increase of the measurement resolution. Using these considerations, this distance is determined through the use of Equation 3.2.

$$B = A \times \frac{1}{SMC} \tag{3.2}$$

The contribution of the number of data elements generated in the target *data* is represented in Figure 3.3 by the expression "$n \cdot A$". Empirically, this contribution is calculated by the multiplication of A by this number of elements, which is represented by the Target Data Count. Thus, Equation 3.3 is used to calculate this contribution.

$$n \cdot A = A \times TDC \tag{3.3}$$

The contribution of the number of elements from source *data*, used by the transformation script , which is represented in Figure 3.3 by the expression "$m \cdot B$", is calculated by multiplying B by the

number of source elements used, i.e. using the Source Data Count. This contribution is calculated using Equation 3.4.

$$m \cdot B = SDC \times B \tag{3.4}$$

The output of the measurement method, the Data Transfer ratio, DTr, must be able to differentiate several transformation scripts in accordance to the data transfer performed by each one of them. To do so, the contribution of the target data elements produced and the source data elements used must be used, being added to each other. This reasoning leads to Equation 3.5.

$$\begin{aligned} DTr &= n \cdot A + m \cdot B \\ &= A \times TDC + SDC \times B \quad \text{, replacing } B \\ &= A \times TDC + SDC \times A \times \frac{1}{SMC} \\ &= A \times \left(TDC + \frac{SDC}{SMC} \right) \quad \text{, replacing } A \\ &= \frac{1}{TMC+1} \times \left(TDC + \frac{SDC}{SMC} \right) \end{aligned} \tag{3.5}$$

Regarding to the assessment of these parameters, the Source Model Count and the Target Model Count have similar approaches to determine its values. While the Source Model Count is determined by counting the number of the data elements defined in the source *model*, the Target Model Count is determined by counting the number of the data elements defined in the target *model*. The Target Data Count, in its turn, is determined by counting the number of data elements present in the target *data*, generated by the execution of the transformation script. The determination of the Source Data Count can be determined by the analysis of the artefact that describes the element mapping. However, this approach is not an option since this artefact can be defined using different technologies, each one with different syntax and characteristics.

The approach proposed in this work to determine the Source Data Count resorts to the execution of the transformation script multiple times in order to determine which data elements of the source *data* produce changes in the target *data* when changed. To execute this procedure, there is the need to use a structure to control the execution of this loop and define the inputs used in each iteration.

To perform this control, Table 3.1 is proposed. This table is divided in two sub-tables: one to represent all the data elements of the source *model* (Sub-table 3.1a), and one to represent all the data elements in the target *model* (Sub-table 3.1b). The columns of sub-table 3.1a ($I_1 \ldots I_n$) correspond to all the data elements defined in the source *model*. Similarly, the columns of Sub-table 3.1b ($O_1 \ldots O_m$) represent the data elements of the target *data*.

In Sub-table 3.1a is performed the control of the inputs, where are assigned data values to each

Table 3.1: Control table

(a) Source elements sub-table

I_1	I_2	...	I_n

(b) Target elements sub-table

O_1	O_2	...	O_m

one of them. These values will be used in the generation of the multiple source *data* that will support the calculation of how many inputs are used by the transformation script to produce the target *data*. Each row corresponds to a test, i.e. to the execution of one transformation. In this sub-table must be represented the reference test and the tests used to represent the changes in each one of the inputs. Therefore in a table with n source data elements there will be $n+1$ rows.

In Sub-table 3.1b are represented the results of each test, where the row number of the result correspond to the test in the same row number in Sub-table 3.1a. The target data elements that are defined in the target *model* but are not present in the target *data* produced in the testing are filled with a specific character that represents this absence. The other target data elements are filled with the data values obtained.

Consider the example depicted in Figure 3.4 corresponding to an element mapping. In this example, the source *model* has four data elements (S_1, S_2, S_3, and S_4) and the target *model* has five data elements (T_1, T_2, T_3, T_4, and T_5). These data elements are mapped by transformations rules f, g, h and m.

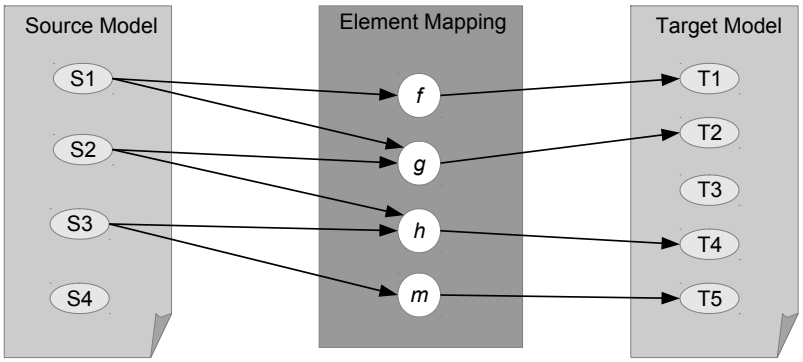

Figure 3.4: Element mapping of the example

In Table 3.2 is represented the example represented in Figure 3.4. In order to simplify the example it is considered that all the elements are strings. It is also considered that the symbol that represents the absence of a data element in the target *data* is "X".

As the source *model* has four data elements, Table 3.2 has five rows. In the first row of Sub-table 3.2a is presented the input for the reference test where, in this case, all the elements will assume the value "A". The four next rows correspond to the testing of each one of the source elements where, for each row, is replaced the default value "A" by the value "B" in the element to be tested. As the target *model* has five data elements, Sub-table 3.2b has five columns, however for one column, corresponding to the target element T_3, never is generated a data value as it is filled

Table 3.2: An example of control table

(a) Source elements sub-table

S_1	S_2	S_3	S_4
A	A	A	A
B	A	A	A
A	B	A	A
A	A	B	A
A	A	A	B

(b) Target elements sub-table

T_1 (S_1)	T_2 (S_1+S_2)	T_3	T_4 (S_2+S_3)	T_5 (S_3)
A	AA	X	AA	A
B	BA	X	AA	A
A	AB	X	BA	A
A	AA	X	AB	B
A	AA	X	AA	A

with the symbol "X" in all the performed tests.

Comparing both sub-tables one can verify that changes in the element S_1 produce changes in elements T_1 and T_2, changes in S_2 cause changes in T_2 and T_4, and changes in S_3 produce changes in T_4 and T_5. Changes in element S_4 do not produce any change in the target *data* which results in the appearance of a result equal to the result corresponding to the reference test.

The Source Model Count is determined by the number of columns in Sub-table 3.2a, therefore $SMC = 4$. Analogous reasoning is applied to determine the Target Model Count, which corresponds to the number of columns in Sub-table 3.2b, resulting in $TMC = 5$. The Source Data Count corresponds to the number of rows in Sub-table 3.2b different of the row corresponding to the reference test row. Thus, for this example, $SDC = 3$. At last, the Target Data Count is determined by the number of columns in Sub-table 3.2b where, in at least one test, there was obtained a result, so $TDC = 4$. Applying these values in Equation 3.5 results in:

$$DTr = \frac{1}{5+1} \times \left(4 + \frac{3}{4}\right) = 0.79$$

3.3 Specification

The goal of the measurement method is to assign a Data Transfer ratio to the element mapping executed by transformation script. To produce this Data Transfer ratio four parameters (the Source Model Count, the Source Data Count, the Target Model Count, and the Target Data Count) need to be determined. These parameters are determined through the analysis of the control table. And to produce a control table, the measurement method must know the structure of the source and the target *models*, as well as, be able to execute the transformation script. This reasoning results in a measurement method that consists in the composition of the following components:

Model Walker walks the *model* received as input, identifying the elements defined in it and their hierarchy, producing a data structure that represents the *model*.

Transformation Analyser performs the execution of the transformation script in order to produce the control table.

Table Counter analyses the control table produced by the Transformation Analyser component

in order to determine the Source Model and Data Counts as well as the Target Model and Data Counts.

Evaluator is responsible for the computation of the data transfer evaluation. To perform this evaluation the Evaluator component resorts to an algorithm to produce a quantified output. The output of this component is also the output of the measurement method (the Data Transfer ratio).

These components and the interconnection between them are depicted in Figure 3.5. Notice that this figure can be seen as a more detailed representation of Figure 3.2.

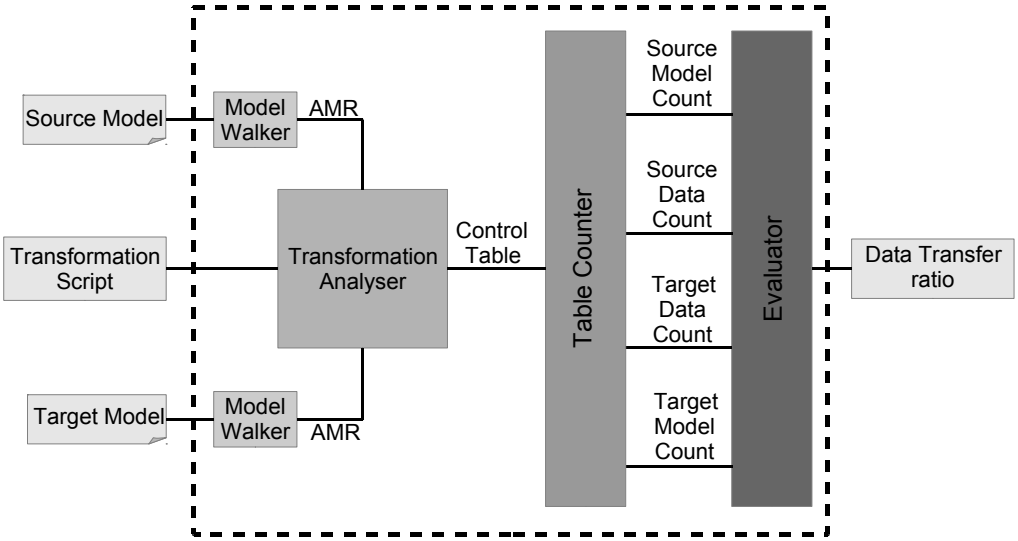

Figure 3.5: Representation of the measurement method

3.3.1 Model Walker

The purpose of the Model Walker component is to walk through the *model* in order to identify the type and constraints associated to each element as well as the relations defined among them. To achieve this goal the Model Walker uses an algorithm to build a data structure that represents the structure of the *model*.

The data structure chosen for this task is an Abstract Syntax Tree (AST). The AST is chosen due to its capacity to represent, in an abstract form, the structure of the *model*, standing out as a flexible tool capable of representing the elements defined in a model. Nevertheless, a AST does not provide means to handle situations of repetition of nodes which is a need that must be attended.

The algorithm to build the AST starts with the identification of each element defined in the model. For each element, the type and constraints associated with it are also identified. After the identification of an element it is introduced into the AST.

In order to address the problem of the handling of repetitions of elements, a data structure is defined to be used along with the AST. This structure consists in a hash table where each entry has

the structure represented in Figure 3.6. This structure is composed by two fields. The first field is responsible for the representation of the reference to the element. This field is used as key in the hash table, therefore it must be unique. The second field correspond to a structure that holds the information about the type of the element (structural or data), its constraints, and the references to the elements that reference this element.

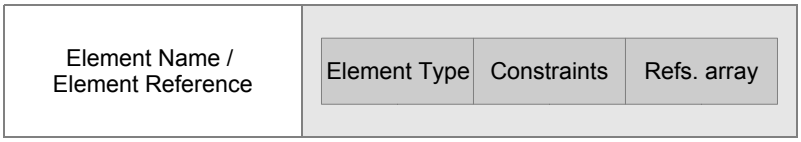

Figure 3.6: Representation of the structure of a referenced element

The algorithm that performs the walk through of the *model* element by element, identifying each element and inserting it into the AST and into the hash table is represented by a flowchart in Figure 3.7. The insertion into the hash table only occurs if the element is not already in the hash table. Every time that the algorithm that performs the walk through of the *model* founds an element that references another element it looks into the hash table to see if the referenced element is already represented. If that element is already represented, then the algorithm adds a reference of the found element to the array of the second field. Otherwise, the algorithm adds a structure to represent the referenced element into the hash table, filling the first field with the reference to the referenced element and the second field with its type, constraints associated, and the reference to the found element.

This algorithm generates a structure called Abstract Model Representation (AMR), which consists in the aggregation of the AST and the hash table. While the AST provides an abstract view of the hierarchy of the elements defined in the *model*, the hash table identifies the unique elements defined in the *model* thus, the hash table can represent the number of elements defined in the *model* without counting the element repetitions. This aggregation of structures can be used to provide an abstract representation of a *model* as this representation is independent of the technology used to describe the model.

3.3.2 Transformation Analyser

The objective of the Transformation Analyser component is to produce the control table. To accomplish this goal there is the need to vary each one of the inputs in order to determine the inputs that can produce changes in the target *data* . This procedure can imply multiple executions of the transformation script since the test of the inputs must be performed individually and each test implies one execution of the transformation script. To allow this behaviour the Transformation Analyser is decomposed in the following functional blocks:

- Control Unit is responsible for controlling the loop of executions of the transformation script in order to produce the control table. This functional block is also responsible for the generation of the data values that will be used in the construction of the source *data* used in the

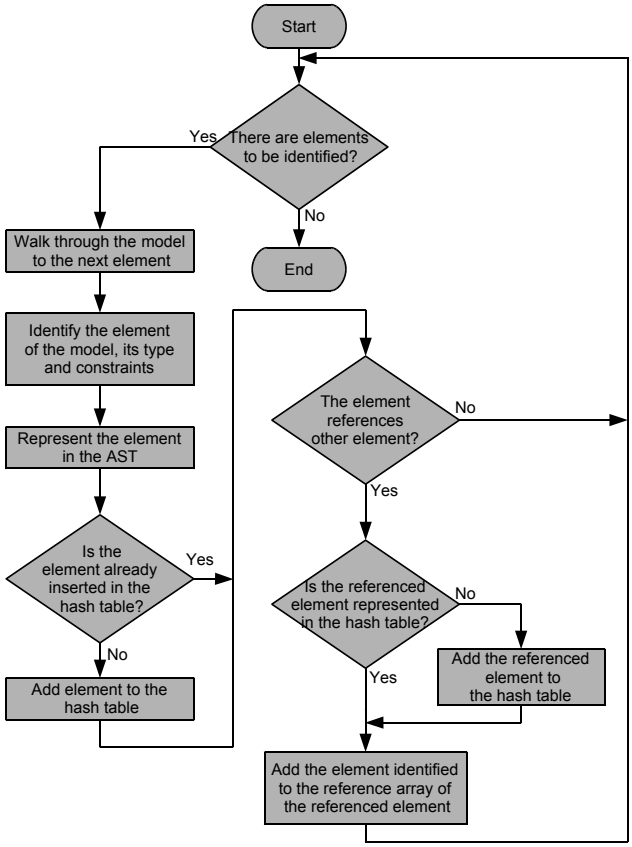

Figure 3.7: Flowchart of the process of construction of an Abstract Model Representation

execution.

- Data Generator builds a source *data* according to the data values generated by the Control Unit and the AMR produced by the Model Walker component that analyses the source *model*.
- Transformation Executor executes a data transformation generating a *data* which is conform to the target *model*, using as input a *data* that conforms to the source *model* and the transformation script that provides a set of actions to execute the element mapping to be measured.
- Results Extractor extracts the data values of the target *data* generated by the Transformation Executor functional block. This functional block uses the AMR, produced by the Model Walker component that analyses the target *model*, to walk in the target *data* artefact.

Figure 3.8 illustrates how these functional block are connected and arranged inside the Transformation Analyser component. The Control Unit generates the data values that will be used by the Data Generator to produce the source *data*. After the generation of the *data*, the transformation script is executed by the Transformation Executor, producing the target *data*. The data values in that *data* are then extracted by the Results Extractor and delivered to the Control Unit to be stored in the control table.

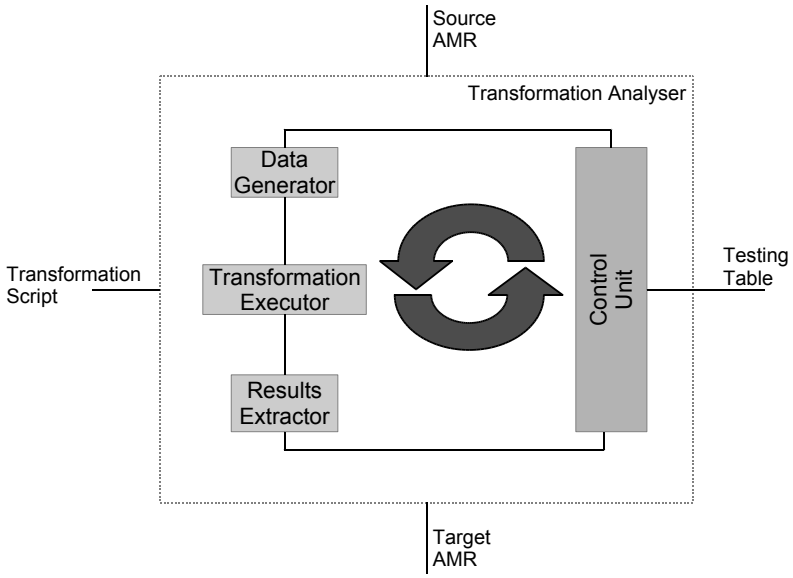

Figure 3.8: Diagram of the Transformation Analyser component

Control Unit

The Control Unit functional block controls the execution flow of Transformation Analyser component, producing the control table. In a first stage, this functional block constructs the structure of the control table using the hash tables inside the source AMR and the target AMR. The columns of the source elements sub-table are defined using the data elements, i.e. the elements which type is a data type, represented in the source AMR. In a similar way is defined the columns of the target elements sub-table, using the target AMR.

After the definition of the structure of the control table, the data values that will be used to generate the source *data* must be generated. To perform this procedure there is the need to identify the data types of each source element. This information is present in the hash table inside the source *model* AMR. After the identification of the data types, it will be generated data values for each primitive data type such as is the case of strings, integers, dates or numbers. In the generation of these data values can be used, for example, specific algorithms that automatically generates random values for these data types, or data values previously defined and stored in some structure. Each row of the source elements sub-table is then represented in a hash table, where the reference of the element is the key the data values generated are placed in the values field. This structure is used to provide the Data Generator functional block with the data values to be used to generate the source *data* that will be transformed.

When this functional block receives the data values from the Results Extractor functional block, they come in a hash table where the reference of the element is the key the data values are in the values field. This hash table is iterated and all the data values are placed in the corresponding cell of the target elements sub-table. When all the rows of this sub-table are filled, the control table is complete, as well as the job of this functional block.

Data Generator

This functional block has the goal of generate a source *data* to be transformed by the Transformation Executor functional block. In order to accomplish this objective, the Data Generator functional block uses the data values generated by the Table Builder functional block and the AMR, corresponding to the source *model*, that is generated by the Model Walker component.

The Data Generator functional block starts the construction of the data using the information provided by the AST inside the AMR being this information used to determine the structure of the *data*. This information is extracted from the AST by walking through it. For each element walked in the AST, the hash table inside the AMR is accessed in order to determine the type of the walked element, as well as, the constraints associated to it. If the type of the element is a data type, then the hash table provided by the Table Builder functional block is queried in order determine the data value the used to generate the *data*.

When the AST is completely walked and all the elements defined in it are instanced, the *data* generation starts the definition of the relations between the elements. This step is assisted by the hash table inside the AMR. This hash table is iterated and when is found an element that does not have an empty array of references then, an identifier of this element is inserted in all the elements present in the array of references. This process of generation a *data* is described as a flowchart in Figure 3.9.

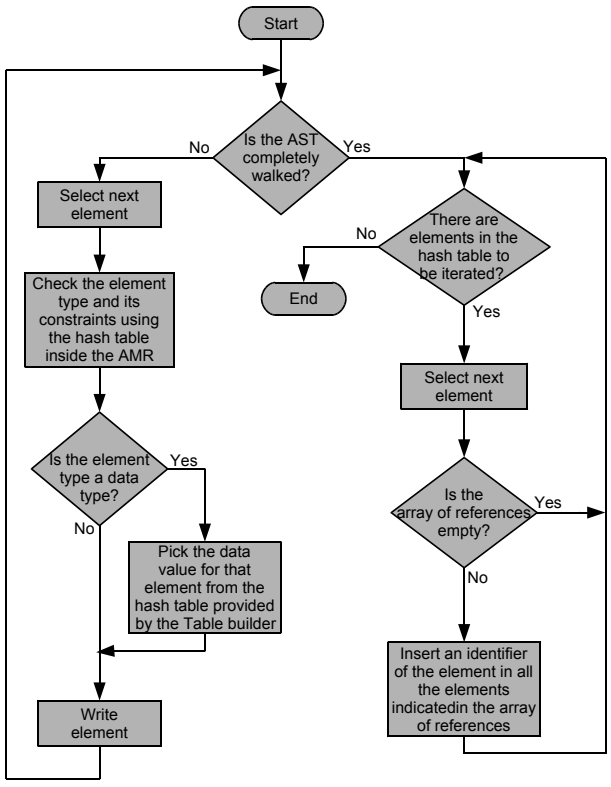

Figure 3.9: Flowchart of the process of generation a *data*

Transformation Executor

This functional block is responsible for the execution of the transformation from a source *data* into a target *data*. To perform this action this block executes a transformation script. This transformation script holds information about the set of transformation actions that needs to be executed. This information is related with how should the transformation steps be executed. This block must provide a support for the script in order to transfer data, and execute the transformation script. The inputs and output provided by this script support are represented in Figure 3.10.

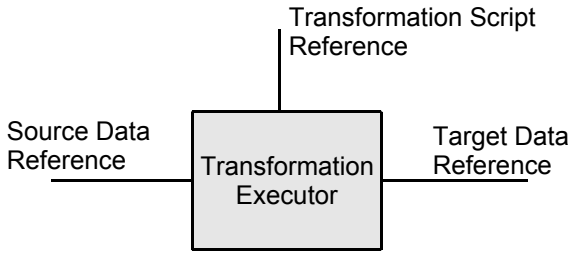

Figure 3.10: Representation of the Transformation Executor

The Source Data Reference represents the *data* which is conform to the source *model*. This reference provides the data elements to be transformed. The Transformation Script Reference provides the location and a handler to the transformation script. This reference supplies all the transformation steps that need to be executed in order to perform the transformation. These transformation steps are executed by the Transformation Script Executor. The Target Data Reference is an output which provides the location of the *data* produced by the transformation.

Results Extractor

This functional block has the function of extract the data values from the data elements produced by the transformation, i.e. the extraction of the data values from the data elements of the target *data*. This extraction requires the walk through of the target *data*. This activity is supported by the target AMR as this structure has the representation of the structure of the target *model*. The data type of the element must be identified, using the target AMR, in order correctly manipulate the data value. The extracted values are then grouped in a hash table where a reference to the data element is used as key and the data value is stored in the value field.

3.3.3 Table Counter

This functional block processes the table produced by the Table Builder in order to determine each of the counts needed to perform the evaluation of transformation script: Source Model Count, Source Data Count, Target Model Count, and Target Data Count. Therefore, these four counts are the outputs of this functional block. The determination of each count requires the process of different parameters of the input table.

The Source Model Count represents the number of data elements defined in the source *model*. Therefore, in order to determine this count, the number of columns in the sub-table that represents the source data elements must be counted. This procedure is possible due to the fact the this sub-table is built using the hash table contained in the AMR of the source *model*, therefore this sub-table contains all the defined data elements and each one only appear once.

Regarding to the Source Data Count, which represents the number of elements of the source *model* where its data values are used by the transformation script. This count can be determined through the comparison of rows in the sub-table that represents the target elements. The comparison is performed row by row, element by element. All rows must be compared with the reference row. For each row the value of each data element is compared with the value of the corresponding element in the reference row. Each time that the values are different, the counter is incremented and another row, if any, is selected to be compared. This algorithm is depicted through a flowchart in Figure 3.11.

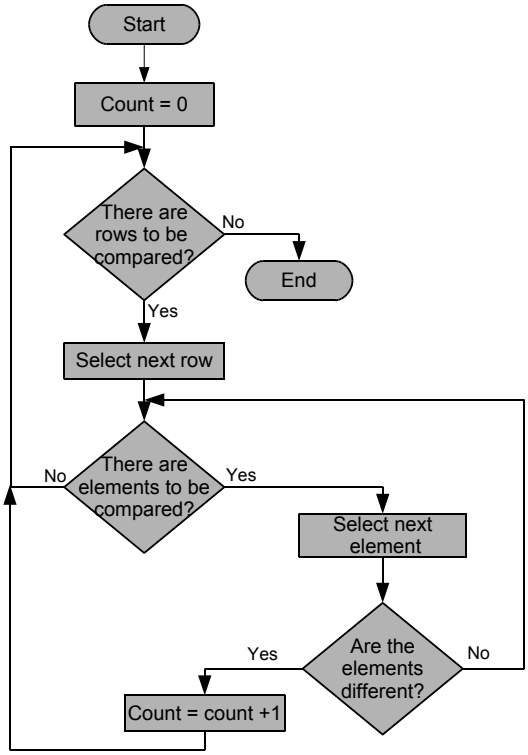

Figure 3.11: Flowchart of the process of counting the Source Data Count

The Target Model Count, which represents the number of data elements defined in the target model, is determined by the total number of columns in the sub-table that represents the target elements must be counted. This procedure is similar to the one responsible for the counting of the Source Model Count, and is only possible due to the fact that this sub-table is built using the hash table contained in the AMR of the target *model*.

With respect to the Target Data Count, this count corresponds to the number of target data elements produced by the transformation script. This count is indicated by target elements where

were obtained at least one data value in one of the transformation script tests, therefore the sub-table the represents the target elements must be used. The counting process walks through each column verifying the output for the corresponding row. When a data value is found the counter is incremented and the next column, if any, is selected to be verified. This algorithm is depicted in Figure 3.12.

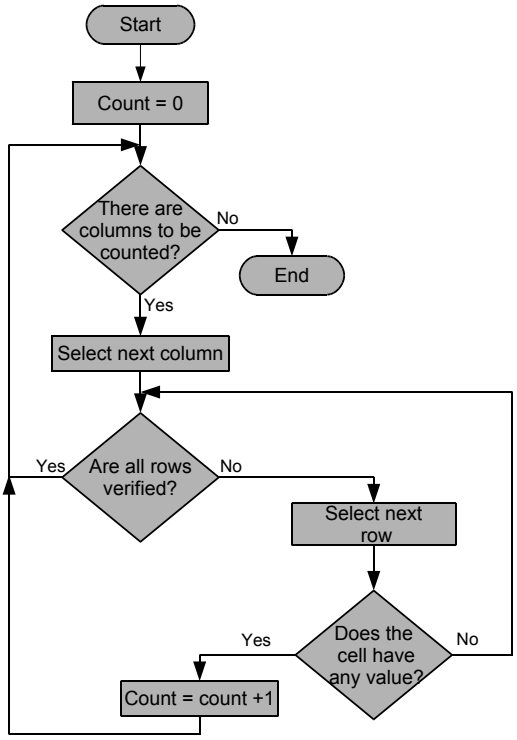

Figure 3.12: Flowchart of the process of counting the Target Data Count

3.3.4 Evaluator

The Evaluator component is responsible for the computation of several inputs in order to evaluate a given transformation script. These inputs correspond to the parameters generated by the Table Counter component, i.e. the Source Model Count, the Source Data Count, the Target Model Count, and the Target Data Count. The calculation performed in this component consists in the computation of Equation 3.6.

$$DTr = \frac{1}{TMC+1} \times \left(TDC + \frac{SDC}{SMC} \right) \tag{3.6}$$

The complete diagram of the measurement method, with the detailed Transformation Analyser component, is presented in Figure 3.13.

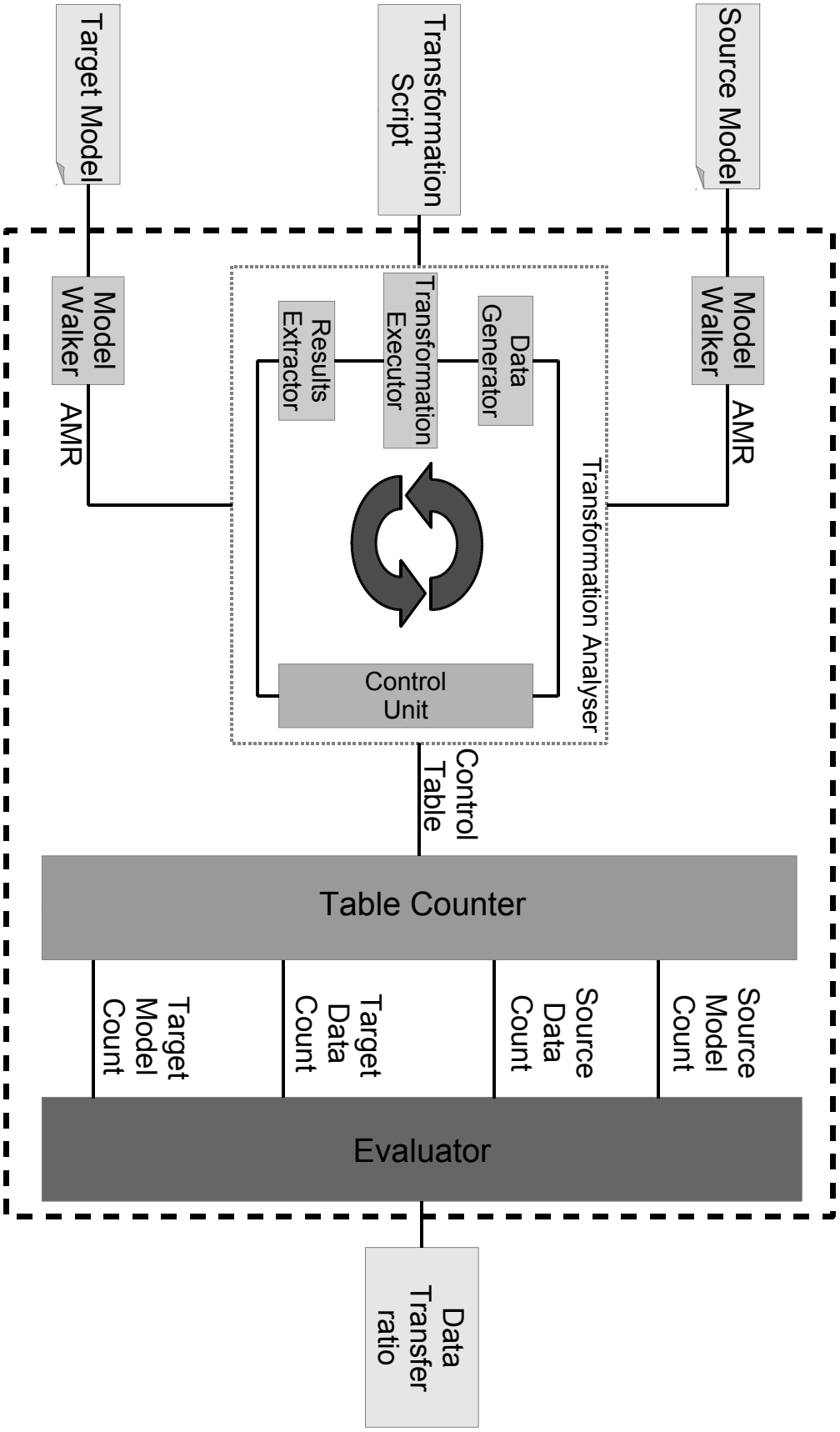

Figure 3.13: Complete diagram of the Measurement Method

4

Testing and Validation

4.1 Testing Methodology

In this chapter is performed the validation of the hypothesis. In order to validate the hypothesis, it has to be subjected to specific tests. These test have as objective the gathering of results that allows the evaluation of the hypothesis facing the characteristics of the problem. All the results gathered are produced in a controlled environment in order to control all the results of the experiment and ensures that the testing can be reproduced. However, these tests cannot ensure a complete correctness of the hypothesis since the testing period has a limited time duration making impossible an exhaustive testing. Due to this fact, testing cannot guarantee that the solution is error free since it can only show the presence of errors and not their absence (Tretmans, 2001).

In relation to testing methodologies, several methodologies exist to evaluate solutions, determining if the given solution is able to achieve its requirements and specifications. These several methodologies differ in the application domain (Onofre, 2007). The testing methodology chosen in this work is the ISO 9646: “Open Systems Interconnection (OSI) Conformance Testing Methodology and Framework”. This standard aims to the definition of a testing methodology, a framework for specifying test suites, and procedures to be followed during testing. ISO 9646 does not specify testing for a specific protocols. It is due to the fact that this standard provides a generic testing methodology that it was chosen.

The testing process described by ISO 9646 is divided in three different steps, as depicted in Figure 4.1. The first step, the Test definition, consists in the definition of an abstract test suit. These

tests are called abstracts due to the fact that they are defined independently of the implementation. The second step is called Test Implementation and consists in the realisation of a specific implementation that allows the execution of the tests. In this step the abstract tests generated in the previous step are also adapted in order apply them to a specific implementation. The last step consists in the Test execution where the tests are executed and observed, leading to a verdict based on the compliance of hypothesis with the characteristics of the problem.

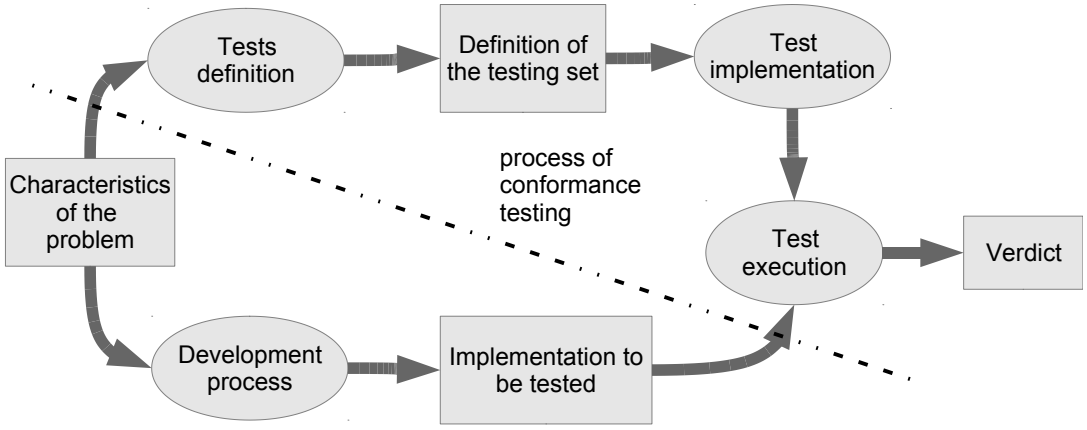

Figure 4.1: Testing process (based on (Tretmans, 2001))

In order to enable the definition of standard and abstracts test suits a well-defined and independent of any implementation test notation needs to be used. The test notation recommended by the standard ISO 9646 is the semi-formal language Tree and Tabular Combined Notation second version (TTCN-2). TTCN-2 follows in the concept of black box model where the internal behaviour of the system is not relevant, being the functionality of the system determined through observation and no reference is made to the internal structure of the program, assessing if the system complies with specification.

The TTCN-2 presents in a tabular form the various parts that define the test, these being the overview about the test, the necessary declarations for implementation, constraints and dynamic part. The overview part contains a table of contents and a description of the test suite. The declarations part declares all messages, variables, timers, data structures and black box interfaces. The constraints part assigns values and creates constraints for inspection of responses from the implementation under test. In the dynamic part, the tests themselves are described, namely its behaviour (Tretmans, 1992). The events that compose the testing behaviour are divided in two types: Actions and Questions. The actions, which are represented with an exclamation mark ('!') at the beginning of the event, define the interactions with the system. The questions, which are represented with a question mark ('?'), are the expected answers from the system.

The verdict can output three different results: **Sucess**, **Fail**, or **Inconclusive**. **Sucess** indicates that the test was executed successfully, **Fail** indicates that the implementation does not conform to the specification, and **Inconclusive** indicates that no evidence of non-conformance was found, but that the test purpose was not achieved.

In Table 4.1 is presented an example of a TTCN-2 based table test. In this example is used an approach based on the TTCN-2 in order to simplify its the abstract test definition. This approach allows the declaration of the inputs needed to the testing and will be used to define all the abstract tests used to validate the hypothesis. The presented test starts with the dial of the phone number of the destination phone. In the next step is verified the establishment of the connection line. If no connection line is established than the verdict is "'Fail"'. Otherwise, if the connection line is established the verdict is "'Sucess"'. However, if there is a busy tone in the connection line the verdict is "'Inconclusive"' as there can be several reasons for this behaviour.

Table 4.1: Simplified example of a TTCN-2 based table test

Test Case		
Test name:	Test the establishment of a Basic Connection	
Purpose:	Check if a phone call can be established	
Inputs:	[I1]: Phone number	
Line number	**Behaviour**	**Verdict**
1	! Dial number [I1]	
2	? Connected line	
3	! Connection Established	SUCCESS
4	! Busy Tone	INCONCLUSIVE
5	? No connection	FAIL

Once defined a abstract test, it is required a structure to represent its execution, i.e. the inputs used, the results obtained and the results expected. The structure chosen is a table named execution table. In Table 4.2 is presented a a execution table for the abstract test defined in Table 4.1. In this execution table are represented the inputs to be used in the testing, the expected results and the results obtained in the testing. Each row of Table 4.2 represents a specific test case. With this approach more than one test case can be represented for each abstract test.

Table 4.2: Test Case example

Test	Input	Result (Line Number)	
	I1: Phone Number	Expected	Actual
1	(+351) 213456789	Success (3)	Success (3)
2	(+351) 213456	Fail (5)	Fail (5)

4.2 Test Definition

The test of the hypothesis consists in testing it, in order to verify if it complies with the characteristics of the problem. In the hypothesis defined in this work, the core of the measurement method is the mathematical expression as it is the component responsible for the calculus of the Data Transfer ratio. As the theoretical concepts in which the mathematical expression is based on are designed to comply to these characteristics, if the behaviour of the mathematical expression corresponds to the behaviour expected from the theoretical concepts, then the mathematical expression complies with the characteristics. To perform this testing the abstract test represented in

Table 4.3 is defined, where the comparison of the behaviours is performed through the analysis of a graph that represents the output of the mathematical expression.

Table 4.3: Abstract test case definition for testing the Data Transfer ratio calculus

Test of the DTr calculus		
Test name:	Test of the Data Transfer ratio calculus	
Purpose:	Test the behaviour of the mathematical expression used to compute the Data Transfer ratio	
Inputs:	[I1]: SMC; [I2]: SDC, [I3]: TMC, [I4]: TDC; [I5]: Expected behaviour	
Line Number	**Behaviour**	**Verdict**
1	! Compute the formula using [I1], [I2], [I3] and [I4]	
2	! Plot corresponding graph	
3	! Analyse the graph and compare the behaviour with [I5]	
4	? Results are conform to [I5]	SUCESS
5	? Results are not conform to [I5]	FAIL

Three sets of execution tests are defined for the abstract test represented in Table 4.3. These sets are defined to allow the verification of the behaviour of the mathematical expression in most of possible scenarios. In the first set is tested the behaviour of the mathematical expression for different interoperability specifications defined between a certain pair of models . The second set is composed by two tests: one where for a certain source model are tested several target models in situations where the target model was fewer, equal, and more data elements then the source model; and one where for a certain target model are tested several source models in situations where the source model was fewer, equal, and more data elements then the target model. The last set corresponds to the testing of different source and target models, being tested in cases where the ratios of source and target data elements are fixed.

4.3 Test Execution

The three sets of tests are implemented and executed using the MATLAB environment. Each implementation generates as result a graph with the DTr values obtained for the inputs used. The obtained graphs must then be analysed in order to verify the behaviour obtained and compare it with the expected behaviour. The MATLAB environment can produce 2D, 3D and 4D graphs, however 4D are complex and are difficult to analyse. This fact results in the choice of use 3D graphs, which limits the number of inputs that can vary in each test to two.

4.3.1 Set 1: Different interoperability specifications for a pair of models

To perform this test it is specified that each source and the target model defines ten data elements each implies that the Source Model Count and the Target Model Count are constant and are equal to 10. As this test must simulate the use of different interoperability specifications the Source and Target Data Counts must vary. Therefore, in this test, each one of these counts will assume values

an range of 1 to 10 in order to simulate all the interoperability specifications possible between that pair of models. This test is presented in Table 4.4.

Table 4.4: Test execution of variation of interoperability specifications between a pair of models

Test	Inputs					Result (Line Number)	
	I1: SMC	I2: SDC	I3: TMC	I4: TDC	I5: Expected behaviour	Expected	Actual
1	10	A	10	B	- Increase of TDC linearly increases DTr - Increase of SDC linearly increases DTr - Increase of TDC produces a greater increase of the DTr than the increase of the SDC	Success (4)	Success (4)

Initial Conditions: SDC≤SMC; TDC≤TMC; A=[1;10]; B=[1;10]

According to the theoretical concepts, Test 1 should show that the increase of either Target Data Count and Source Data Count increases linearly the Data Transfer ratio, since all target data elements have the same weight, as well as all the source data elements. It is also expected that changes provoked to the Data Transfer ratio by the variation of Target Data Count is greater than the provoked by the variation of Source Data Count, as it was defined that the use of all source data elements has, at most, the same importance that the use of one target data element.

In the graph of Figure 4.2 is shown that the increase of either the Source Data Count or the Target Data Count results in the increase of the Data Transfer ratio. Moreover, in both cases the increase of the Data Transfer ratio performed is linear. Also, as expected, the increase of the Target Data Count increases more the Data Transfer Count than the increase of the Source Data Count.

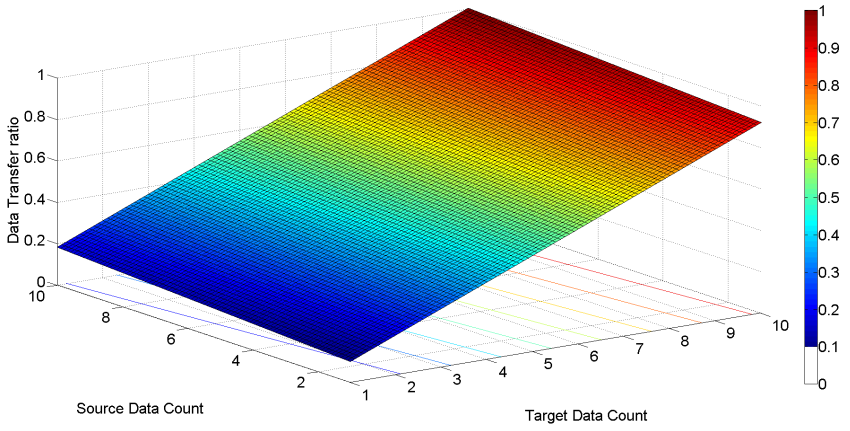

Figure 4.2: Variation of the Source and Target Data Counts

4.3.2 Set 2: Variation of a source or target model

This set is composed by two test. In the first is tested the behaviour of the mathematical expression for different target models, where for each target model is also tested the use of different interoperability specifications where only the number of data elements produced varies. In this test the Source Model and Data Counts are fixed to the value 5 and the Target Model and Data Counts vary in a range between 1 and 10. The second test is executed with the goal of test the behaviour of the mathematical expression for different source models different interoperability specifications where only the number of data elements used varies. To execute this test the Target Model and Data Counts are constant with the value 5 and the Source Model and Data Counts vary in a range between 1 and 10. These tests are presented in Table 4.5.

Table 4.5: Data Transfer ratio calculus for different source or target models

Test	Inputs					Result (Line Number)	
	I1: SMC	I2: SDC	I3: TMC	I4: TDC	I5: Expected behaviour	Expected	Actual
1	5	5	A	B	- When TDC is equal to TMC, DTr is maximum - Increase of TDC linearly increases DTr, for a given TMC value - Increase of TMC, for a given TDC, results in the decrease of the DTr, converging to the value corresponding to the contribution of the source counts	Success (4)	Success (4)
2	A	B	5	5	- When SDC is equal to SMC, DTr is maximum - Increase of SDC linearly increases DTr, for a given SMC value - Decrease of SMC, for a given SDC, results in the increase of the DTr, converging to the value corresponding to the contribution of the target counts	Success (4)	Success (4)

Initial Conditions: SDC$\leq$SMC; TDC$\leq$TMC; A=[1;10]; B=[1;10]

In this test is expected that the Data Transfer ratio varies between the maximum (value 1), reached when Target Model Count is equal to Target Data Count, and $2/(TMC+1)$ that corresponds to the use of all the source data elements and one target data element. It is also expected that, for a certain value of Target Model Count, the increase of Target Data Count results in the increase of the Data Transfer ratio, as well as is expected the increase of the Data Transfer ratio with the decrease of the Target Model Count for a given value of Target Data Count. These behaviours are expect since in all these cases the ratio between the Target Data Count and the Target Model Count increases.

In the graph shown in Figure 4.3 is noticeable that when the Target Data Count and the Target Model Count are equal, the Data Transfer ratio reaches its maximum. Also, for the same value of Target Model Count, the increase of the Target Data Count results in the linear increase of the Data Transfer ratio. The increase of the Target Model Count, for a specific Target Data Count,

produces a decrease of the Data Transfer ratio. This decrease is not linear, being more sharp for lower values of the Target Model Count

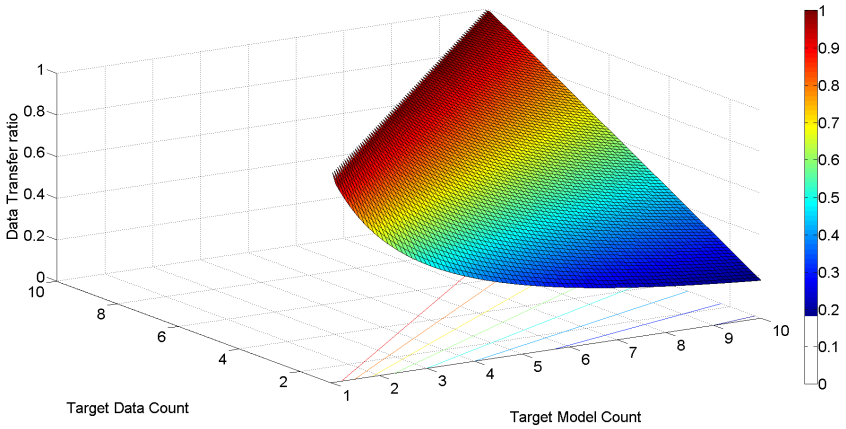

Figure 4.3: Variation of the Target Model Count and Target Data Count

.

Regarding to Test 2, is expected that for a certain value of Source Model Count, the increase of Source Data Count results in the increase of the Data Transfer ratio, as well as the increase of the Data Transfer ratio with the decrease of the Source Model Count for a given value of Source Data Count. These behaviours are expect since in all these cases the ratio between the Source Data Count and the Source Model Count increases.

The graph in Figure 4.4 is shows a behaviour similar to the behaviour noticed in Figure 4.3. When the Source Model Count is equal to the Source Data Count, Data Transfer ratio hits the scale maximum. The Data Transfer ration increases linearly when the Source Data Count increases for a specific Source Model Count, and converges to the value corresponding to the use of the target elements when the Source Model Count increases for a specific Source Data Count.

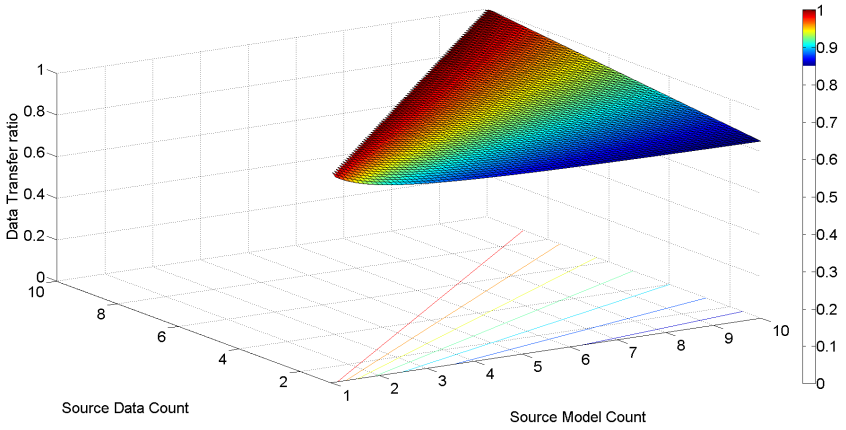

Figure 4.4: Variation of the Source Model Count and Source Data Count

4.3.3 Set 3: Different pairs of models

This set is composed by two tests. In the first are used different source and target models and the interoperability specification describes an optimal data transfer. The execution of this test considers that the Source Model and Data Counts are equal, as well as the Target Model and Data Counts, which value vary in a range between 1 and 10. The second test is similar to the first one, however the interoperability specification does not describe an optimal data transfer. In this execution the Source Data Count is 60% of the Source Model Count and the Target Data Count is 50% of the Target Model Count. These tests are presented in Table 4.6.

Table 4.6: Data Transfer ratio calculus for different pairs of models

Test	Inputs					Result (Line Number)	
	I1: SMC	I2: SDC	I3: TMC	I4: TDC	I5: Expected behaviour	Expected	Actual
1	A	A	B	B	- DTr is maximum for any value of the source count and target count	Success (4)	Success (4)
2	A	A×0.6	B	B×0.5	- For a given TMC, DTr does not depends of SMC - Weight of source coverage in DTr decreases when TMC increases	Success (4)	Success (4)

Initial Conditions: SDC≤SMC; TDC≤TMC; A=[1;10]; B=[1;10]

As the ratios between the target counts and the source counts is 1, then is expected that the Data Transfer Ratio will always have the maximum value. Through the analysis of the graph in Figure 4.5, it is verified that for any combination of source and target models, respectively represented by Source Model Count and the Target Model Count, the Data Transfer ratio is optimal.

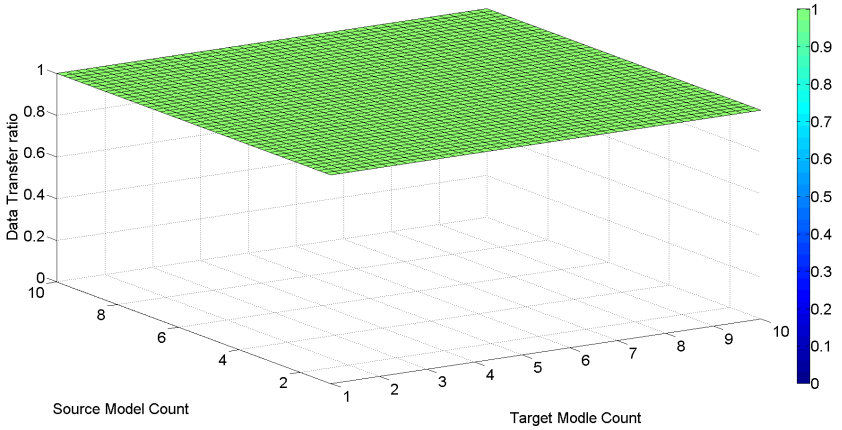

Figure 4.5: Variation of the Source Model Count and Target Model Count with with optimal source and target coverages

In Test 2 is expected that the variation of the source counts will not produce any change in Data Transfer ratio, since the importance of the use of the source elements is defined by the Target Model Count and the ratio of source data elements used is constant. It is also expected that, with the decrease of the Target Model Count, the usage of the source data elements will have a greater weight in the computation of the Data Transfer ratio, rising its value. This behaviour implies that,

with the increase of the Target Model Count, the weight of the source count will decrease, making the Data Transfer ratio converge to the value corresponding to the contribution of the use of the target data elements.

Analysing Figure 4.6, is noticeable that, for a given Target Model Count, the Data Transfer ratio does not change with the variation of the Source Model Count. It is also noticeable that with the increase of the Target Model Count, the contribution of the source coverage for the Data Transfer ratio decreases.

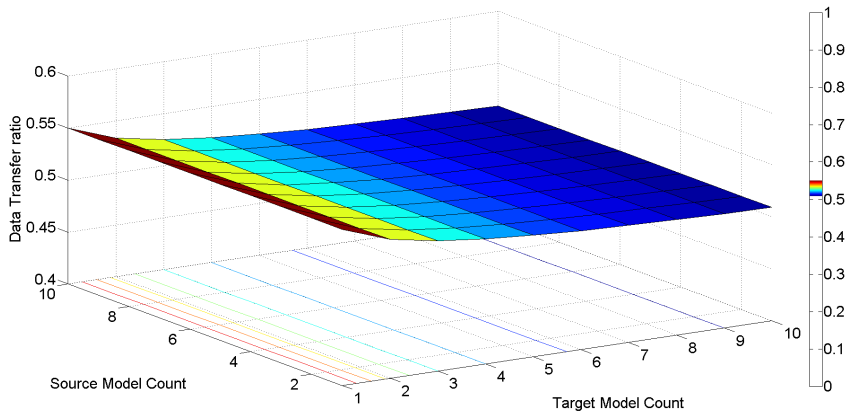

Figure 4.6: Variation of the Source Model Count and Target Model Count with different source and target coverages

4.4 Verdict

Analysing the result of the executes tests, it can be concluded that the mathematical expression is in conformance with the theoretical concepts since all the test results successfully match the corresponding expected results. In order to allow the validation of the mathematical expression, the characteristics of the problem must be fulfilled. Therefore, it must be proven that he mathematical expression: 1) deals with the lack of semantic definitions, 2) can handle the heterogeneity of the interoperability specifications, and 3) allows the comparison between different pairs of data formats.

Regarding to the first characteristic, as the mathematical expression only resorts to the number of data elements to perform the calculus, can be concluded that no semantic definition is used, being used the relations between the number of data elements defined in the models and the number of data elements used.

A similar reasoning can be developed regarding to the second characteristic. Since the mathematical expression uses the number of data elements inside models, then it does not resorts to any direct parameter of the interoperability specification, which proves that the mathematical expression is independent of the heterogeneity of interoperability specifications.

The fulfilment of the last characteristic is proven by the tests executed. Test 1 from Set 1 proves that, for the same pair of source and target model, can exist several interoperability specifications, being the values of Data Transfer ratio correspondent to each one comparable. Test 1 from Set 2 proves that, varying the target model and the interoperability specification used, while fixing the source model, the values obtained are comparable. A similar conclusion can be draw from Test 2 from Set 2, where for a fixed target model is used, varying the source models and the interoperability specifications. Test 1 and 2 from Set 3 shows that the Data Transfer ratio does not depend of the source and target models, but from the ratio of data elements used, being the Data Transfer ratio constant for the the same ratio of source data elements used and target data elements produced.

A situation occurs when, for a given element mapping, the TDC is zero. In this case, as no data element is generated, the Data Transfer ratio should be zero. However, the Data Transfer ratio assumes the value corresponding to the use of the source data elements. Nevertheless, this situation can be overlooked through the consideration that, in order for a element mapping describe a data exchange, it needs to use at least one source data element (SDC$>$0) and must produce at least one target data element (TDC$>$ 0).

5

Conclusions and Future Work

Internet of Things consists in a network of objects, which can be connected to the Internet. Internet of Things scenarios are usually composed by heterogeneous object, implementing different standards, for the same or for different application scenario. This heterogeneity can cause communication problems at physical and data levels, where the physical level refers to the physical connections and the data level refers to the data exchange between the objects. This work addresses the communication problems related to the data level. An approach to this kind of communication problems has been defined in the research group Uninova-GRIS, being defined the concept of data formats to represent the objects and the concept of interoperability specification to specify the steps that need to be taken in order to enable the data exchange between two data formats. This approach is called Plug'n'Interoperate.

In a large scenario, composed by several objects, there can exist several interoperability specifications that need to be managed. For instance several interoperability specifications can be defined for describing the data exchange between two data formats. This situation is problematic as only one can be used, which implies the selection of the most suitable for that scenario. Usually the preference falls for the interoperability specification that provides the best data exchange, which implies the measurement of the data exchange provided by each interoperability specification in order to find the best one. Other interesting situation is the comparison between many (or even all) interoperability specifications in the environment, regardless of the pair of data formats that it corresponds, a select the most suitable to be improved. This situation also requires the measurement of the data exchange provided by each interoperability specification, in order to use it as base of comparison.

So then, How to measure the data exchange provided by an interoperability specification in an Internet of Things environment? This is the research question that defines the problem addressed in this work. The key characteristics of the problem are:

1. Lack of Semantic Definitions - From the research group experience it is known that the manufacturers of IoT devices do not provide semantic definitions between the concepts related to sensors, which difficult the use of semantic relations to perform the measurement.
2. Heterogeneity of Interoperability Specifications - Interoperability specifications can be defined using different languages/technologies, therefore the solution needs to address this heterogeneity in order to allow the measurement of the data exchange described by an interoperability specification.
3. Comparison between Different Pairs of Data Formats - The solution must produce results able to be compared since, to determine the best or the worst interoperability specification, the comparison between measurement result is required. The comparison must be independent of the pair of data formats used in the data exchange.

The background research focuses in the identification of approaches that use the concept of interoperability specification and present ways to measure it. From this research four approaches were identified, described, analysed: *a)* Approach 1: Measurement of Metrics of the Model Transformation; *b)* Approach 2: Measurement of Metrics of the Models; *c)* Approach 3: Model Transformations Verification using Assertions; and *d)* Approach 4: Model Transformation Analysis using Alloy.

The aim of the background research is the identification of the contribution of each approach to the hypothesis. Regarding to the first characteristic, the concept of Completeness proposed by Approach 1 as the relation between data elements, will be addressed by the hypothesis, focusing the measurement in the data exchange. Regarding to the second characteristic, the hypothesis will measure the data exchange without looking into the model transformation, not changing the source or target models. To achieve this, the hypothesis will compare properties of the source data with properties of the target data, as proposed by approaches 3 and 4, produced by the execution of the model transformation, as proposed by Approach 2. With respect third characteristic, the hypothesis will produce as result number within a well-defined range of values, in order to allow to comparison of data exchanges between the same pair of models and between data exchange defined between any models, as proposed by Approach 1. Approach 4 proposes the generation of an instance of the source model to use in the model transformation, which enables the execution of a generic measurement as it does not depends of a specific instance of the source model.

The hypothesis is developed to be used in the a scenario were exist several interoperability specifications, either defined between the same pair of data formats or between different pairs of data formats. The hypothesis proposed in this work consists in a measurement method that performs the classification of each interoperability specification based on the data exchange provided that

it performs. This classification supports the comparison between data exchanges performed between the pair of data formats and between data exchanges performed between different pairs of data formats.

In order to perform the measurement in this scenario, the measurement method requires two inputs: a source model to describe the transmitter object, a target model to describe the receiver object, and a transformation script to provide an executable interoperability specification. The output of the measurement method is called Data Transfer ratio as it evaluates the data exchange based in the ration of data elements, of the source and target models, used by the transformation script. In order to determine the Data Transfer ratio, four parameters are used: *1)* Source Model Count representing the number of data elements defined in the source model; *2)* Source Data Count representing the number of data elements, defined in the source model, used by the interoperability specification; *3)* Target Model Count representing the number of data elements defined in the target model; and *4)* Target Data Count representing the number of data elements, defined in target model, produced by the interoperability specification.

Regarding to the measurement method, it starts with the construction of a representation for the source and target models, called control table. Using these representations, a structure is made to be used as support to the calculation of the four parameters used to determine the Data Transfer ratio. This structure is computed through multiple executions of the transformation script in order to test how each data element of the source model influences the model produced by the transformation script. Constructed the structure, it is then computed in order to determine the four parameters that are used as inputs to the mathematical expression used to determine the Data Transfer ratio related to the transformation script executed.

In order to validate the hypothesis, tests need to be made where is assessed the conformity of the hypothesis regarding the characteristics of the problem. The testing process is based on the standard ISO 9646, where an abstract test suit is defined independently of the implementation. The tests are defined using a notation based on TTCN-2.

The testing of the hypothesis consists in the testing of the mathematical expression used to calculate the Data Transfer ratio. To test the mathematical expression it is computed using specific inputs, being the output presented as a graph. This graph is then analysed and the behaviour observed is compared with the behaviour expected from the theoretical concepts. Three sets of execution tests are defined to be executed: *1)* different interoperability specifications for a pair of models, *2)* variation of a source or target model, and *3)* different pairs of models. For all the tests executions the behaviour observed corresponds to the expected behaviour which validate the hypothesis against the characteristics of the problem.

5.1 Future Work

The measurement method proposed in this work presents two limitations, caused by the use of source and target models with a large number of data elements defined, that should be addressed. The number of data elements in the target model has influence in the resolution of the Data Transfer ratio, the increase of the number of data elements in defined in the target model reduces the influence of the each data element in the Data Transfer ratio. This limitation can lead to situations where, due to the lack of significant figures, two transformations scripts generating, each one, a model conform to the same target model and producing a similar but not equal number of data elements, have the same Data Transfer ratio value. These situations are more noticeable with source models.

Another limitation caused by the definition of a large number of data elements in the source model is the time required to construct the testing table, since the number of executions required to construct this structure is equal to the number of data elements defined in the source model plus one execution. This limitation can slow the overall performance of the measurement method, which can be problematic in situations where the all the transformation scripts in the system need to be classified since the number of transformation scripts in the system can also be large.

Other aspect that can be addressed as future work is to limit the utilisation of the data elements to the data elements used by a certain application. This change would produce more reliable results as data exchange will only be performed between the data elements that will be use by the application. The data elements used vary from application to application which would require the use of new parameters to identify the data elements that will be used.

5.2 Publications

From this work resulted two scientific articles titled "‘Towards measuring information interoperability based on model transformations"’, which was published in the 6th Iberian Conference on Information Systems and Technologies (CISTI) in 2011, and another titled "‘Towards an Interoperability Management System"’ published in the same conference. The first article presents initial ideas and concepts considered in the initial stages of this work. In the second article is performed a contribution to the approach presented by proposing an extension to it.

6

Bibliography

Amstel, M. van. (2011). *Assessing and improving the quality of model transformations*. Unpublished doctoral dissertation, Eindhoven University of Technology.

Anastasakis, K., Bordbar, B., & Küster, J. (2007). Analysis of model transformations via alloy. *ModeVVa'07*, 47–56.

Asztalos, M., Lengyel, L., & Levendovszky, T. (2010). Towards automated, formal verification of model transformations. In *Software testing, verification and validation (icst), 2010 third international conference on* (pp. 15–24).

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, *54*(15), 2787–2805.

Callaway, E., Gorday, P., Hester, L., Gutierrez, J., Naeve, M., Heile, B., et al. (2002, aug). Home networking with ieee 802.15.4: a developing standard for low-rate wireless personal area networks. *Communications Magazine, IEEE*, *40*(8), 70 - 77.

CERP-IoT. (2010). Vision and challenges for realising the internet of things. *Cluster of European Research Projects on the Internet of Things (CERP-IoT)*.

Czarnecki, K., & Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, *45*(3), 621–645.

IEEE. (1990). *IEEE standard computer dictionary. a compilation of IEEE standard computer glossaries.* Available from `http://ieeexplore.ieee.org/servlet/opac?punumber=2267` (IEEE Std 610)

Jackson, D. (2006). *Software abstractions: Logic, language, and analysis*. The MIT Press. Available from `http://www.amazon.com/Software-Abstractions-Logic-Language-Analysis/dp/0262101149%3FSubscriptionId%3D0JYN1NVW651KCA56C102%`

26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0262101149

Jouault, F., & Kurtev, I. (2006). Transforming models with ATL. In *Satellite events at the models 2005 conference* (pp. 128–138).

Katasonov, A., Kaykova, O., Khriyenko, O., Nikitin, S., & Terziyan, V. (2008). Smart semantic middleware for the internet of things. In *Proceedings of the 5-th international conference on informatics in control, automation and robotics* (pp. 11–15).

Onofre, S. M. (2007). *Plataforma para testes de conformidade de sistemas baseados em módulos conceptuais step*. Unpublished master's thesis, Departamento de Engenharia Electrotécnica ; Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia. Available from http://hdl.handle.net/10362/1870

Saeki, M., & Kaiya, H. (2007). Measuring model transformation in model driven development. In *Proceedings of the CAiSE* (Vol. 7).

Schafersman, S. (1997). An introduction to science: Scientific thinking and the scientific method. *Online Whitepaper*. Available from http://geoweb.tamu.edu/Faculty/Herbert/geol641/docs/ScientificMethod.pdf

Seidewitz, E. (2003, sept.-oct.). What models mean. *IEEE Software*, *20*(5), 26 - 32.

Sendall, S., & Kozaczynski, W. (2003). Model transformation: The heart and soul of model-driven software development. *IEEE Software*, *20*(5), 42–45.

Singh, Y., & Sood, M. (2009). Models and transformations in mda. In *Proc. first int. conf. computational intelligence, communication systems and networks cicsyn '09* (pp. 253–258).

Tretmans, G. J. (1992). *A formal approach to conformance testing*. Unpublished doctoral dissertation, Enschede. Available from http://doc.utwente.nl/58114/

Tretmans, J. (2001). An overview of osi conformance testing. *Samson, editor, Conformance Testen, in Handboek Telematica*, *2*, 4400.