Based on Past Experience: Highlighting Potential Human Value Issues in Domain Modelling

Jasneet Kaur

A thesis submitted to McGill University
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

McGill University
Montréal, Québec, Canada

April 2023

Abstract

In this technologically evolving era, important human values such as freedom and social responsibility are frequently overlooked in software systems, which can have significant negative social consequences. For example, in an infamous breach Facebook provided unauthorized access to more than 50 million user profiles during US elections or Delta Airlines ticket prices spiked during a natural disaster. Therefore, it is important to help software developers incorporate human values considerations throughout the software development process. In this thesis, we focus on domain modelling with class diagrams, an important technique for requirements engineering and early design activities. To investigate how model elements in a domain model may affect various human values, we analyze the domain model of WhatsApp through the lens of Schwartz's taxonomy of human values to compile a list of issues concerning human values. To collect these past experiences including how to mitigate them, we propose a domain-specific language called HVT (Human Value Trigger). Practitioners may utilize this language to contribute more such examples to grow a catalogue of human value issues over time. Furthermore as proof-of-concept, a prototype implementation addresses the need for human values to be integrated in software engineering by providing suggestions on the model element type, name, and the semantics based on synonyms for a domain model based on these collected past experiences. To find synonyms of a model element, an analysis of eight synonym services is performed to find the optimal synonym service or combination of synonym services to use with the implementation. Moreover, an extensive testing process is performed to evaluate the capabilities of the prototype.

Abrégé

En cette ère d'évolution technologique, des valeurs humaines importantes telles que la liberté et la responsabilité sociale sont souvent négligées dans les systèmes logiciels, ce qui peut avoir des conséquences sociales négatives importantes. Par exemple, lors d'une tristement célèbre violation, Facebook a fourni un accès non autorisé à plus de 50 millions de profils d'utilisateurs lors des élections américaines ou les prix des billets de Delta Airlines ont grimpé en flèche lors d'une catastrophe naturelle. Par conséquent, il est important d'aider les développeurs de logiciels à intégrer les considérations relatives aux valeurs humaines tout au long du processus de développement de logiciels. Dans cette thèse, nous nous concentrons sur la modélisation de domaine avec des diagrammes de classes, une technique importante pour l'ingénierie des exigences et les premières activités de conception. Pour étudier comment les éléments de modèle dans un modèle de domaine peuvent affecter diverses valeurs humaines, nous analysons le modèle de domaine de WhatsApp à travers le prisme de la taxonomie des valeurs humaines de Schwartz pour compiler une liste de problèmes concernant les valeurs humaines. Pour collecter ces expériences passées, y compris comment les atténuer, nous proposons un langage spécifique au domaine appelé HVT (Human Value Trigger). Les praticiens peuvent utiliser ce langage pour contribuer davantage d'exemples de ce type afin de développer un catalogue de problèmes de valeur humaine au fil du temps. De plus, en tant que preuve de concept, une implémentation de prototype répond au besoin d'intégrer les valeurs humaines dans le génie logiciel en fournissant des suggestions sur le type d'élément de modèle, le nom et la sémantique basée sur des synonymes pour un modèle de domaine basé sur ces expériences passées collectées. . Pour trouver des synonymes d'un élément de modèle, une analyse de huit services synonymes est effectuée pour trouver le service synonyme optimal ou la combinaison de services synonymes à utiliser avec l'implémentation. De plus, un processus de test approfondi est effectué pour évaluer les capacités du prototype.

Acknowledgements

I would like to take this opportunity to express my heartly gratitude towards all those individuals who have helped me in the successful completion of my thesis. First and foremost, I would like to thank my thesis supervisor Professor Gunter Mussbacher, whose constant support, guidance, encouragement, and expertise was invaluable in completing my research and writing. It has been an honour and a privilege to have worked under his guidance.

Other than my supervisor, I would also like to extend my sincere thanks to my fellow student and friend Prabhsimran Singh and Rijul Saini who helped me immensely with the understanding of certain research related concepts that could have been difficult for me.

Finally, I would like to thank my parents, Mr. Balwinder Singh and Mrs. Ranjit Kaur and my brother Inderveer for their constant support, encouragement, and countless blessings throughout the course of my study and in life. I am also immensely thankful to my friends in Montreal who made these years great fun. This accomplishment would not have been possible without them.

Thank you.

Table of Contents

List of Tables

List of Figures

List of Listings

List of Abbreviations

AI	Artificial Intelligence
HVT	Human Value Trigger
HVTF	humanValueTriggerFile
HVTS	Human Value Trigger System
NLP	Natural Language Processing

1 Introduction

Human values are crucial in life and serve as an inspiration for all of one's action. People use values as a criterion to assess actions, people, and events. We all have a variety of values that are important to us in different ways. In recent years, there has been an increased focus on the impact of systems on human values [40]. Modern socio-technical systems have great influence on the interpersonal relationships amongst people as well as human-machine interactions. Due to their ever-increasing importance and wide-ranging impact on our daily lives and society in general, it is crucial to incorporate human values into these system. When these values are missed, they often lead to considerable social consequences. For example, in an infamous breach, Facebook provided unauthorized access to more than 50 million user profiles to a data firm during US elections which were subsequently utilized to customize political adverts for individual US voters to influence their voting decisions [7]. Another example is the Delta Air Lines scandal where people from evacuating areas hit by Hurricane Irma were charged by the ticket reservation system five times more than the usual ticket price. It was seen as a breach of human values by the system [19]. In another instance, Instagram was partially blamed for the suicide of a British teenager who was exposed to self-harming images 'normalized' among other images [2] which resulted in a huge backlash from the public. Consequently, it is important to incorporate human values considerations throughout the software development process.

1.1 Problem Statement

Thesis Statement. Lack of consideration for human values when developing software often leads to social repercussions. The proposed Human Value Trigger System aims to address the need for human value consideration during domain modelling by providing suggestions based on collected past experiences.

Considering human values in software engineering is challenging due to the lack of methods to track and incorporate those values in all phases of development cycle. This leads to the development of software systems that act in a different way than anticipated and have detrimental consequences. Galhotra *et al.* [14] highlight these incidents and express the necessity to integrate human values into the software.

A solution is required to help practitioners consider human values when building different software. Some approaches exist that aim to consider human values in software development. Jon Whittle *et al.* highlight the importance of human values and present ideas to address them in software [40]. Perera *et al.* demonstrate that GDPR can be utilized to incorporate concrete definitions to human values in the context of software [29]. Mougouei *et al.* offer a roadmap to operationalizing human values in software [23]. However, there is currently no tool support for human value analysis in domain modelling - an important activity for requirements engineering and early design activities, even though Mussbacher *et al.* [25] offer initial evidence that domain models indeed incorporate human values. To address this challenge, we explore further human values in domain modelling and provide tool support to incorporate value focused elements to avoid human value-based implications.

1.2 Thesis Methodology and Contribution

To incorporate human values in domain model, this thesis proposes a domain-specific language called HVT (Human Value Trigger) which captures different examples for potential human value issues. This includes a detailed description that shows how the presence or absence of model element in the domain model positively or negatively impacts the human values identified by Schwartz's taxonomy [34].

Furthermore, this thesis proposes the Human Value Trigger System (HVTS) that provides suggestions for a domain model based on captured past experiences. The proposed system takes two inputs, a domain model and the human value trigger file. The system iterates over the model elements of the human value trigger file and the domain model and performs various checks and comparisons. These checks and comparisons determine suitable matches found based on model element type, name, and the semantics based on synonyms. Once the system completes the matching process, it presents possible suggestions to the modeller, asks the modeller to select which suggestions to incorporate in the domain model, and then integrates the modeller-selected suggestions in the domain model.

To demonstrate the feasibility of HVTS, a prototype tool that implements the proposed system is developed as proof-of-concept. A comprehensive test suite containing tests with different scenarios for domain models and human value trigger files ensures the matching capabilities of the system.

The contributions of this thesis in the form of the Human Value Trigger System are as follows:

- To further illustrate the need to include value-based elements in domain modelling, the domain model of the WhatsApp system is examined while taking into account the values in Schwartz's taxonomy [34].
- A domain-specific language called Human Value Trigger is specified that enables practitioners to contribute more examples to the captured past experiences.
- When a new system is built, the Human Value Trigger System provides information to modellers to help them be aware of the implications of human values that might be present in the system based on similar past situations.
- To find the synonyms for a given name of a model element, an analysis of eight synonym

services and their combinations is performed to find the best synonym service or combination of synonym services to use in the system. These synonym services include dictionaries and thesauri along with Natural Language Processing (NLP)-based and Artificial Intelligence (AI)-based services.

- A test suite containing tests with varied human value trigger files and domain models is used to verify the HVTS.

1.3 Thesis Overview

The thesis is organized as follows:

- Chapter 2: Background

  This chapter explains Schwartz's taxonomy of human values, the concept of the domain model, TouchCORE (the software and the modelling framework used for the proof-of-concept implementation), the concept of the metamodel, and the Xtext language used to specify the proposed domain-specific language in this thesis. The author contributed to the full chapter.

- Chapter 3: Motivating Example

  This chapter includes an analysis of the WhatsApp domain model to motivate our approach to incorporate values in domain modelling. The author contributed to the full chapter.

- Chapter 4: Metamodel

  The metamodel used for HVT is explained in detail in this chapter. This includes the purpose and definition of all metamodel elements and the description of the grammar definition specified with Xtext. The author contributed to the full chapter.

- Chapter 5: Analysis of Synonym Services

  This chapter includes an analysis of eight synonym services and their combinations to find the best synonym service or combination of synonym services to use for the proposed system in this thesis. The author contributed to the full chapter.

- Chapter 6: Human Value Trigger Algorithm

This chapter discusses the HVTS algorithm that provides the suggestions to integrate in the domain model and the verification process followed to evaluate the capabilities of HVTS. The author contributed to the full chapter.

- Chapter 7: Related Work

  This chapter discusses similar work done by various authors and outlines the difference between our work and prior work. The author contributed to the full chapter.

- Chapter 8: Conclusions

  This chapter summarizes the contributions of our thesis and discusses future work. The author contributed to the full chapter.

2
Background

This chapter provides background information relevant to understand this thesis. Section 2.1 summarizes Schwartz's basic theory of human values which is used in our research. The model elements of class diagrams which is the notation used for domain modelling are discussed in Section 2.2. Section 2.3 gives information about TouchCORE, a modelling tool used to create domain models, i.e., the input and output for our proposed Human Value Trigger System (HVTS). Furthermore, the concept of metamodel is described in Section 2.4 together with information about the Xtext language which is used to create our proposed domain-specific textual language called Human Value Trigger (HVT) in Section 2.5.

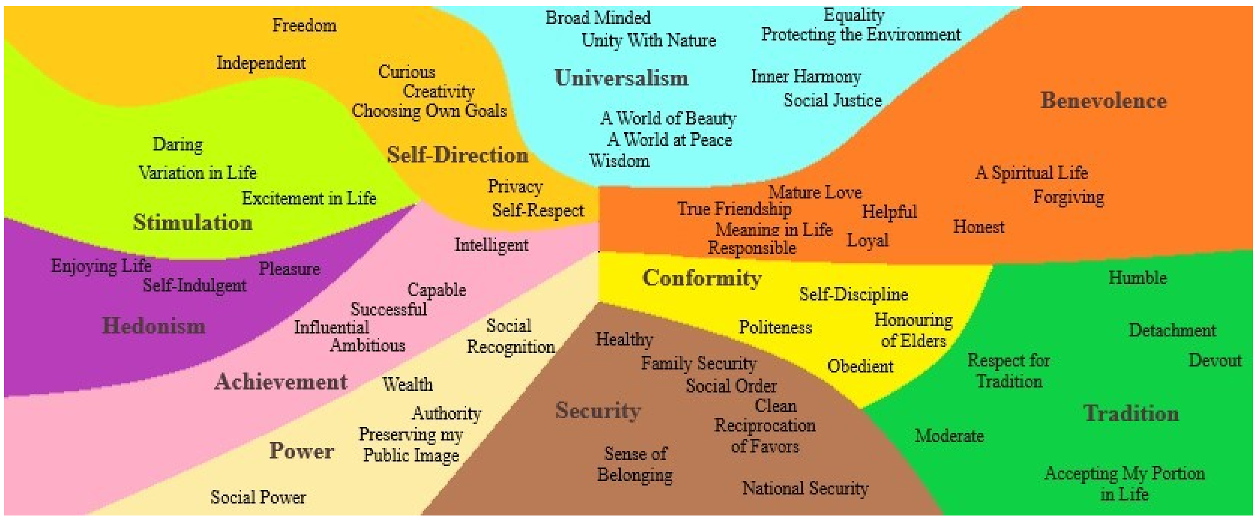


Figure 2.1: Schwartz's Theory of Basic Human Values (adapted from source [31])

2.1 Schwartz Taxonomy

Human values are valuable in our life as they help us to grow and develop. In 1992, Schwartz [34] [31] introduced the basic theory of human values which defines ten broad category values according to universal requirements of human existence which are needs of the individual, social interaction, survival, and welfare. The type of goal or motivation each human value represents is the underlying means to set them apart from one another. It measures these ten motivationally distinct category values using 58 distinct values. The ten category values are arranged in a circular structure as shown in Figure 2.1 to depict the relations of conflict between different values. Values located close to each other are complementary, whereas values further apart tend to be in tension with each other and similarly values which are diagonal are more opposite to each other and are harder to reconcile. For example, values that highlight the concern for welfare and the interest of others (universalism, benevolence) conflict with the values that show an individual's own ambitions and success and control over others (power, achievement). In our work, we use the values defined by Schwartz to identify positive or negative impact on them due to the presence or absence of model elements in a domain model.

2.2 Domain Modelling

A domain model [6] is the visual depiction to describe and model real world entities and relationships of a problem domain. It is used to represent the selected concepts of a domain to solve the problem. To create a domain model, the Unified Modeling Language (UML) is used which is a general-purpose modelling language to visualize the design of a system [18]. In UML, a class diagram is one way to describe the structure of the system. Basically, a domain model uses the class diagram notation to specify a particular problem. However, a domain model does not use all model elements of a class diagram (e.g., it does not specify operations) [32].

The domain model elements include classes, attributes, enumerations, and relationships between different classes. A class describes a set of similar objects, it can be a thing, place, an event and many more. An example domain model is shown in Figure 2.2. Here, Owner, LicencePlate, Wheel, Motor, ParkingCompany, Vehicle, Car, and Truck are the classes for the domain model. An attribute is a piece of data related to the class. It has a simple data type associated with it (String, Integer, Date, Time etc.). For example, weight is the attribute for Vehicle with datatype as double. An enumeration describes a predetermined list of options. An individual option is referred as literal, e.g., TransmissionType in the example represents an enumeration whereas Manual and Automatic are its literals.

A class may be involved with one or more relationships with other classes. A relationship can be an association, composition, aggregation, or generalization. An association defines a relationship between two classes and describes the role and multiplicity each class plays in this relationship. A multiplicity defines the number of instances of a class that are related to an instance of the other class in the relationship. It can be one (1), many (*), and optional (0). By default, these association are considered bi-directional, but it is possible to limit the direction by using directed arrows. For example, the relationship between Owner and Vehicle represents an association between them.

A composition is like an association, but it is considered a whole-part relationship. Furthermore, if the parent (container) in a composition is destroyed, the parts will automatically be destroyed. For example, a Vehicle is composed of at least one Motor and at least four Wheels. An aggregation is also considered a whole-part relationship except the parts are not destroyed with the parent (e.g., the relationship between Vehicle and LicencePlate).

A generalization represents an “is-a” relationship between two classes. In this relationship, the subclass inherits the features of the superclass, e.g., “a car and a truck are a vehicle” where Vehicle is the superclass and Car and Truck are its subclasses. An association class is used to add properties to an association, aggregation, or composition that cannot be placed in either class participating in the relationship. Furthermore, an association class stipulates that at the most one instance of the association class exists with the same pair of instances from the classes participating in the relationship (e.g., the Registration association class).

In our work, we focus on how to incorporate human values in domain modelling. To accomplish this task, we use domain models as input to HVTS to provide suggestions based on model elements matched to past experiences.

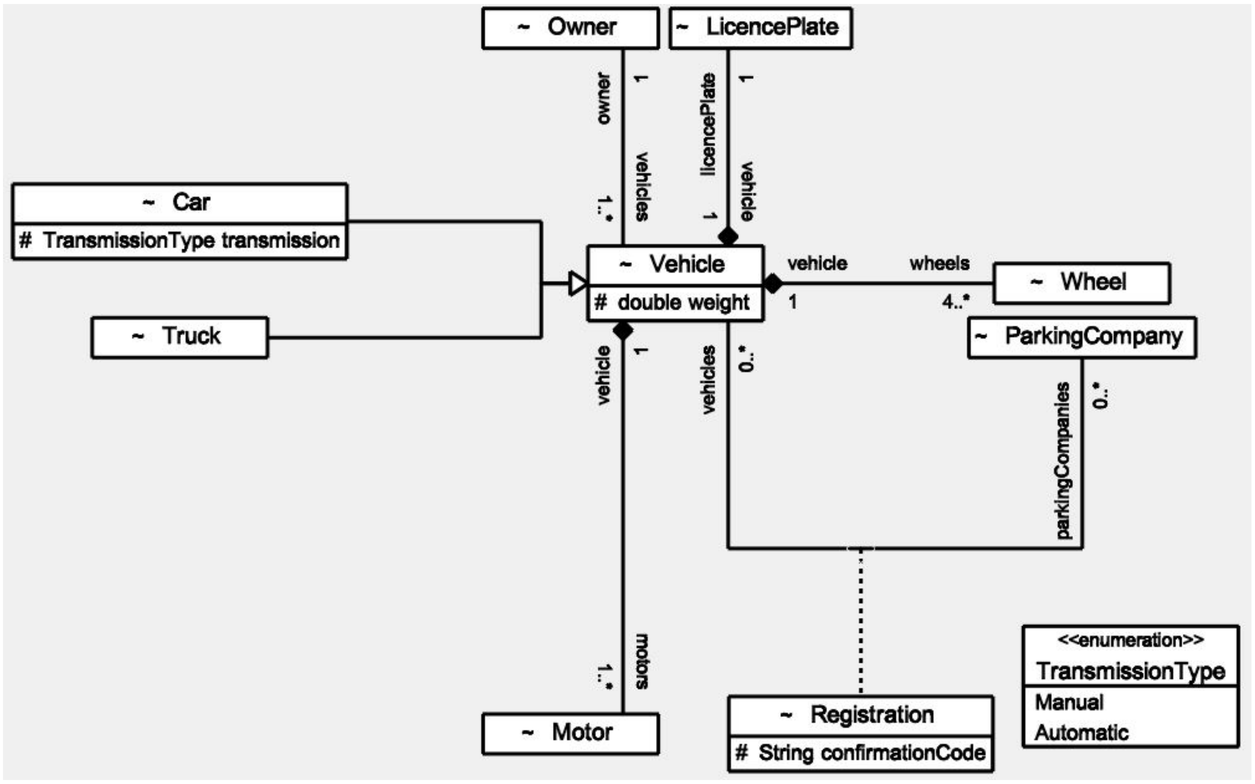


Figure 2.2: Domain Model Example Expressed with UML Class Diagram

2.3 TouchCORE

TouchCORE [38] [33] is a multi-touch enabled tool for creating reusable design models. It is built on top of EMF (Eclipse Modeling Framework) [10] and MT4J (a multi-touch library for java) [24].

We use the TouchCORE class diagram editor to define domain models which we then analyze for potential human value issues.

Figure 2.3 shows the TouchCORE editor for creating domain models. It allows the creation of classes, enumerations, and data types. After a class has been created, it allows the addition of attributes, associations, aggregations, compositions, generalizations, and association classes.

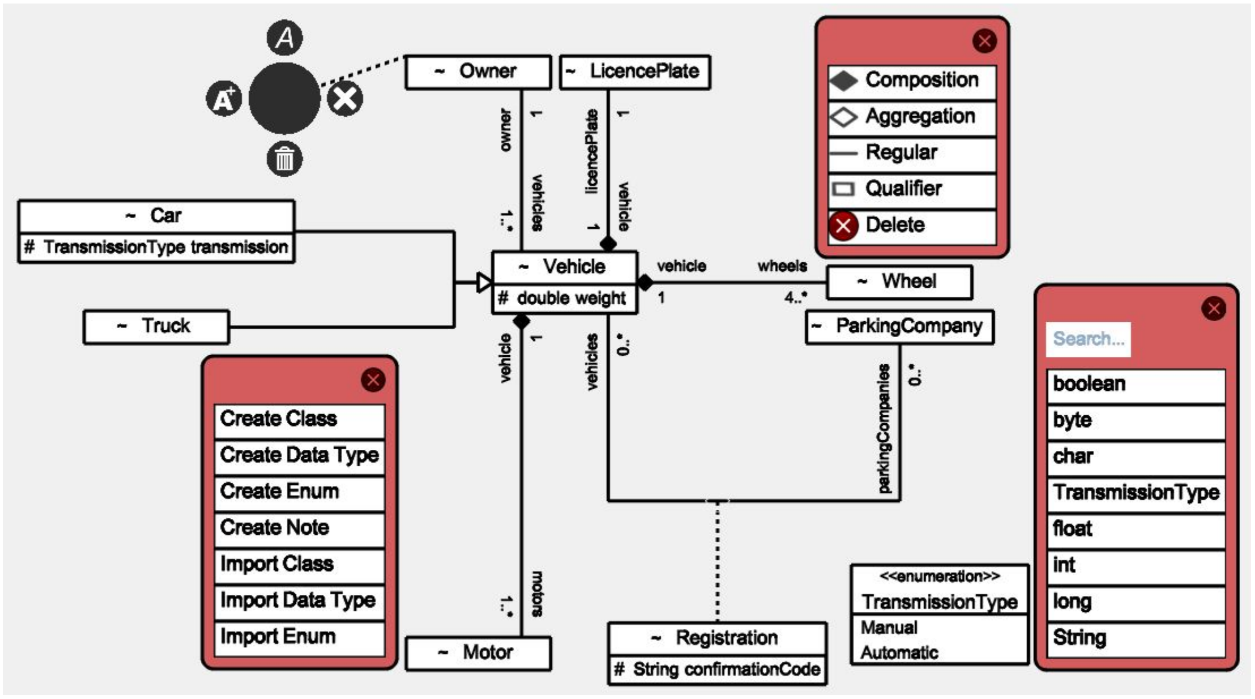


Figure 2.3: TouchCORE Editor for Creating Domain Models

2.4 Metamodel

A metamodel [9] refers to a model of a model. Basically, a metamodel is a higher-level model which describes a language to define the lower-level model. As an example of a metamodel, the TouchCORE metamodel for class diagrams (CDM) is shown in Figure 2.4. To define a class in the domain model, the Class from the CDM metamodel is used. Every Class is a Classifier which sets the properties such as datatype, abstract, and visibility type for the user-defined class. This Classifier class is contained in the root class called ClassDiagram. An attribute for a Class is defined using the Attribute class which is contained inside the Classifier. To define the datatype for an Attribute, the metamodel support the primitive datatypes (e.g., String, Boolean, . . .) defined under

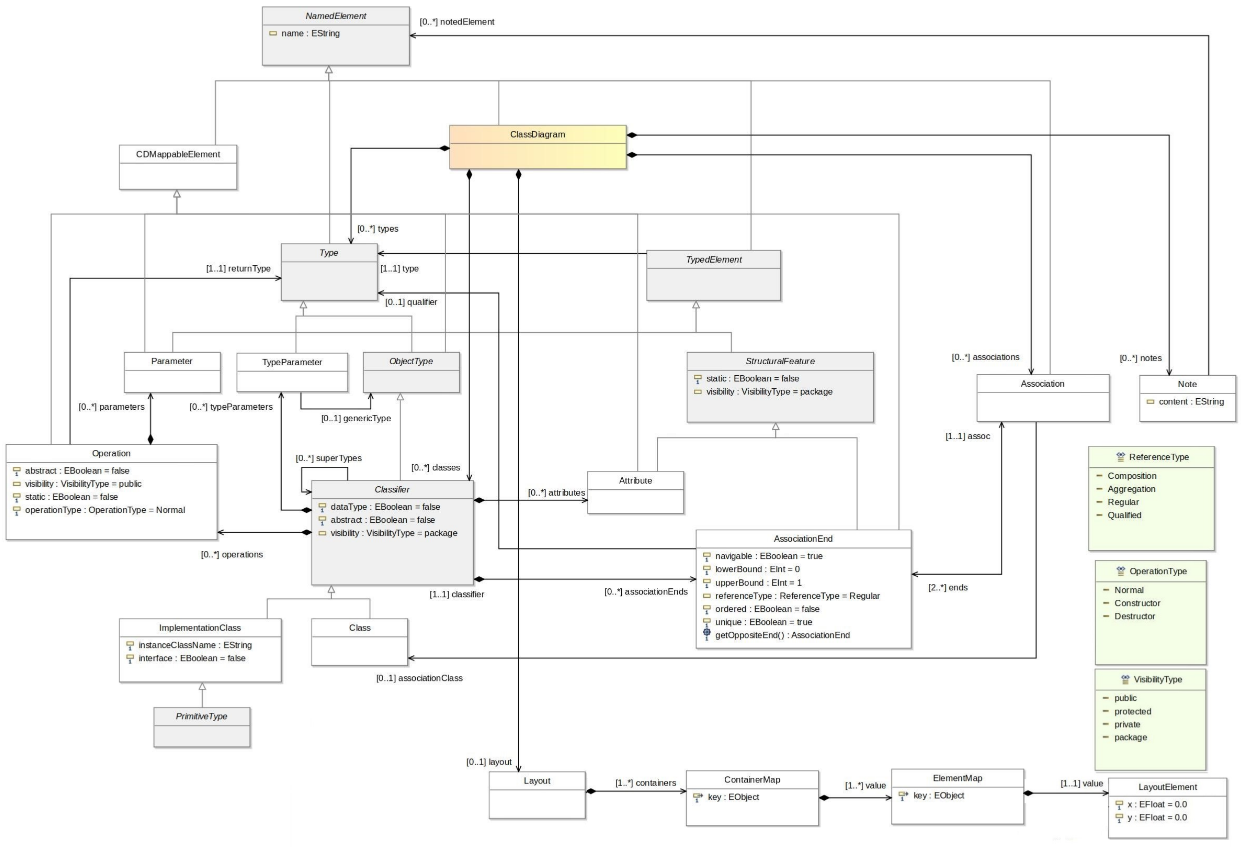


Figure 2.4: Class Diagram Metamodel for TouchCORE (adapted from source [5])

the PrimitiveType Class. The datatypes present in the CDM metamodel are shown in Figure 2.5. To define the relationships between two classes, an Association class from the metamodel is used which has two or more AssociationEnds attached to it. The AssociationEnds are contained in their Classifier. The lowerBound and upperBound attributes in the AssociationEnd class are used to define the multiplicities for the association end. Similarly, the referenceType attribute allows us to define a composition, aggregation, regular, or qualified relationship. All Types and Associations created are contained inside the root class. The inputs to HVTS are created using this metamodel and the integration of recommended suggestions that are accepted by the modeller also requires knowledge of the CDM metamodel.

Figure 2.6 illustrates the concept of metamodel hierarchy with the help of the MindMap example. The lowest-level layer is M0 which refers to the user objects in memory space, i.e., the runtime instances of the M1 layer. The M1 layer is an instance of the M2 layer and correlates

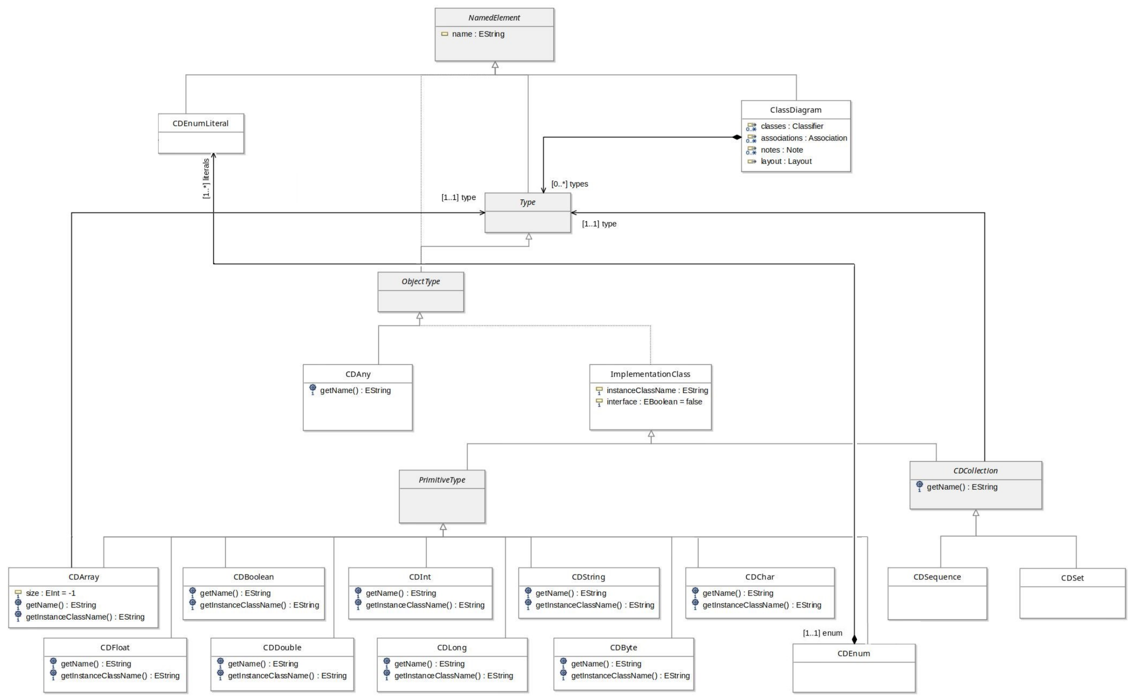


Figure 2.5: Datatypes in Class Diagram Metamodel for TouchCORE (adapted from source [5])

to the user model. As an example of the M1 layer, Figure 2.6 shows a model that contains the Robotics MindMap, two Topics named as Sensors and Ultra-Sonic, and two links named as topics and subtopics. Another example of M1 is an input domain model created using the TouchCORE metamodel. To create these models, we need something that can be used to define these instances. Similar to the relationship between the M1 layer and M2 layer, the M2 layer is an instance of the M3 layer and corresponds to the metamodel layer. In the example, the M2 layer contains the metamodel of MindMap, i.e.,the MindMap class that is used to define Robotics, and the Topic class that is used to define Sensors and Ultra-sonic. It also contains the composition between MindMap and Topic that is used to define topics and the self-composition of Topic that is used to define subtopics. Another example of M2 is the CDM metamodel of TouchCORE. The highest-level layer is M3 which refers to the metametamodel layer that is used to define languages to specify metamodels (i.e., metalanguages). In the example, the M3 layer includes the language elements from the Ecore metalanguage that are used to define MindMap and Topic as EClass. Similarly, EAttribute is used to define the attribute description, and EReference is used to define the compositions and associations in M2. The CDM metamodel is also defined in Ecore.

In summary, the User Objects layer (M0) describes a specific situation in an information do-

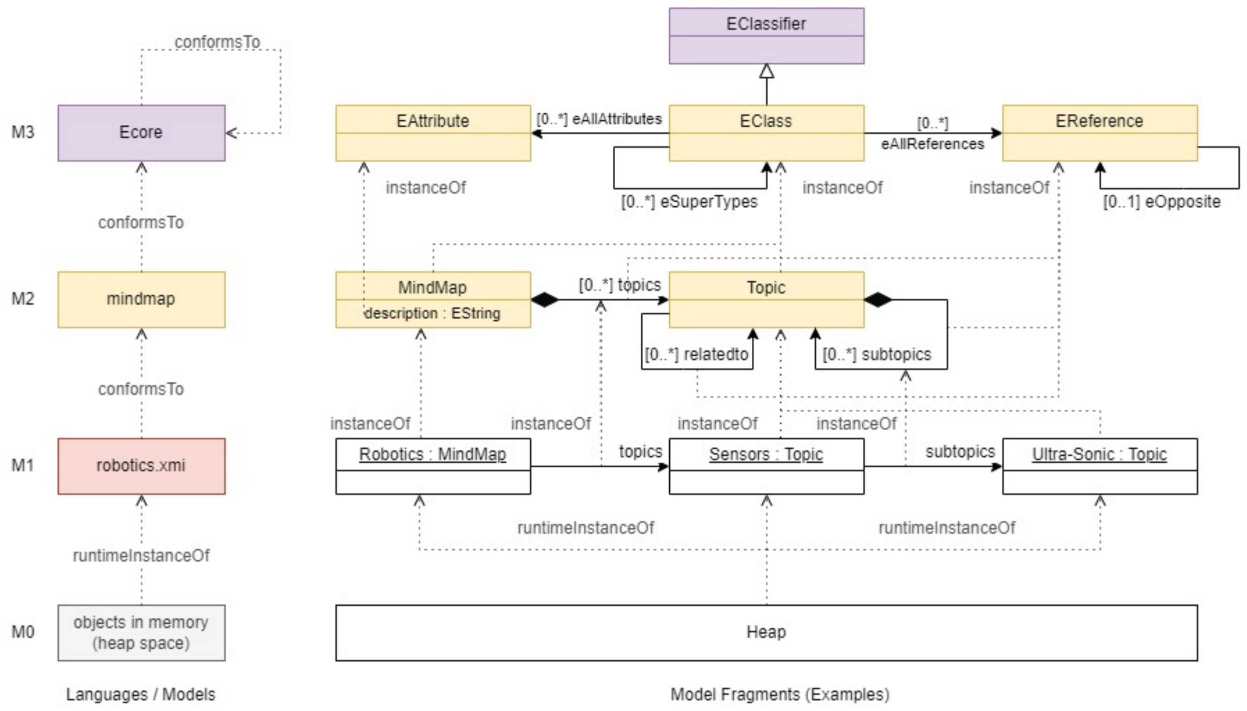


Figure 2.6: Metamodel Hierarchy (adapted from source [3])

main. The Model Layer (M1) defines a model to describe the information domain, e.g., a model created using UML. To define M1 models, technologies like UML and CWM (Common Warehouse Metamodel) are used which are defined in the Metamodel Layer (M2). Different language engineering environments exist for the Metametamodel Layer (M3) based on metamodelling. Ecore [12] and MOF (MetaObject Facility) [15] are examples of metamodelling-based environments. MOF is an object-oriented modelling language which offers concepts such as classes, primitive types, enumerations, generalization, attributes, associations, and operations for the definition of software languages. Ecore is the metametamodel language of the Eclipse Modeling Frameworks (EMF). It is a java-based implementation of MOF. It facilitates automatic generation of different implementations such as Java code, XML documents, and XML Schemata.

Context-Free Grammars [13] are an alternative language engineering environment to metamodelling. BNF (Backus-Naur form) [1] and EBNF (Extended Backus-Naur form) [41] are examples of context-free grammars which allow the specification of textual languages. These are used to describe the abstract and the concrete syntax at the same time but focus on the concrete syntax. Elements of a language are defined with production rules in context-free grammars. Context-free grammars are a tree-based representation as cyclic relationships are not expressed explicitly with these grammars. Instead, a string is defined which implicitly identifies the relationship with the desired production rule instead of an actual reference. Metamodelling, on the other hand, clearly separates the abstract syntax from the concrete syntax. Furthermore, metamodelling is a graph-based approach which allows cyclic relationships to be expressed explicitly. Given the advantages of metamodelling, we decided to use Xtext as our language engineering environment for HVT. Xtext combines metamodelling with BNF specifications for the concrete syntax, which makes explicit the desired metamodel and leverages the concept of cross reference to effectively model graphs, as discussed in the next section.

2.5 Xtext

Xtext [42] is an open-source framework used to create a textual domain-specific language (DSL). It is used to generate a metamodel, parser, and editor from the grammar definition. The grammar definition for Xtext is similar to BNF [1] but combines it with metamodelling by integrating type rules for the definition of metamodel elements. These rules contain terminals, assignments, and cross references.

To better understand an Xtext grammar definition, let us take an example statement to model: A shelf has multiple books written by authors and published by publishers, and each book has multiple chapters. Each book is of one genre which is of type fiction, drama, or poetry. Each book has one or more authors associated with it and has one publisher. Some of the books are rated as top sellers. Each chapter has a title and a number of pages. The grammar definition for this example is shown in Listing 2.1.

Each Xtext grammar starts with a header that defines the properties of the grammar. The first line declares the name of the language. The *with* specification represents an existing grammar provided by Xtext to inherit and use predefined rules. For example, line 1 in Listing 2.1 indicates that the name of the grammar is Bookshelf and that the grammar uses `org.eclipse.xtext.common.Terminals` which introduces the common set of rules.

Each rule is defined using a series of terminals and non-terminals and is converted to a class in the metamodel. Here terminals are considered as the quoted strings and non-terminals are considered as the names of the other rules. For example, ‘Book’, ‘<’, ‘popular’, ‘:’, ‘by’, ‘,’, ’published’, ’by’, and ’>’ in line 11 are terminals and ID, Genre, Author, Publisher, and Chapter are the non-terminals. The name of the rule is used as the name of the class. For example, rules Shelf, Book, Chapter, Author, and Publisher in lines 5, 10, 16, 19, and 22, respectively are converted to the classes Shelf, Book, Chapter, Author, and Publisher, respectively, in the metamodel.

Listing 2.1: Grammar Definition in Xtext

```
1  grammar org.xtext.example.bookshelf.Bookshelf with org.eclipse.xtext.common.Terminals
2
3  generate bookshelf "http://www.xtext.org/example/bookshelf/Bookshelf"
4
5  Shelf:
6    (authors+=Author)*
7    (publishers+=Publisher)*
8    (books+=Book)*;
9
10 Book:
11   ’Book’ ’<’ name=ID topSeller ?= ’popular’ genre=Genre ’:’ ’by’ (authors+=[Author] (’,’ authors+=[
       Author])*)? ’published’ ’by’ (publisher=[Publisher] ’>’ ’{’ (chapters+=Chapter (’,’ chapters+=
       Chapter)*)? ’}’;
12
13 enum Genre:
14   FICTION=’fiction’ | DRAMA=’drama’ | POETRY=’poetry’;
15
16 Chapter:
17   title=ID ’#’ numPages=INT;
18
19 Author:
20   ’Author’ name=ID;
21
22 Publisher:
23   ’Publisher’ name=ID;
```

An enumeration rule is used to map strings to enumeration literals. For example, in line 13, the datatype rule Genre is defined with alternative values (Fiction, Poetry, Drama) as literals using vertical bar (|). This is converted into the Enumeration named as Genre with literals as Fiction, Poetry, Drama in the metamodel.

For assignment operators, the = sign is used for a feature of the current class (rule) that accepts only one element, the += sign is used for a multi-valued feature which adds a list of elements for the right hand side, and the ?= sign is used for a Boolean feature. For example, in line 12, a single Publisher is assigned (=) to a feature called publisher, i.e., multiplicity 0..1 in the metamodel. In line 8, multiple Books are assigned (+=) to a feature called books, i.e., multiplicity 0..* in the metamodel. Similarly, in line 11 topSeller represents a Boolean feature.

A cross reference to a rule is indicated by square brackets on the right-hand side during the assignment to a feature. For example, the feature named publisher in line 11 is a cross reference (note the square brackets).

Xtext is using Extended Backus-Naur Form (EBNF) to define the different repetitions for features. There are four possible repetitions: if no operator is used it is considered as exactly one, the ? operator is used to indicate the repetition to be zero or one, the * operator sets the repetition to be zero or more, and lastly the + operator sets the repetition to be one or more. For example, in lines 6, 7, and 8, Shelf contains an arbitrary number (*) of books, publishers, and authors. If the * for the books in a Shelf were to be replaced with +, then at least one book would have to be specified. However, ? used for authors and chapters in a Book indicates a repetition of zero or one (i.e., authors and chapters are optional). The features name and genre of Book, on the other hand, are mandatory as no repetition operator is used.

A feature of the current class (rule) is converted to an attribute in the metamodel when the right-hand side indicates an enum or datatype rule or an inherited rule such as ID. A feature is converted to a reference in the metamodel when the right-hand side indicates another rule. If the reference to another rule is using square brackets, then an association is used in the metamodel whereas if it is not, then a composition is used. For example in line 17, title and numPages are features converted to attributes for the class Chapter in the metamodel. To define these attributes, we have title = ID. Here, ID is the standard identifier specified in the parent grammar Terminals (see first line). The left-hand side refers to the feature name (title) of the current class (Chapter). The right-hand side is an EString in this case. Similarly, we have numPages = EInt. Moreover, the feature called name in the Book, Author, and Publisher rules is converted to an attribute in the metamodel. On the other hand, the reference between Book and Publisher is a regular association because square brackets are used for Publisher in the Xtext rule for Book. The reference between Shelf and Book is a composition, because square brackets are not used for Book in the Xtext rule for Shelf. Figure 2.7 shows the elements of the metamodel resulting from the Xtext grammar, i.e., the abstract syntax which defines the concepts of the language and their relationships among themselves.

The grammar also specifies the concrete syntax by using the keywords which makes it easier to read and understand a textual specification. For example, the rule Book starts with the keyword

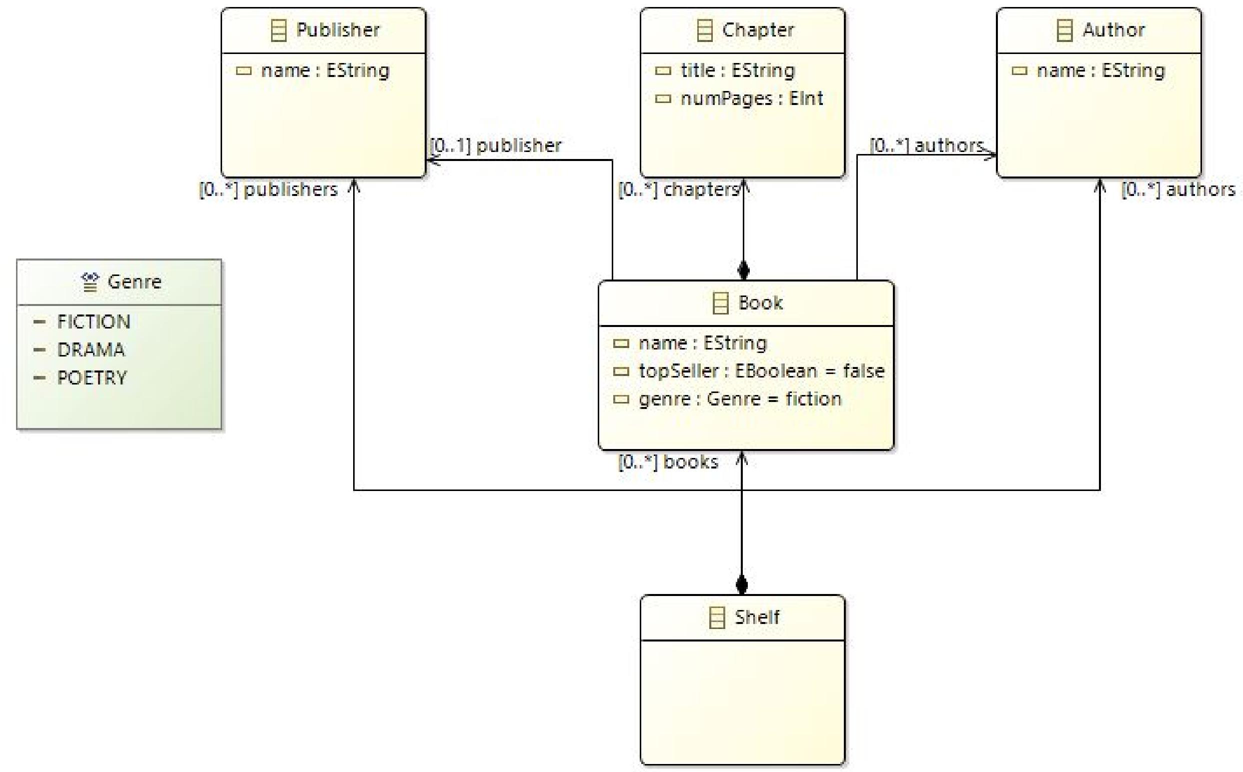


Figure 2.7: Generated Metamodel from Xtext Grammar

‘Book’ and then uses ‘:’ and ‘by’ to differentiate Genre and Authors, and ‘published’, ‘by’ is used to differentiate the Authors and the Publisher. In the same rule, Chapters and Authors are comma separated. Moreover, for the rule of enum Genre, it is possible to use alternative literals for enumerations like ‘fiction’ for FICTION. Also, in the rule Chapter, the name and the numPages are separated by ‘#’. Furthermore, the rule for Author and Publisher includes keywords ‘Author’, and ‘Publisher’, respectively, to make the textual specification clearer. Listing 2.2 shows an example model that conforms to the specified grammar.

Listing 2.2: Example Model Conforming to Specified Grammar

```
Author FredrikBackman
Publisher SimonSchuster
Book < AManCalledOve popular drama : by FredrikBackman published by SimonSchuster > { chapter1#4,
    chapter2#8 }
```

We have used Xtext to create the domain-specific language referred as Human Value Trigger (HVT). A detailed description of the HVT metamodel is explained in Chapter 4.

2.6 Summary

In this chapter, we discuss the technologies used in this thesis. It explains Schwartz's taxonomy of human values, the concept of domain model, the TouchCORE modelling framework, the concept of metamodel, and the Xtext language in detail.

The next chapter provides an analysis of an example domain model to motivate our approach to include human values in domain modelling.

3 Motivating Example

This chapter investigates an example system to motivate our approach to include human values-based elements in domain modelling. We examine the domain model for the WhatsApp System considering the human values in Schwartz's taxonomy.

3.1 WhatsApp

WhatsApp is a chat application that provides instant messaging and calling services to its user. Figure 3.1 exhibits all the classes, attributes, and relationships for the domain model of the WhatsApp system. For this example, we followed the process suggested for addressing human values in a domain model [25]. The initial step involves identifying the classes, then related attributes, associations, compositions, aggregation, association classes, and generalization are considered without taking human values explicitly into account. The next step includes the analysis of usage scenarios, i.e., we considered various features of WhatsApp such as privacy, document sharing, messaging privately, emojis, video, and voice calls. The final step performs human values analysis using Schwartz's theory to identify model elements and the impact on human values. This is an iterative process even though its description is rather sequential. The domain model under discussion is created by the authors based on their knowledge and experience gained from the use of WhatsApp.

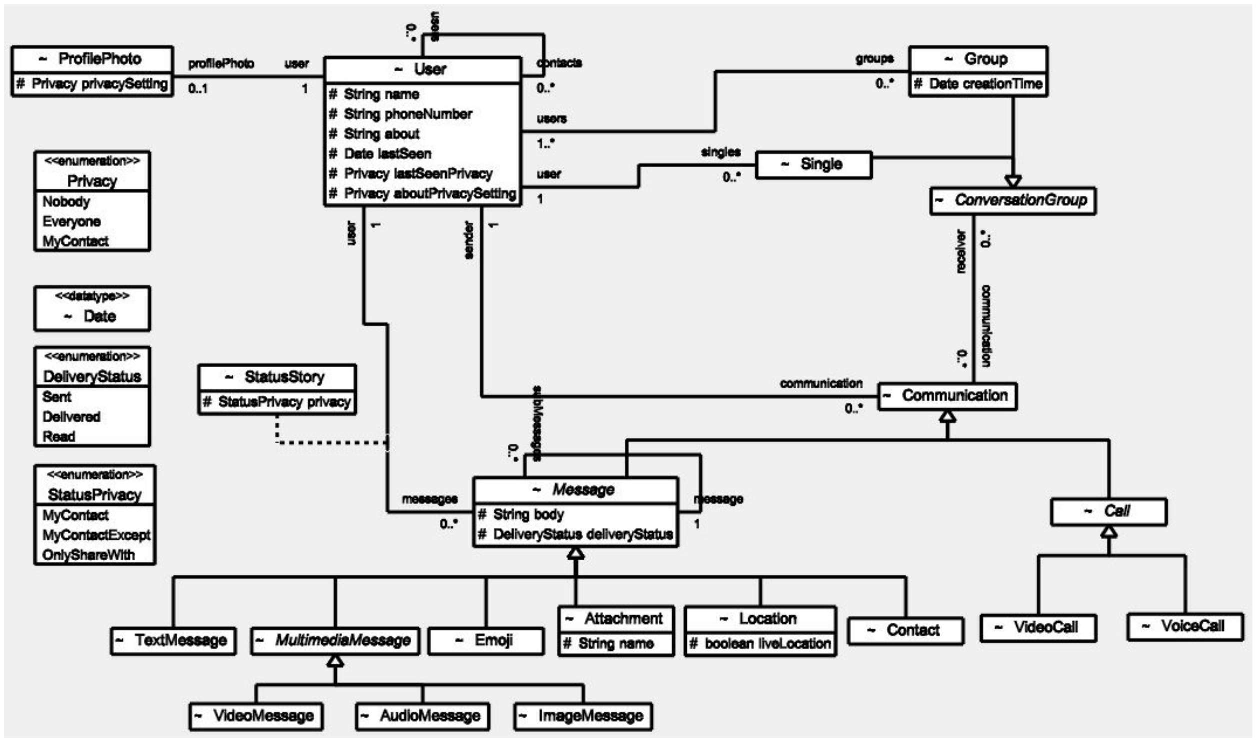


Figure 3.1: WhatsApp Domain Model

The domain model supports the functional requirements such as user registration, adding contacts, one to one chat, and group chat. Different types of communication between the users are covered such as text messages, voice calls, and video calls but also multi-media messages, emojis, attachments, location information, and contact information. Privacy settings for the profile photo and the status stories shared by the user on his profile are captured. After scrutinizing the domain model in terms of how different model elements interact with human values, we have come up with scenarios where essential human values were disregarded. These scenarios are covered in the following discussion.

i) For the Group class, various WhatsApp groups exist and anyone knowing of their existence can share information. So, misinformation could potentially be shared by members from one group to another which leads to the circulation of rumors sometimes and ultimately may cause disruption in the society or even a crisis in the society. For example, elections may be manipulated by conveying false information in the society. Nothing in this domain model indicates the verification of information being shared again and again. Moreover, end-to-end encryption makes shared messages immune to third parties, so it is difficult to trace back to the origin. The absence of model elements to address these issues may lead to violation of values like Privacy, Freedom, and many more as defined by Schwartz's taxonomy. In total, we identify 19 values that may be impacted. A complete list of values affected by the different scenarios is shown in Table 3.1 where we systematically analyze the impact on the 58 human values compiled by Schwartz. To prevent this breach, we could have added the flag attribute in the Message Class which can then be used as a poll to validate the information according to the opinion shared by different people.

ii) For the ProfilePhoto class, we have only options like Everyone, My Contacts, and Nobody to secure the privacy of the profile picture. But sometimes people need to save a random contact number to have one-time contact with another person. There is a minute possibility that the profile photo can be saved, e.g., by taking the screenshot which can be misused. This could lead to the compromise of values such as Privacy, Self-Respect, and many more. In total, we identify 7 values that may be impacted. To avoid this situation, we could have added a custom based option for selecting the contacts with whom the person wants to share the photo.

iii) Another possible scenario is that if a random person somehow has the contact number of another person, then that random person can send messages to the other person. While the receiver has the option to block the sender after receiving the message, that message can have an impact on the receiver in various ways depending upon the type of information being shared in the message. This has detrimental impact on values like Privacy, Pleasure, and many more. In total, we identify 6 values that may be impacted. Nothing in this domain model points to something that could have avoided this state. As a precautionary measure, we could have the option to create a whitelist of contact persons as a protection step.

iv) People get addicted to WhatsApp, and they do the same thing repeatedly which results in reduced (or no) interaction with other people. This negatively affects values like Creativity, Healthy, and many more. In total, we identify 16 values that may be impacted. To eliminate this possibility, we could have added an attribute in the model for which a user can enter the value as a time a person wants to spend on the app and the user gets a notification when the entered time is over.

v) Lastly, currently there is no way for people with disabilities to use the WhatsApp especially if the user is visually impaired. This impacts values like Freedom, Choosing your own goal, and many more. In total, we identify 9 values that may be impacted. To overcome this shortcoming, an attribute such as virtualAssistantIsRequired could be added in the User class.

Table 3.1: List of Values Affected by Five Different Scenarios for WhatsApp Model

	Misinformation is being shared	**Profile Picture is being shared**	**Messages from unknown user**	**Addiction**	**People with disabilities**
Self-Direction					
Creativity				X	X
Privacy	X	X	X		
Curious				X	
Freedom	X				X
Choosing own goal	X				X
Independent					
Stimulation					
Daring					
A varied life				X	
An exciting life					X
Hedonism					
Pleasure			X		
Enjoying life	X		X		X
Self-indulgent				X	
Achievement					
Successful	X				
Capable					
Ambitious					
Influential	X				
Intelligent					
Self-respect		X	X		
Power					
Social power	X				
Authority	X				X
Wealth					

Preserving my public image	X	X	X		
Social recognition		X			
Security					
Healthy	X	X		X	
Family security	X				
Social order	X				
Clean					
Reciprocation of favours					
National security	X				
Sense of belonging					X
Conformity					
Self-discipline	X			X	
Politeness	X			X	
Honoring of elders				X	
Obedient				X	
Tradition					
Humble				X	
Detachment					
Devout					
Respect for tradition				X	
Moderate				X	
Accepting my portion in life					
Benevolence					
Helpful					
Honest	X				
Forgiving				X	
Loyal				X	
Responsible	X				
True friendship				X	
Mature love				X	
Meaning of life					

A spiritual life		X	X		
Universalism					
Equality for all					X
Protecting the environment					
Inner harmony					
A world of beauty					
A world of peace	X				
Unity with nature					
Broad minded/tolerance	X				
Social justice		X			X
Wisdom					

Based on the motivating example, we can see that value assessment is challenging, and it is easy to overlook certain circumstances unless the practitioner exhibits a profound understanding of the domain. Even with deep knowledge of human value issues, it is still possible that the potential concerns may have been discussed during earlier phases in the software development life cycle but may not have been thoroughly documented or followed up on. Therefore, we observe the need to capture these issues from the past experiences so that when somebody builds a new system and encounters a similar situation, they may use this information to help them to be aware of the implication of human values they might have in the system. So, the goal is to build a tool that flags potential human value violations. This requires us to build a metamodel to collect past experiences by capturing different examples for each human value and model element type. Based on this, we formalize a DSL that can express such detailed scenarios, the human values impacted by the scenario, and the involved model elements.

3.2 Summary

This chapter includes an example domain model (i.e., WhatsApp) to motivate our approach used in HVT. An analysis of the example is performed which shows the need to capture human value issues in domain modelling.

In the next chapter, we present a detailed explanation of the metamodel that specifies the HVT.

4

Metamodel

This chapter discusses the HVT metamodel in detail and explains the use of the Xtext language to define the grammar for the proposed domain-specific language HVT.

4.1 Human Value Trigger (HVT) Metamodel

The Human Value Trigger (HVT) metamodel as shown in Figure 4.1 defines the human values as well as the suggestions provided in response to the detection of a potential human value issue in the analyzed domain model. It contains the main classes *HumanValue, Suggestion,* and *Trigger*. The HumanValue class refers to the information about the 58 human values, their definition, and respective categories according to the Schwartz theory, e.g., for the *Creativity* human value, *Self-Direction* is its category, and *Able to create new and original ideas* is considered as the concise definition explaining the value itself.

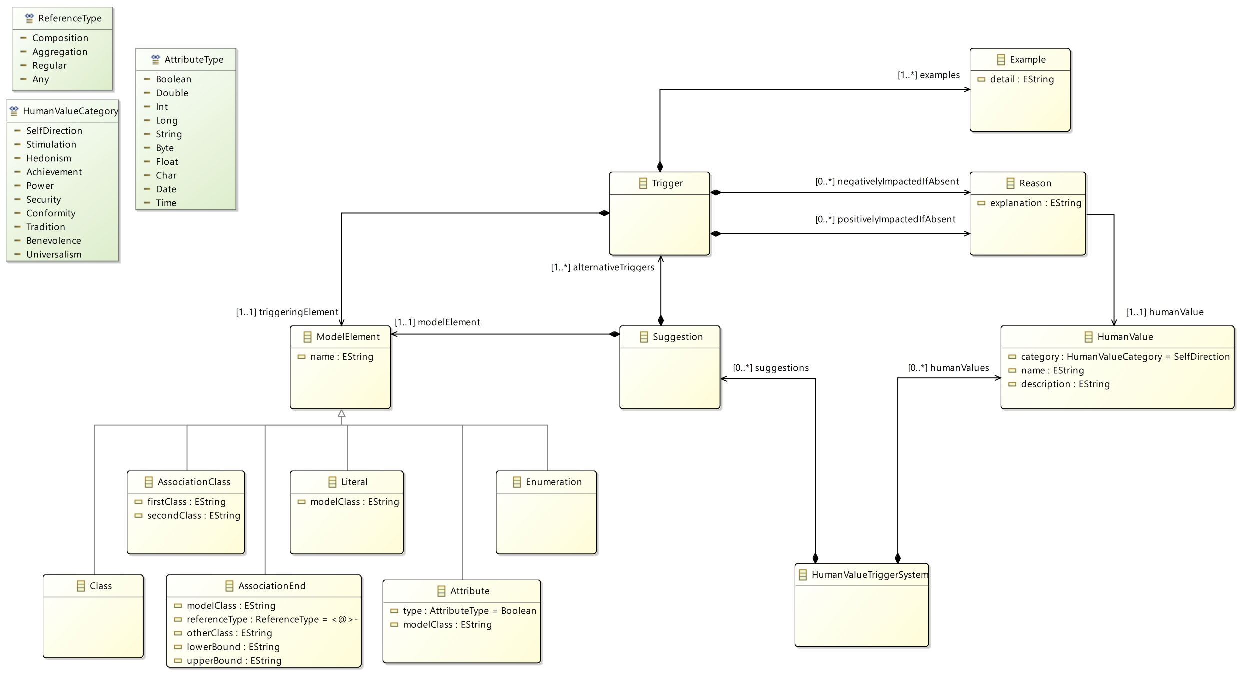


Figure 4.1: Human Value Trigger Metamodel

The *Suggestion* class contains the different triggers collected based on past experiences and the suggested elements that could be added to prevent an impact on various human values. Each *Suggestion* has one[1] modelElement as a suggested element that could be added in the analyzed domain model. For example, in scenario (iv) discussed in the motivating example in Chapter 3, *"timeYouWantToSpend"* is a suggested attribute for the User Class to avoid a similar situation. Each *Suggestion* can have multiple alternative triggers that each could lead to that scenario.

The *Trigger* class is defined to capture various cases and their significant impact on human values. Each *Trigger* has one triggering element which corresponds to the *ModelElement* which is being matched in the analyzed domain model. A *ModelElement* correspond to different elements in the class diagram, i.e., either a Class, Attribute, Enumeration, Literal, AssociationClass, or AssociationEnd. For example, in scenario (iv), the "User" class of the WhatsApp domain model is considered as the triggering element which Corresponds to the *Class* subclass of *ModelElement.*

Each *Trigger* also contains multiple examples and their corresponding reasons on how presence or absence of the element impacts different human values. Each *Example* refers to a detailed explanation for the Trigger. Here *"People get addicted to WhatsApp, and they do the same thing*

[1]Ecore/EMF does not enforce automatically mandatory associations (multiplicity greater than 0) but rather provides validation support to ensure that a model conforms to its metamodel

repeatedly which results in reduced (or no) interaction with other people" is considered as the example for this scenario. Each *Reason* contains the explanation and the positive and negative impacts on the corresponding value if the element is absent in the analyzed domain model. For the same example, *"People spend more time facing health issues like migraine problem"* is referred as a justification for the Reason and *"Healthy"* corresponds to the name of the *HumanValue* which is impacted negatively when the element is not in the analyzed domain model.

We decide to restrict ourselves to one *triggeringElement* for a Trigger as well as one *modelElement* for a Suggestion as the sample scenarios we investigated can be modelled with this approach. By restricting ourselves to one triggering element, we increase the chances of matches in the analyzed domain model compared to more complex patterns that could be matched. This increases the exposure of potential human value issues to the modeller with the trade-off that more false positives may be presented to the modeller. In the case where we need multiple model elements for the same Suggestion, a duplicate of the Suggestion could be created for each additional required model element. In future work, the multiplicity could be changed from 1 to 1-to-many.

Each *ModelElement* conforms to the TouchCORE Class Diagram Metamodel (CDM). This means that *ModelElement* could point directly to an element in the CDM, instead of specifying different model elements in the HVT metamodel. However, a complete class diagram would be required if we were to point directly to a CDM model element as a single CDM model element cannot exist in isolation. Furthermore, there are many more features defined for CDM model elements that are of limited use for our purpose but would have to be specified. In our HVT metamodel, we can focus on those elements that we actually need, making that clearer and explicitly defined. For example, if we want to express an AssociationEnd using the CDM metamodel, we have to specify additional attributes like navigable and ordered and additional classes like Association conforming to the CDM definition of *"AssociationEnd"*. Further with addition of these attributes, we have to discover a method to capture the elements being matched and the ones which are not being matched for the analyzed domain model. Including these elements directly in HVT provides a clear view of the exact pattern utilized for matching and helps maintain only the information required for matching.

The HVT metamodel discussed in Figure 4.1 is equivalent to the grammar definition specified with Xtext as shown in Appendix A. This grammar shows the metamodel and the concrete syntax of a domain-specific language which is used by a modeller to produce or read HVT models to describe potential human value issues. We use this language as it helps the modeller express the information in a natural and textual way. Additionally, the editor generated by Xtext from the grammar specification features auto-completion, syntax highlighting, and syntactic validation which helps with the correctness of the content. Listing 4.1 shows what the motivating example discussed previously looks like as an HVT model.

The grammar in Appendix A specifies three main features *HumanValue, Suggestion,* and *Trigger*. The rule for HumanValue starts with the keyword "HumanValue" followed by the feature named as category which is separated from the identifier name using the "." as shown in the concrete example in Listing 4.1. These keywords are used to specify what is anticipated next. In Suggestion, there can be one modelElement and a number of Triggers which are added to a feature called alternativeTriggers. In *ModelElement*, we use a vertical bar (|) to show the alternatives for the elements. For each model element, we use a keyword for the specific type for clear understanding of the instance created, e.g., an Attribute is identified by the keyword "Attribute".

Attribute:

'Attribute' type=AttributeType modelClass=STRING '.' name=ID

Similarly, to represent a *Trigger*, we use the keyword 'Trigger' which acts as a syntactic declaration. If we look at the model in Listing 4.1, we have a pattern to match which is "Class User". Here, User will be matched to suggest possible recommendations for the analyzed domain model. If a match exists and the modeller agrees, the suggestion "Attribute Time 'User'.timeYouWantToSpend" will be added to the analyzed domain model. Similarly, one or more examples from past experience are defined using the grammar. Then, one or more *Reasons* list out the impacted values and their respective explanations in the format *"negIfAbsent Creativity because People are addicted to the social media so they keep on doing the same thing every day, which kills creativity"*. Here, "negIfAbsent" and "because" represent the keywords to indicate the properties they define. Similarly, "posIfAbsent" is used to represent the postive impact on values if the model element does not exist in the analyzed domain model. Capturing negative and positive impact allows for trade-off analyses. Practitioners may utilize this language to contribute more such suggestions, triggers, examples, and reasons.

Listing 4.1: Scenario (iv) as an HVT Model

```
HumanValue Universalism.BroadMindedOrTolerance 'Liberal in views and reactions'
HumanValue Universalism.SocialJustice 'Everyone deserves equal economic, political, and social
    rights and opportunities'
HumanValue Universalism.Wisdom 'The quality of having experience, knowledge, and good judgment'
//Apart from the above mentioned values, there are a total of 58 values defined according to
    Schwartz's taxonomy of human values (not all shown for brevity).

Attribute Time 'User'.timeYouWantToSpend
Trigger Class User
        Example 'People get addicted to WhatsApp, and they do the same thing repeatedly which
    results in reduced (or no) interaction with other people'
        negIfAbsent Creativity because 'People are addicted to social media so they keep on doing
    the same thing every day, which kills creativity'
        negIfAbsent Curious because 'People get addicted to chatting so they do not study news or
    current topics or study their course.'
        negIfAbsent AVariedLife because 'People are addicted to WhatsApp and doing the same thing
    over and over again.'
        negIfAbsent SelfIndulgent  because 'People are addicted to WhatsApp not knowing which
    person is sitting next to them.'
        negIfAbsent Healthy because 'People spend more time facing health issues like migraine
    problem.'
        negIfAbsent SelfDiscipline because 'People spend more time on WhatsApp due to which people
    forget to do important work.'
        negIfAbsent Politeness because 'There is a change in style of conversation; basically
    WhatsApp changed the way people talk mostly on instant messages or on call'
        negIfAbsent HonoringOfElders because 'WhatsApp changed the way people talk mostly on
    instant messages or on call so people forget the right way.'
        negIfAbsent Obedient  because 'Due to constant usage people tend to lose track of time.'
        negIfAbsent Humble  because 'Due to excessive use there is a change of behavior among the
    users.'
        negIfAbsent RespectForTradition  because 'People are so involved in using the WhatsApp that
     they forget about the customs and tradition.'
        negIfAbsent Moderate  because 'Due to addiction people tend to go to extremes.'
        negIfAbsent Forgiving  because 'As people spend more time on WhatsApp the same things come
    up again and again.'
        negIfAbsent Loyal  because 'Due to addiction people tend to talk more than needed, which
    may lead to miscommunication among users.'
        negIfAbsent TrueFriendship  because 'Due to spending more time on WhatsApp people ignore
    their relationship with the people around them.'
        negIfAbsent MatureLove  because 'Spending more time on WhatsApp can impact relationships by
     decreasing the amount and quality of time people spend together.'
```

The human value trigger system proposed in this thesis compares every CDM element of the analyzed domain model with the triggers collected from past experiences. For example, if the analyzed domain model contains the class “User”, then all the triggers for class “User” from the collection are presented to the user. Based on the matched elements, the examples and the reasons for the impacted values are presented for each trigger to better understand the suggested element and the potential human value issue that needs to be considered. To optimize this matching process, we support the semantic detection of human value issues based on synonyms. The next section provides more detail about the process followed to accomplish this task.

4.2 Summary

This chapter explains the HVT metamodel, its purpose and use of its elements in detail. The concepts of suggestion, trigger, and human value are defined. Moreover, this chapter explains the use of the Xtext language to define the grammar for the proposed domain-specific language HVT.

In the next chapter, we discuss the details of the analysis performed on eight synonym services and various combinations of synonym services to find the best one to use as a synonym service in the envisioned tool.

5 Analysis of Synonym Services

This chapter provides a detailed explanation of the analysis performed on eight synonym services in Section 5.1 and the analysis performed on the various combinations of these synonym services in Section 5.2 to determine the best combination of synonym services for the HVTS.

5.1 Analysis of Single Synonym Service

To understand the requirement for the optimization of matching, let us consider an example domain model where the class “User” is referred as “EndUser” in their model as shown in Figure 5.1. When the proposed system is used, all the triggers matched for class User must be recommended for this analyzed domain model as User and EndUser are synonyms to each other. To accomplish this task, we have analyzed eight synonym services and various combinations of them to extract words with similar meaning. These synonym services include dictionaries and thesauri together with NLP-

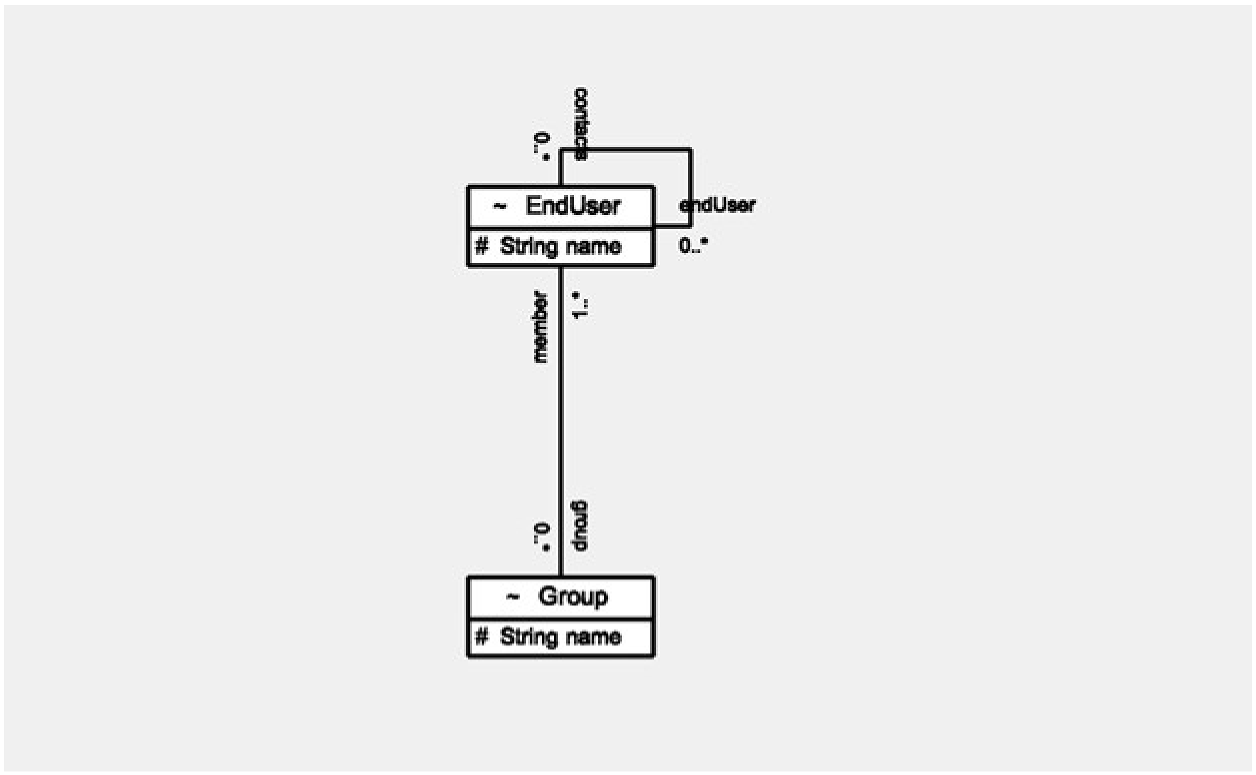


Figure 5.1: Example Domain Model

based and AI-based services. Table 5.1 depicts the list of different synonym services and links to the information to use various APIs to access these synonym services. All synonym services used in the analysis have a defined list of synonyms except for GloVe and ChatGPT. GloVe contains vector representation for words and compares the cosine similarity between the words to find the similar words. The cosine similarity measures the angle between two vectors. Given two word vectors, the cosine similarity score varies from -1 to 1, with a score of 1 indicating that the two words are identical, and a score of -1 indicating that they are completely dissimilar. The higher the similarity score between vectors, the more they are semantically similar [11] [28]. ChatGPT is a chatbot based on a large language model that uses deep neural networks to estimate the probability of the next word given a sequence of words. Also, some of the synonym services such as dictionary.com and thesaurus.com are not used in this thesis as they are not accessible by an API.

Based on our motivating examples, WhatsApp and the Airline Reservation System [25], we compile a list of words to compare the results from different synonym services. The words must be taken from an example domain model, because the appropriateness of a word suggested as a synonym by a synonym service depends on the context in which the word is used. The words used are *price, person, user, airline, delivered, status, text, message, active, privacy, story*, and *phoneNumber*. The list of words includes a combination of nouns like *price, person, user, airline,* verbs like *delivered,* and adjectives like *active.* We also include the word *"phoneNumber"* as the name of a model element is often a combination of more than one word. For such words, various synonym service APIs require different combinations such as "phone number", "phone-number", etc. to return the synonyms.

Table 5.1: List of Different Synonym Services and Information to use Various APIs

Synonym Service	**Link used to Access the Synonym Service**	**Part of API**
Wordnet	`https://projects.csail.mit.edu/jwi/` (User's Manual- edu.mit.jwi_2.4.0_manual.pdf)	Dictionary & Thesaurus
GloVe	`https://medium.com/analytics-vidhya/basics-of-using-pre-trained-glove-vectors-in-python-d38905f356db` `https://nlp.stanford.edu/projects/glove/`	Natural Language Processing
Word Association	`https://rapidapi.com/twinword/api/word-associations/`	Dictionary
Word Dictionary	`https://rapidapi.com/twinword/api/word-dictionary/`	Dictionary
Wordnik	`https://developer.wordnik.com/`	Dictionary
Oxford	`https://developer.oxforddictionaries.com/documentation`	Thesaurus
Webster	`https://dictionaryapi.com/`	Thesaurus
ChatGPT	`https://openai.com/blog/chatgpt`	Artificial Intelligence Chatbot

For each synonym service, we investigate the results based on the number of good and bad words returned as a synonym. The words are categorized as a good or a bad word as a synonym of the given word based on the interpretation done by the author of this thesis. Another researcher verified the interpretation, and any disagreements were discussed and resolved. After labeling the outcomes for different words, we formulate two criteria, to consider the best of these various synonym services:

Criterion 1: The average of the percentages of all good words *(Good words % = (Good words / (Good words + Bad words) *100)* for those words where at least one synonym is found. This is the precision of the result. We cannot calculate the recall of the result because the number of false negatives is not known in the absence of a gold truth.

Criterion 2: The average percentage of the count of words where at least one synonym is found *(Count % = Count words with synonym found / Count all words * 100).*

Table 5.2: Detailed Results for Wordnet

Word	**Good words**	**Bad words**	**Number of Good Words Found**	**Number of Bad Words Found**	**Percentage of Good Words Found**
price	cost, price, monetary value, toll	damage, terms	4	2	67%
person	individual, someone, somebody	soul, mortal	3	2	60%
user		exploiter	0	1	0%
airline			0	0	−−
delivered			0	0	−−
status	condition	position	1	1	50%
text	textual matter		1	0	0%
message	message	substance, subject matter, content	1	3	25%
active		active agent	0	1	0%
privacy	secrecy, privateness	seclusion, concealment	2	2	50%
story	narrative, write up, tale, narration, story, chronicle	level, taradiddle, history, storey, news report, report, floor, account, fib	6	9	40%
phone number			0	0	−−

For example, for the word "price" in Wordnet, the synonyms *"damage", "cost", "price", "terms", "monetary value",* and *"toll"* are found. Out of these, *"cost", "price","monetary value",* and *"toll"* are counted as good words while *"damage",* and *"term"* are counted as bad words. Similarly, for *"user"* we get *"exploiter"* from the synonym service which is considered as a bad word for our analysis. For words such as *"airline", "delivered",* and *"phone number"*, there are no results returned from this synonym service. Table 5.2 shows the results for the given list of words for Wordnet. Here, the percentage of good words found is calculated based on the number of good words found divided by the total number of words found for the word. As for the word *"price"*, the percentage is 67% and for *"user"* the percentage is 0% whereas for *"airline", "delivered",* and *"phone number"* the percentage is not considered. Then, we evaluated the values for Criterion 1 and Criterion 2 which are 32% (i.e., (67+60+0+50+0+25+0+50+40)/9) and 75% (i.e., 9/12*100), respectively. Further, we consider the range of good words by taking into account the lowest and the highest number of good words found which is 0-6 (*"airline"* − 0, *"story"* − 6). Lastly, we determined the average of good words by considering the number of good words found for all words which is 1.5 in this case (i.e., (4+3+0+0+0+1+1+1+0+2+6+0)/12).

Table 5.3: Results for all Synonym Services

Synonym Service	**Criterion 1**	**Criterion 2**	**Range of Good Words Found**	**Average of Good Words Found**
Wordnet	32%	75%	0-6	1.50
Glove	28%	92%	0-5	2.58
Word Association	8%	100%	0-4	2.41
Word Dictionary	**32%**	**100%**	**0-3**	**1.75**
Wordnik	8%	83%	0-3	0.66
Oxford	21%	67%	0-12	3.50
Webster	20%	83%	0-10	2.33
ChatGPT	31%	100%	1-13	5.75

Table 5.3 shows the results for the given list of words. As shown in the table, we obtain the maximum of 32% for Wordnet and Word Dictionary under Criterion 1, but Word Dictionary receives 100% under Criterion 2. For Word Association, we receive results for all words, but the percentage for Criterion 1 is lower compared to Wordnet and Word Dictionary. We obtain a high range for good words for Oxford with an average of 3.50. Despite these results, the values for Criterion 1 and 2 are lower when compared to Word Dictionary, because many false positives are also reported by Oxford. This is not ideal for an automatic approach we would like to integrate into HVTS. For Glove, we receive an average of 2.58 for good words but there is a decrease of 4% and 8% for Criterion 1 and Criterion 2 respectively when compared with Word Dictionary. Though, there is a decrease of 0.83 in average of good words for Word Dictionary compared to Glove, Word Dictionary performs better overall for an automatic approach. To conclude, the analysis above

shows that amongst the considered synonym services, Word Dictionary performs best overall. But to find the optimal synonym service for synonyms, we ideally want to choose the one that meets both selection criteria with a score of 100%. Another option would be to allow the modeller to specify the synonyms explicitly as done by Singh *et al.* [37] in their proposed mistake detection system. To implement this manual approach of providing synonyms, Oxford would be a good option because we could just present the long list from Oxford and the modeller could choose the appropriate synonyms. However, Oxford is not a good choice for our automatic approach.

Additionally, we investigate the results from ChatGPT. As shown in Table 5.3, ChatGPT performs well and the results look promising. But the results are not guaranteed, as the responses are generated by a large language model-powered chatbot that is known for varying its output. To support this assertion, we further explore ChatGPT to see the variance in the response by posing the same question five times in different runs (i.e., what are the synonyms for word "price"?). After comparing the results, we noticed a significant difference between the responses ranging from 53% to 90% with an average of 76%. This demonstrates that in some cases, the quality of responses may be very good and highly relevant, while in other cases, it may be mediocre. Therefore, due to the significant variance in the results from one run to another we exclude ChatGPT from further analysis. However, ChatGPT is a viable option for an interactive, manual approach and could even outperform Oxford in that case. One could also experiment with providing additional context together with the question to receive more accurate and relevant responses.

Therefore, to further investigate the synonym services with the aim to improve the outcomes for both criteria, we aggregate the results from different synonym services by selecting the common values for the synonym services under discussion. For example, the results for the word *"price"* for Wordnet are *"damage", "cost", "price", "terms", "monetary value",* and *"toll"* and for Word Dictionary *"discount", "charge", "cost", "toll",* and *"ticket"*. Consequently, the words *"cost"* and *"toll"* are considered as the result for the combination of these synonym services (Wordnet + Word Dictionary). The criterion used for the analysis of combinations of synonym services is explained in the next section.

Table 5.4: Results for the Pair-Wise Combination of Synonym Services

Synonym Service	**Criterion 1**	**Criterion 2**	**Range of Good Words Found**	**Average of Good Words Found**
Oxford + Word Dictionary	67%	50%	0-3	0.75
Oxford + Webster	37%	67%	0-7	1.75
Oxford + Wordnet	38%	42%	0-2	0.50
Oxford + GloVe	40%	42%	0-2	0.33
Word Dictionary + Webster	**89%**	**42%**	**0-2**	**0.66**
Word Dictionary + Wordnet	83%	42%	0-3	0.75
Word Dictionary + GloVe	50%	50%	0-2	0.33
Webster + Wordnet	26%	42%	0-4	0.50
Webster + GloVe	75%	33%	0-2	0.33
Wordnet + GloVe	89%	25%	0-2	0.33

5.2 Analysis of Combinations of Synonym Services

The selection criteria to combine the synonym services is based on the values obtained for Criterion 1. According to Table 5.3, we obtained more than 10% for Criterion 1 for five synonym services namely Wordnet, GloVe, Word Dictionary, Webster, and Oxford. Based on this, we decide to keep only these for further analysis as indicated in Table 5.4. We investigate each pair-wise combination of these five synonym services.

As indicated in the last two rows of Table 5.4, the results for Criterion 1 and Criterion 2 are lower when compared to other combinations (e.g., Word Dictionary + Webster). Therefore, we choose to disregard these two rows. The remaining combinations can be grouped into three groups based on Criterion 2: one row with 67%, two with 50%, and five with 42%. For each of those groups, we then choose the combination with the highest score in Criterion 1, as the best combination from that group, i.e., Word Dictionary + Webster for the combination of synonym services with 42% for Criterion 2, Oxford + Word Dictionary for the combinations with 50% as Criterion 2, and Oxford

+ Webster for the combinations with 67% as Criterion 2.

Table 5.5: Results for the Combination of Three Synonym Services

Synonym Service	**Criterion 1**	**Criterion 2**	**Range of Good Words Found**	**Average of Good Words Found**
Wordnet + GloVe + Word Dictionary	89%	25%	0-2	0.33
Wordnet + GloVe + Webster	83%	17%	0-2	0.25
Wordnet + GloVe + Oxford	67%	25%	0-2	0.25
Wordnet + Word Dictionary + Webster	**92%**	**33%**	**0-2**	**0.33**
Wordnet + Word Dictionary + Oxford	75%	33%	0-2	0.41
Wordnet + Webster + Oxford	45%	42%	0-2	0.41
GloVe + Webster + Oxford	67%	25%	0-2	0.25
GloVe + Webster + Word Dictionary	**100%**	**17%**	**0-2**	**0.25**
GloVe + Oxford + Word Dictionary	**100%**	**17%**	**0-2**	**0.25**
Oxford + Word Dictionary + Webster	**78%**	**50%**	**0-2**	**0.66**

When comparing the results from Word Dictionary + Webster with Oxford + Word Dictionary, Criterion 1 in the first combination is 22% more than the second combination while Criterion 2 in the first combination is 8% less than the second combination with a similar range and average of good words. Therefore, we move forward with the first combination (Word Dictionary + Webster).

Further, when we compare the results from Word Dictionary + Webster with Oxford + Webster, it is very clear from Table 5.4 that Criterion 1 is 52% more for the first combination. Consequently, Word Dictionary + Webster is the best choice from Table 5.4, even though Criterion 2 of Oxford + Webster is 25% more than Word Dictionary + Webster. However, a high result for Criterion 1 is more important for our desired automated approach as the number of incorrect synonyms is minimized.

We observe a similar tradeoff between the best choices from Table 5.3 and Table 5.4. Criterion 2 and the average of good words for the combined synonym service (Word Dictionary + Webster) has decreased by 58% and 1.09, respectively, when compared with the results from Word Dictionary. Now, in contrast to Criterion 2, Criterion 1 is 57% more in case of the combined synonym services than Word Dictionary alone. Although the decline in Criterion 1 causes us to miss some good synonyms, there is a significant increase in the quality of words we are receiving as a synonym. Based on this conclusion, we finalize the combined synonym services (Word Dictionary + Webster) as the synonym service for the proposed HVTS.

To assure us that a combination with three synonym services does not yield a better result, Table 5.5 reports the results of all combinations of three synonym services. The rows in bold font indicate those combinations where either Criterion 1 or Criterion 2 is better than the best choice from Table 5.4. While Criterion 1 reaches 100% now for two combinations, Criterion 2 drops to 17%, which we deem to be too low for the results to be useful for our proposed HVTS. The remaining two bold options from Table 5.5 are also discarded, because an increase in one criterion is accompanied by a larger decrease in the other criterion. Hence, Word Dictionary + Webster is indeed our combination of choice.

5.3 Summary

This chapter discusses the process followed for the analysis of eight synonym services and various combinations to find the optimal synonym service to use as a synonym service in the proposed HVTS.

In the next chapter, we discuss the details of our prototype tool that can be used to integrate human values in a domain model and explain how the metamodel discussed in Chapter 4 and the synonym services discussed in this chapter are used by the human value trigger algorithm in

HVTS to match possible human value issues based on past experience and integrate user-approved suggestions into the analyzed domain model. Moreover, it also provides insights on the verification process followed for HVTS.

6
Human Value Trigger Algorithm

This chapter discusses the algorithm that provides the suggestions to integrate the consideration of human values in domain modelling based on collected past experiences. The chapter first gives the overview of the algorithm in Section 6.1 and then explains in detail the data structures initialized by the algorithm in Section 6.2. In addition, the chapter covers the comparison between collected experiences and the analyzed domain model as well as the addition of suggested elements in the domain model in Section 6.3. Furthermore, Section 6.4 covers the verification process of the HVTS.

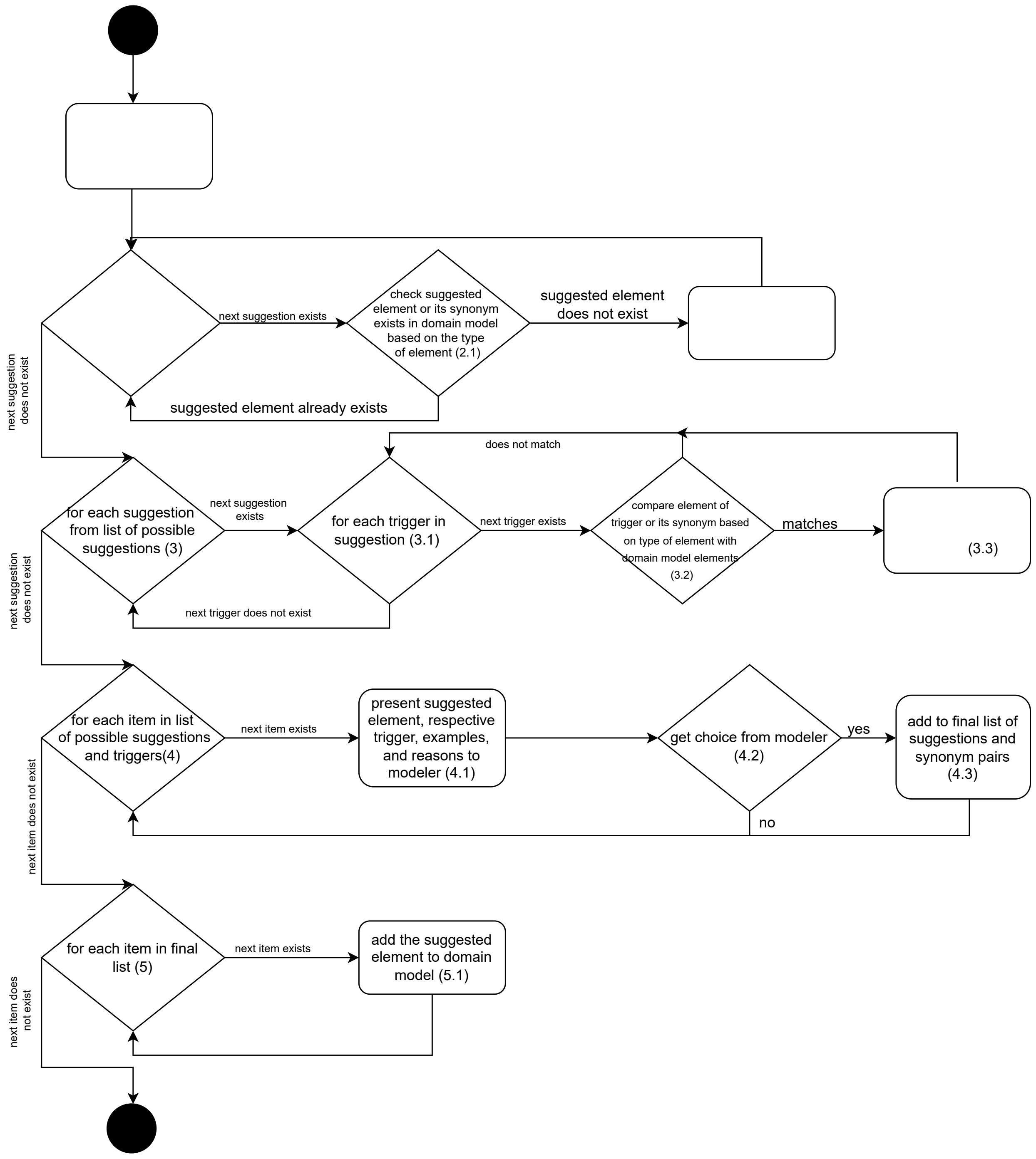
6.1 Algorithm Overview

The algorithm overview is shown in Figure 6.1. The algorithm takes two inputs, a model humanValueTriggerFile (HVTF) specified by the Xtext grammar as described in Chapter 4 and a CDM model domainModel (Action 1 in Figure 6.1). In general, the algorithm compares the model elements of the humanValueTriggerFile and the domainModel by matching them based on type, name similarity, and semantics. At the start, the algorithm compares the modelElement (see HVT Metamodel in Figure 4.1) of Suggestion (see Figure 4.1) in the HVTF with the domain model to validate the existence of these elements in the domain model (Action 2.1 in Figure 6.1). After comparing, if there are possible Suggestions which do not exist in the domain model, then the algorithm stores this information in a list of possible suggestions (Action 2.2). Then, the algorithm compares the triggeringElement (see Figure 4.1) of the alternativeTriggers (see Figure 4.1) from the list of possible suggestions with the elements of the domain model (Action 3.2) and adds any matches to the list of possible suggestions and triggers (Action 3.3). The next step in the algorithm is to present each possible Suggestion, its Trigger (from the alternativeTriggers in Figure 4.1), its Examples (from the Trigger's examples in Figure 4.1), and Reasons (see positivelyImpactedIfAbsent and negativelyImpactedIfAbsent in Figure 4.1) to (i) make the modeller aware of a potential human values issue and (ii) get the input from the modeller regarding the suggestion the modeller wants to integrate into the domain model (Actions 4.1 and 4.2). The algorithm then adds the "yes" choices of the modeller to the final list of suggestions to include in the domain model (Action 4.3). In the last stage, the algorithm integrates the selected suggestions into the domain model (Action 5.1).

6.2 Algorithm Initialization

In Action 1 of Figure 6.1, the algorithm initializes the data structures to capture the information while comparing elements of the *HVTF* and the *domain model*. It creates and initializes the hash maps for storing the suggestions and their respective synonyms, for storing the matched suggestions and their triggers, and for storing the complete set of synonyms for matched suggestions and triggers.

Whenever the algorithm finds a suggestion which does not exist in the domain model, it saves those suggestions and possible synonyms of the model elements of the suggestion in a hash map to keep track of the different possibilities (Action 2.2). We use the suggestion as key and the list of synonyms as its value as this helps us to easily navigate throughout the algorithm. Each item in the list of synonyms further stores information using a hash map with the model element as key



next suggestion exists
check suggested element or its synonym exists in domain model based on the type of element (2.1)
suggested element does not exist
suggested element already exists
next suggestion does not exist
does not match
for each suggestion from list of possible suggestions (3)
next suggestion exists
for each trigger in suggestion (3.1)
next trigger exists
compare element of trigger or its synonym based on type of element with domain model elements (3.2)
matches
(3.3)
next trigger does not exist
next suggestion does not exist
for each item in list of possible suggestions and triggers(4)
next item exists
present suggested element, respective trigger, examples, and reasons to modeler (4.1)
get choice from modeler (4.2)
yes
add to final list of suggestions and synonym pairs (4.3)
no
next item does not exist
for each item in final list (5)
next item exists
add the suggested element to domain model (5.1)
next item does not exist

Figure 6.1: Overview of Algorithm

and its synonym as the value.

In Action 3.3, the algorithm stores the matched triggers of a possible suggestion for the domain model using a hash map with the suggestion as key and the list of triggers as its value. We use the suggestion as the key as it is possible to have more than one trigger for a suggestion. This makes it easier to store the different triggers with the same suggestion without repetition of the suggestion.

In Action 3.3, the algorithm also stores the synonyms of the model elements of suggestions and the triggers using a hash map with suggestion and trigger as the key and the list of possible synonyms pairs as its value. The list of synonym pairs is implemented in the same way as described earlier. We use both suggestion and trigger as key, as it will help to find the exact match to use during the process of presenting the elements to the modeller.

In Action 4.3, the algorithm also stores the suggestion and the synonyms of the model elements of the suggestion for each suggestion that the modeller wants to integrate in the *domain model* using a hash map with suggestion as the key and an item from the list of possible synonym pair as its value. The item from the list of possible synonym pairs is implemented in the same way as described earlier. This makes it easier to add the suggestion in the domain model with the right synonym pair.

6.3 Algorithm Details

To better understand the algorithm let us again consider an example domain model where the class *"User"* is referred to as *"EndUser"* in their *domainModel* as shown in Figure 6.2 and compare it with the HVTF as shown in Listing 6.1. The algorithm begins by iterating through the suggestion in the HVTF file as shown in Line 1 in Listing 6.1 (Action 2 in Figure 6.1) and checks for the type of the element in the suggestion, in this case the AssociationEnd (see HVT Metamodel in Figure 4.1).

The algorithm then compares the elements of the AssociationEnd in Suggestion with all the AssociationEnds which exists in the *domainModel* (Action 2.1). To compare the elements of an AssociationEnd, the algorithm starts by comparing its corresponding classes which is *"User"* for both *modelClass* and *otherClass* (see Figure 4.1) with all the classes in the *domainModel* and checks for the existence of an instance of the *Class* (see Class Diagram Metamodel in Figure 2.5) with the

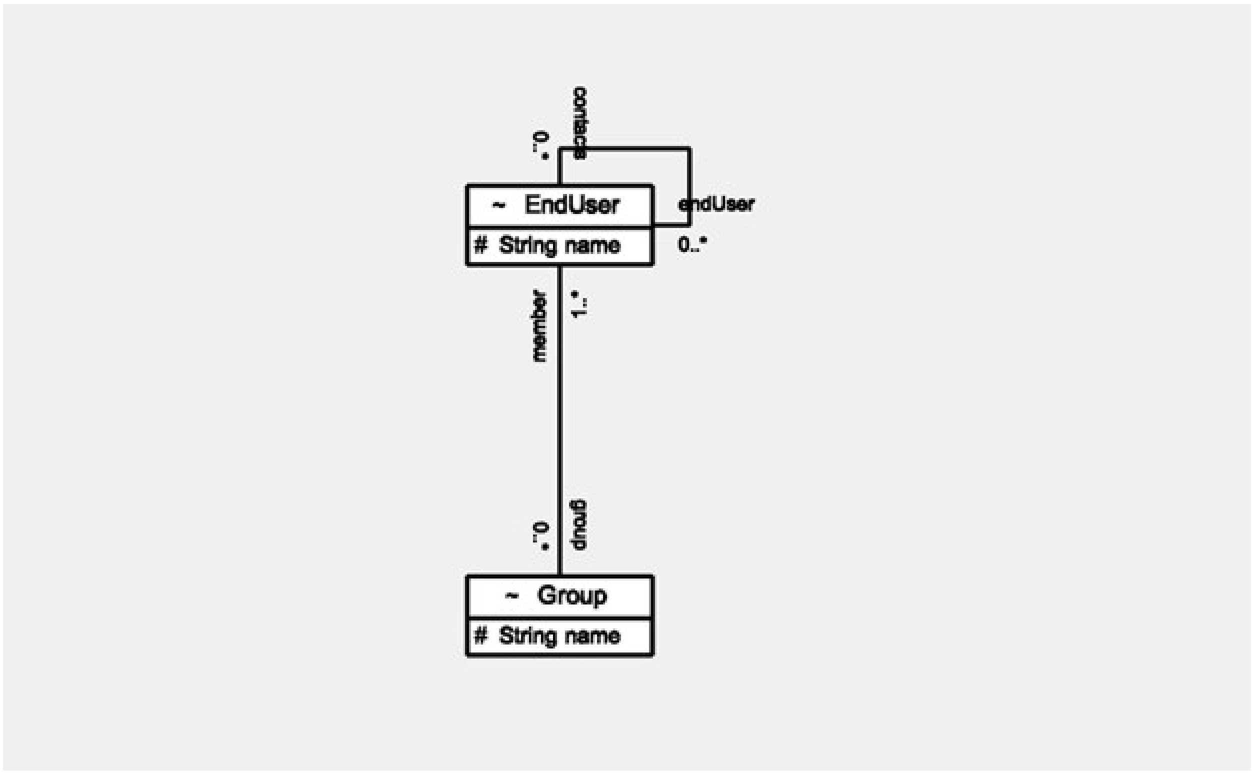


Figure 6.2: Example Domain Model

same name in the *domainModel.* If it does not exist, then the algorithm checks for its synonym and if the synonym service returns the synonyms for the *"User"*, then it further compares those results with the *domainModel.* In this case, the synonym service will return *"EndUser"* and after comparison the match is found as the Class *"EndUser"* exists in the *domainModel.* After confirming that both the classes associated to the AssociationEnd exists in the domain model, the algorithm then compares the AssociationEnd *"whitelistContacts"* with the AssociationEnds of the matched classes in the *domainModel.* Since, the association end with an exact match does not exist in the *domainModel*, then the algorithm further checks for the existence of the synonyms in *domainModel.* As there is no match found, the algorithm adds this suggestion to the list of possible suggestions (Action 2.2). As this suggestion does not exist in the *domainModel*, the algorithm also stores the information of the found synonyms of the elements in the hash map as discussed in Section 6.2.

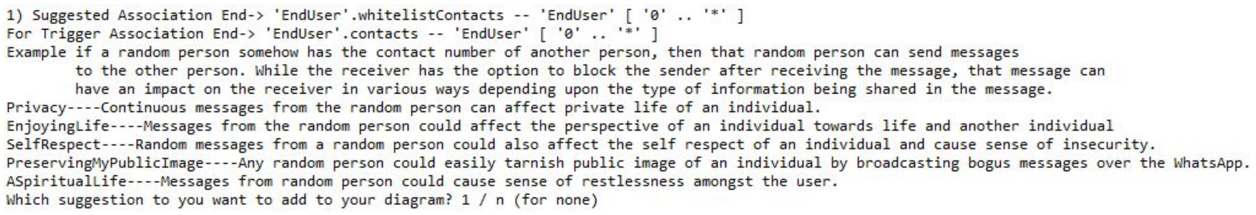
The algorithm iterates through the list of possible suggestions (Action 3). Then, the algorithm checks whether the trigger for the suggestion exists in the list of possible suggestions (Action 3.1). The algorithm then checks the type of the element of *Trigger* which is of type AssociationEnd as shown in Listing 6.1. The algorithm compares the AssociationEnd and its associated classes in the similar way as discussed in the above paragraph (Action 3.2). In this case, the match is found as the elements of the Trigger *"User"*, *"contacts"*, and *"User"* exist in the *domainModel*. Then, the algorithm adds the suggestion and its trigger to the list of possible suggestions and triggers and compiles the list of synonym pairs after combining the synonyms pairs which exist for the suggested element and the trigger (Action 3.2). While Loop 3 (i.e., Actions 3.1, 3.2, and 3.3) could be nested inside Loop 2, we retain this illustration to help make the phases more explicit.

Listing 6.1: Suggestion in Human Value Trigger File

```
AssociationEnd 'User'.whitelistContacts -- 'User' [ '0'..'*' ]
Trigger AssociationEnd 'User'.contacts -- 'User' [ '0'..'*' ]
       Example 'If a random person somehow has the contact number of another person, then that
    random person can send messages to the other person. While the receiver has the option to block
     the sender after receiving the message, that message can have an impact on the receiver in
    various ways depending upon the type of information being shared in the message.'
       negIfAbsent Privacy because 'Continuous messages from a random person can affect the
    private life of an individual.'
       negIfAbsent Pleasure because 'Messages from a random person can cause unnecessary stress
    which further impacts pleasure.'
       negIfAbsent EnjoyingLife because 'Messages from a random person could affect the
    perspective of an individual towards life and another individual.'
       negIfAbsent SelfRespect because 'Random messages from a random person could also affect the
     self respect of an individual and cause sense of insecurity.'
       negIfAbsent PreservingMyPublicImage because 'Any random person could easily tarnish the
    public image of an individual by broadcasting bogus messages over the WhatsApp.'
       negIfAbsent ASpiritualLife because 'Messages from a random person could cause sense of
    restlessness amongst the individual.'
```

The algorithm then displays these suggestions to the modeller. The algorithm iterates through the list of possible suggestions and triggers (Loop 4) and then uses the combination of suggestion and trigger to fetch the list of synonym pairs to be used which exist in the domain model. Then, the algorithm iterates over the list of synonym pairs and displays the suggested element, trigger, example, and the reasons (Action 4.1).

There are multiple ways to ask the consent of the modeller about integration of new suggestions in the domain model. One simple way is to show the recommended suggestions and ask the modeller *"Which suggestion do you want to add to your diagram?"* and providing the option to modeller for selecting, e.g., "1", "2", ..., or "n (for none)" (Action 4.2). In Figure 6.3, only one option exists, so "1/n (for none)" is shown. The modeller can then choose 1 or none if the new suggestions should be integrated in the model or not. Another option is to show the modeller what impact the suggestion could have on the domain model before adding the suggestions by displaying the elements in the



```
1) Suggested Association End-> 'EndUser'.whitelistContacts -- 'EndUser' [ '0' .. '*' ]
For Trigger Association End-> 'EndUser'.contacts -- 'EndUser' [ '0' .. '*' ]
Example if a random person somehow has the contact number of another person, then that random person can send messages
        to the other person. While the receiver has the option to block the sender after receiving the message, that message can
        have an impact on the receiver in various ways depending upon the type of information being shared in the message.
Privacy----Continuous messages from the random person can affect private life of an individual.
EnjoyingLife----Messages from the random person could affect the perspective of an individual towards life and another individual
SelfRespect----Random messages from a random person could also affect the self respect of an individual and cause sense of insecurity.
PreservingMyPublicImage----Any random person could easily tarnish public image of an individual by broadcasting bogus messages over the WhatsApp.
ASpiritualLife----Messages from random person could cause sense of restlessness amongst the user.
Which suggestion to you want to add to your diagram? 1 / n (for none)
```

Figure 6.3: Display the Suggested Element, Trigger, Examples, and the Reasons to Modeller

UI as shown in Figure 6.4 (Action 4.2). In this figure, the elements highlighted in red indicate the suggested element and the elements highlighted in green indicate the trigger. The pop-up provides the option to the modeller about integrating the suggestion in the domain model. Another pop-up option can be added to demonstrate the examples and reasons for the recommendation. After displaying the suggestions to the modeller, the algorithm then saves the combination of suggestion and respective synonym pair which the modeller wants to integrate in the domain model to the final list (Action 4.3).

The algorithm then iterates over the final list of suggestions to integrate them in the *domainModel* (Action 5). The algorithm takes the suggested element to add and checks the type of element of Suggestion, i.e., AssociationEnd in the example. The algorithm then fetches the name of the AssociationEnd and its associated classes and checks if that AssociationEnd exist in the domain model. This step of checking "if the element exists in the domain model" is done to make sure that the element that needs to be added by the current suggestion has not already been added by another suggestion. If it does not exist in domain model, the algorithm checks if the corresponding classes exists. If they exist, then the algorithm fetches the instances of those classes, and if they do not exist, then the algorithm creates the classes and adds them to the diagram. To create the class, the algorithm creates the instance of the class and then for the instance created, the default properties are added to conform with the CDM metamodel in Figure 2.5. The class is added to the list of classes in the domain model. In this case, the associated classes with the AssociationEnd which is *"EndUser"* already exists in the *domainModel*, so the algorithm fetches the instance from the domain model.

In the next step, the algorithm creates the AssociationEnd with name *'whitelistContacts'* and uses values of the *lowerBound, upperBound,* and *referenceType* (see HVT Metamodel in Figure 4.1) to set the properties to conform with the CDM metamodel. It then adds the association end to the

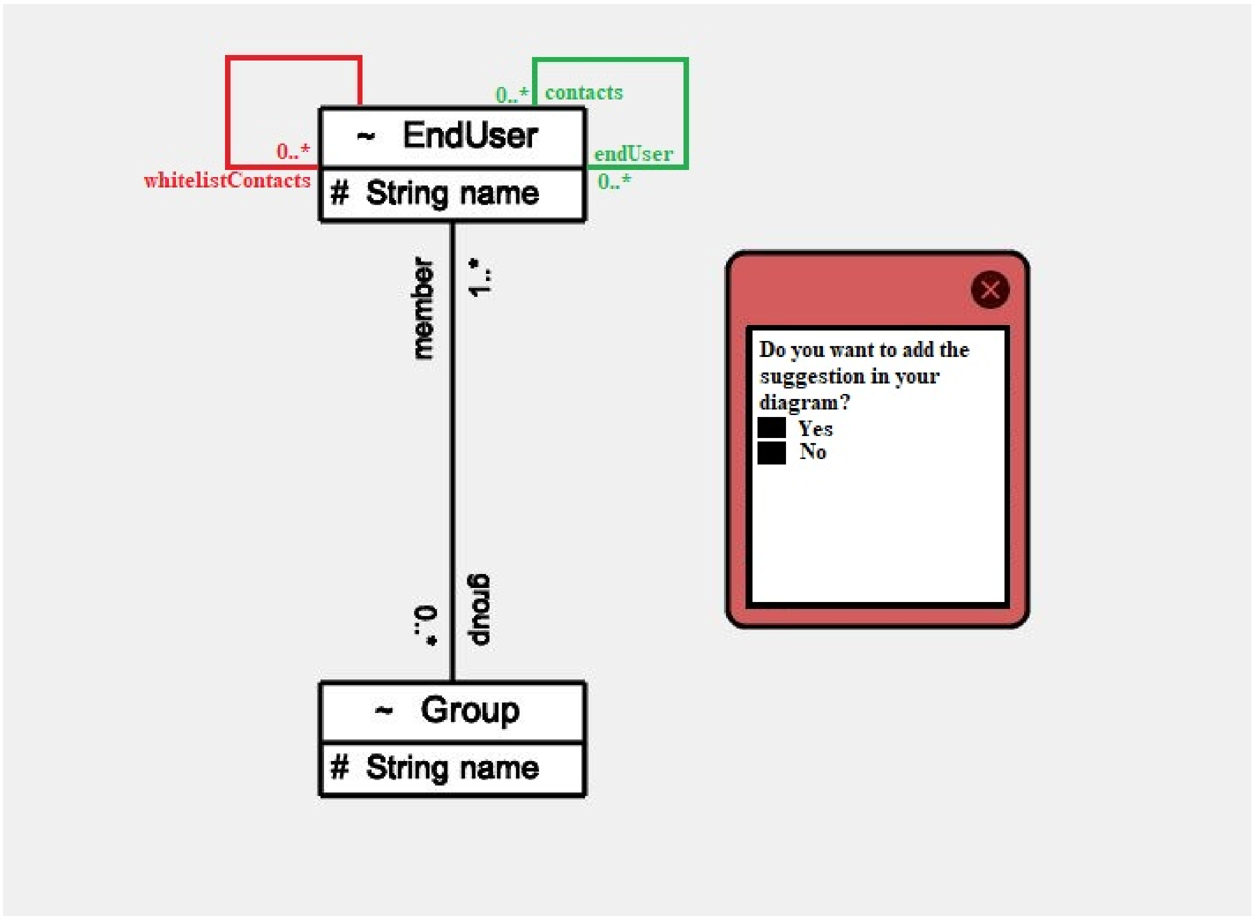


Figure 6.4: Visualize the Impact on Domain Model using UI

modelClass and then creates another association end with default values and adds the end for the *otherClass*. The algorithm then creates the Association for the two classes and adds it to the list of associations in the *domainModel*. Then in the last step, the algorithm links the Association with the two ends created. After adding all the modeller-selected suggestions to the domain model, the algorithm then saves the domain model at the end (Action 5.1).

6.4 Verification

To evaluate the capabilities of HVTS, we have performed an extensive testing process. These tests focus on testing the matching logic of the implemented Suggestion and Trigger in the HVT. To cover edge cases, multiple test cases are created for each type of ModelElement. For example, to test the matching logic for Class as a ModelElement, test cases cover the scenario where the

Table 6.1: Different Test Scenarios Covered for the Matching Logic of Suggestion and Trigger

ModelElement tested	**Different scenarios for matching existing element in domain model**		
Class	Class exists	Synonym of Class exists	
Attribute	(Attribute name, modelClass) exist	(Synonym of Attribute name, Synonym of modelClass) exist	(Attribute name, Synonym of modelClass) or (Synonym of Attribute name, modelClass) exist
Enumeration	Enumeration exists	Synonym of Enumeration exist	
Literal	(Literal name, modelClass) exist	(Synonym of Literal name, Synonym of modelClass) exist	(Literal name, Synonym of modelClass) or (Synonym of Literal name, modelClass) exist
AssociationClass	(AssociationClass name, firstClass, secondClass) exist	(Synonym of AssociationClass name, Synonym of firstClass, Synonym of secondClass) exist	Combinations (Synonym of AssociationClass name, Synonym of firstClass, Synonym of secondClass) or (Synonym of AssociationClass name, firstClass, secondClass) or so on exist
AssociationEnd	(AssociationEnd name, otherClass, modelClass) exist	(Synonym of AssociationEnd name, Synonym of modelClass, Synonym of otherClass) exist	Combinations (Synonym of AssociationEnd name, Synonym of modelClass, Synonym of otherClass) or (Synonym of AssociationClass name, modelClass, otherClass) or so on exist

Class with same name exists as an instance of the Class, or its synonym exists in the domain model. This also includes a test case where the Class matches with two synonyms in the domain model. Similarly, to test the matching logic for the Attribute, test cases cover the scenarios where the modelClass (see HVT Metamodel in Figure 4.1) and the name of the Attribute exist, or their synonyms exist in the domain model. This also includes the test case where the modelClass matches with two instances as a synonym in the domain model, but only one instance of Class contains the Attribute being matched. To test the matching logic for AssociationClass (see Figure 4.1), the test cases include the scenarios where the instance of firstClass (see Figure 4.1) and secondClass (see Figure 4.1) or their synonyms exist in the domain model. The test cases also cover the mix and match of the existence of the instance of the Class or its synonym. This also includes the test case where the AssociationClass as a Suggestion already exist in the domain model as an instance of the Class. To test the logic for AssociationEnd, test cases cover the scenarios where the instance of AssociationEnd, its modelClass, and otherClass exist or their synonyms exist in the domain model. Multiple test cases are created to cover the existence of the combination of the original element or its synonyms for all the elements being matched for the AssociationEnd. To test the Enumeration (see Figure 4.1) and Literal (see Figure 4.1), test cases cover the scenario where the instance of the Enumeration exists, or its synonym exist in the domain model. This also includes the test case where the Enumeration matches with two synonyms in the domain model. Similarly, to test the matching logic for the Literal, test cases cover the scenarios where the modelClass (see Figure 4.1) and the Literal exist, or their synonyms exist in the domain model. Table 6.1 shows the different scenarios covered during the testing of matching with different model elements. These scenarios are tested for both Suggestion and Trigger in the test HVTF.

To cover the structure used by the test cases and explain the assertions utilized in these tests, let us consider the example domain model shown in Figure 6.5 and compare it with the HVTF shown in Listing 6.2[2]. The HVTF contains the Class "Cost" as the Trigger. When we run this test case with the example domain model, HVTS detects the two instances of the Class, i.e., Price and Charge, hence passing the test case and verifying that it matches with the synonyms[3].

Listing 6.2: Human Value Trigger File (HVTF) for First Test Case

```
Class AuthorizedPayment
Trigger Class Cost
        Example 'Check for the authorized person to make payment'
        negIfAbsent Authority because 'Unauthorized users are not provided with any authorization
    over the data/info transfered.'
```

The code of the test case is shown in Listing 6.3. Every test case requires a .cdm file which is created using the TouchCORE tool and the .hvt file which is specified as an instance of HVT using

[2]For testing purposes only; does not necessarily reflect past experience.

[3]Used synonym service (Chapter 5) considers price and charge synonyms of cost

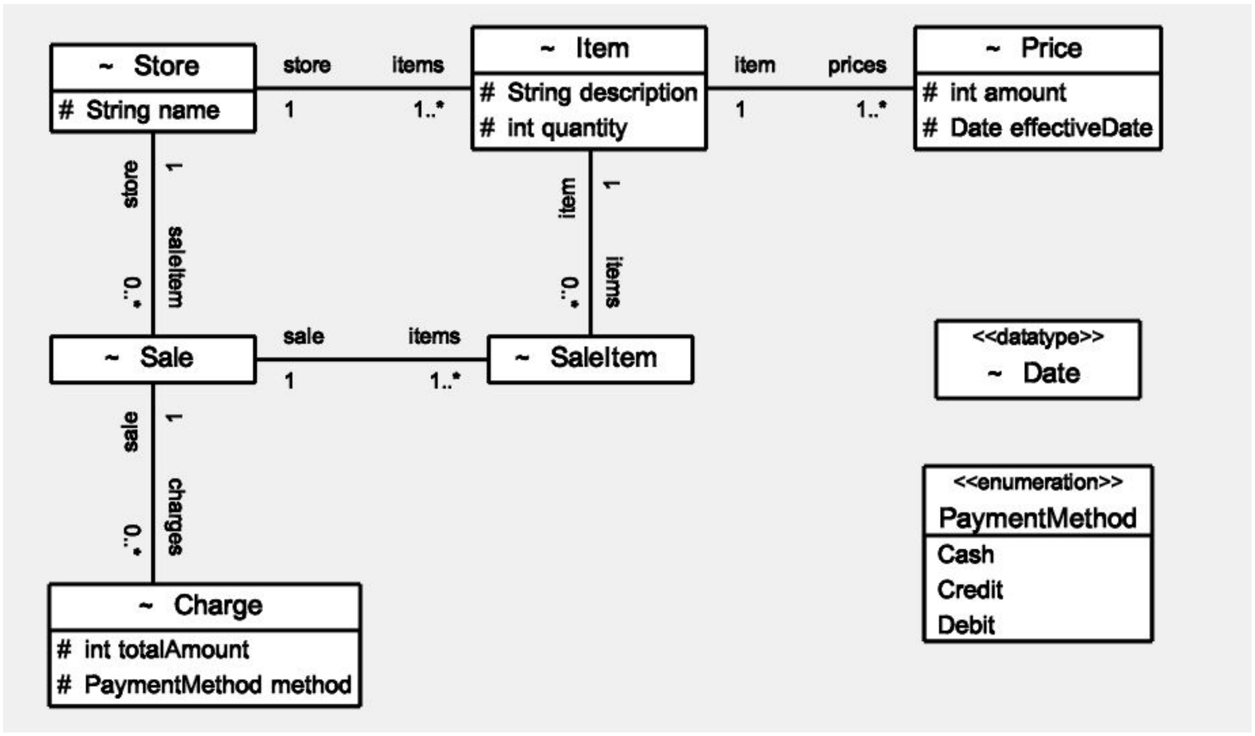


Figure 6.5: Example Domain Model for Test Case

the grammar specified in Appendix A. The two inputs are passed to the compareSuggestedElement function in line 7 to see if there are Suggestions in HVTF which do not exist in the domain model. Then, the output from this function and the domain model is passed to the compare function in line 9 that creates the static rememberTrigger hash map and rememberSynonymPair hash map which are further used for assertions. We first assert the number of possible suggestions recommended to the modeller in line 11. For this case, it should be one. We then fetch the first Suggestion in line 12, then the Triggers given this Suggestion in line 13. We then assert the number of triggers, which in this case should be one in line 14. Then the synonym pairs which exist in the domain model are fetched in line 16. We then assert the number of synonym pairs that exist for this test case in line 17, i.e., there should be two. After this, we check the modelElement and the triggerringElement for the recommended suggestion in lines 19 to 21. In this test, the modelElement should be “AuthorizedPayment” and the two triggeringElements should be “Charge” and “Price”.

The test case for another scenario uses Figure 6.5 again as the domain model and the HVTF in Listing 6.4[4] which contains the recommendation for Attribute “amount” as a ModelElement with

[4]For testing purposes only; does not necessarily reflect past experience.

modelClass as “Cost”. HVTS again detects two instances of the Class, i.e., Price and Charge, in the domain model but the Attribute “amount” matches only with the attribute in Class “Price”. This is accomplished by comparing all the attributes of the matched classes, i.e., Price and Charge, with the ModelElement, i.e., amount. Here, the attribute “amount” of class “Price” is matched. The algorithm also checks for the synonyms of all the attributes to find the possible matches. For example, the synonyms for the attribute “totalAmount” in class “Charge” are not matched to “amount” because “totalAmount” is not a synonym of ”amount” according to the synonym services. Hence the test case is passed, verifying that it matches with the correct synonyms.

Listing 6.3: Test for Multiple Synonyms Exist for a Class

```
@Test
  public void testMultipleSynonymCaseForClass() {

    var fileName="../humanValueTrigger/cdm/StoreManagement/ClassDiagramLanguage/StoreManagement.
    domain_model.cdm";
      var classDiagram = HumanValueTrigger.cdmFromFile(fileName);
      var humanTrigger = HumanValueTrigger.exportXMI("../humanValueTrigger/trigger/Example/
    Class_MultipleSynonym.hvt");
      List<Suggestion> suggestionList=HumanValueTrigger.compareSuggestedElement(humanTrigger,
    classDiagram);

    HumanValueTrigger.compare(classDiagram,fileName,suggestionList);

    assertEquals(1,HumanValueTrigger.rememberTrigger.size());
    var suggestion=HumanValueTrigger.rememberTrigger.entrySet().stream().findFirst().get().getKey()
    ;
    var triggers=HumanValueTrigger.rememberTrigger.get(suggestion);
    assertEquals(1,triggers.size());

    var pairs=HumanValueTrigger.getListOfPairFromTheMap(HumanValueTrigger.rememberSynonymPair,
    suggestion,triggers.get(0));
    assertEquals(2,pairs.size());

    assertEquals("AuthorizedPayment",suggestion.getModelElement().getName());
    assertEquals("Price",pairs.get(0).get("Cost"));
    assertEquals("Charge",pairs.get(1).get("Cost"));

  }
```

Listing 6.4: Human Value Trigger File (HVTF) for Second Test Case

```
Attribute String 'Cost'.slip
Trigger Attribute Int 'Cost'.amount
        Example 'Providing the slip helps to maintain the record for the purchase'
        negIfAbsent healthy because 'It increases the risk of miscommunication between seller and
    the buyer which impacts overall mental healthy for both'
```

The code of the test case is shown in Listing 6.5. The input files (cdm and hvt) are loaded and then passed to the compareSuggestedElement and compare function to find the recommendations for the domain model in lines 7 to 9. Then, the rememberTrigger list is used for assertions. We first assert the number of possible suggestions recommended to the modeller in line 11. For this case, it should be one. We first fetch the Suggestion in line 12 and Triggers in line 13. We then assert the number of triggers, which in this case should be one in line 14. Then, the synonym pair which exists in the domain model is fetched in line 16. We then assert the number of synonym pairs that exist in line 17, i.e., there should be one. After this, we check the modelElement and the triggerringElement for the recommended suggestion in lines 19 to 22. In this test, the modelElement of the Suggestion should be “slip” as an Attribute with modelClass as “Price” and the triggeringElement should be “amount” as an Attribute and the modelClass as “Price”. Here, the “Cost” class is replaced by the matching pair for both the Suggestion and the Trigger.

Listing 6.5: Test for Multiple Synonyms Exist for a Class but only for one Attribute

```
 1 @Test
 2   public void testMultipleSynonymCaseForAttribute() {
 3 
 4     var fileName="../humanValueTrigger/cdm/StoreManagement/ClassDiagramLanguage/StoreManagement.
       domain_model.cdm";
 5       var classDiagram = HumanValueTrigger.cdmFromFile(fileName);
 6       var humanTrigger = HumanValueTrigger.exportXMI("../humanValueTrigger/trigger/Example/
       Attribute_MultipleSynonym.hvt");
 7       List<Suggestion> suggestionList=HumanValueTrigger.compareSuggestedElement(humanTrigger,
       classDiagram);
 8 
 9     HumanValueTrigger.compare(classDiagram,fileName,suggestionList);
10 
11     assertEquals(1,HumanValueTrigger.rememberTrigger.size());
12     var suggestion=HumanValueTrigger.rememberTrigger.entrySet().stream().findFirst().get().getKey()
       ;
13     var triggers=HumanValueTrigger.rememberTrigger.get(suggestion);
14     assertEquals(1,triggers.size());
15 
16     var pairs=HumanValueTrigger.getListOfPairFromTheMap(HumanValueTrigger.rememberSynonymPair,
       suggestion,triggers.get(0));
17     assertEquals(1,pairs.size());
18 
19     assertEquals("slip",suggestion.getModelElement().getName());
20     assertEquals("Price",pairs.get(0).get(((org.xtext.example.hvt.humanValueTrigger.Attribute)
       suggestion.getModelElement()).getModelClass()));
21     assertEquals("Price",pairs.get(0).get("Cost"));
22     assertEquals("amount",pairs.get(0).get("amount"));
23 
24   }
```

6.5 Discussion

Taking into account the system's accuracy, human intervention is necessary to ensure context awareness during domain modelling. The proposed system is designed in such a way that it helps to highlight the potential issues and bring them to the modeller's attention. Furthermore, the system provides suggestions to incorporate in the domain model to address the possible concerns. The system requires the modeller's intervention to choose the suggestions to incorporate in domain model. It is possible that bias is introduced using the catalogue of collected experience. Bias can influence the way modellers perceive, interpret, and analyze information. This can have negative consequences on the developed domain modelling-based solution. To reduce the risk of bias, it is important to have a diverse and inclusive team throughout the modelling process that represents society very well. Additionally, it is helpful to be aware of personal biases and use multiple perspectives and reviews to ensure context and accuracy. This also applies to the task of adding new examples of human value issues to the catalogue of collected experience.

Moreover, it is possible that the proposed system provides suggestions where software / system goals may contradict human values. For example, consider scenario (iv) in Chapter 3, i.e., "People get addicted to WhatsApp, and they do the same thing repeatedly which results in reduced (or no) interaction with other people". In this circumstance, the system could block users after some time. This suggestion will be contradicting to stakeholders responsible for WhatsApp as they may benefit from users engaging with the app as much as possible (even if it is due to addiction). HVTS cannot force stakeholders to adopt suggestions. However, government rules and regulations or public pressure could force stakeholders to adopt the suggestions. To conclude, it is essential for a diverse and inclusive team of humans to be involved in the modelling process to ensure appropriate examples are included in the domain model.

6.6 Summary

This chapter explains the matching algorithm of HVTS in detail. The usage of HVT metamodel elements and the matching criteria for providing the suggestion are explained in this chapter. All the steps are explained with the help of an example. Moreover, this chapter also provides insights into the process followed for the verification of the HVTS. It explains the structure used by the test cases using two examples.

In the next chapter, we discuss the work done by other researchers related to this thesis.

7 Related Work

This chapter first discusses the work done in the field of human values in software engineering in Section 7.1 and then investigates the approaches proposed on pattern matching of class diagrams and in query languages in Section 7.2.

7.1 Human Values

Waqar Hussain *et al.* propose the Value Design Hub (VDH) [19] framework which considers social values when creating design patterns. To carry out the valuefication of design pattern, this framework is created with the collaboration of software developers, users, and social scientists. This study comprises of six component which includes a collection of Guidelines, Indicators, Tools, and Techniques (GITTs) to achieve the goal of VDH. The initial component is a VDH Classifier which consists of a GITTs for categorizing current design patterns and their respective value implications.

The next component "VDH Pattern Maker" uses GITTs to extend or create design pattern which specifically include social values. The subsequent component "VDH Guide" employs GITTs to make it easier for software projects to use value-driven design patterns. The following component "VDH Connector" acts as an input to the aforementioned components as it uses GITTs for capturing and assessing insights on the value facets of design patterns. The next component includes "VDH Monitor" which enables tracking, gathering, and evaluating comments from collaborators on the adoption of value-based design patterns. The last component comprises of "VDH Maintainer" which tracks the variations in value-driven design patterns and GITTs. Fundamentally, this study influences the collaborative integration of social values in software design patterns.

With respect to this work, our research focuses – instead of design patterns – on domain modelling with class diagrams, an important technique for requirements engineering and early design activities. We design a prototype tool which addresses the need for human values to be integrated in software engineering by providing suggestions for a domain model.

Mougouei *et al.* offer a roadmap to operationalize Human Values in Software [23]. In this study, the authors talk about the challenges to accomplish this task as well as open ended research questions that may be investigated in future. The main obstacle indicated in the research is to define human values in a way that can be applied to software, the second difficulty specified discusses about existing software design decisions which are frequently made without considering values, and the third problem specifies the missing metrics to quantify human values in engineering software.

The above stated work focusses on identifying challenges faced while integrating human values in software. The major problem highlighted by this paper includes the lack of practical definitions that are applicable to software designs. Our work focuses on identifying frequently overlooked human values in domain models based on past experiences and providing recommendations to incorporate them. This helps in making design decisions while considering human values.

Perera *et al.* investigate the General Data Protection Regulation (GDPR) [29] to determine the extent to which it encompasses fundamental human values. GDPR is an endeavor by the European Union to safeguard data and personal information of EU people by establishing data protection principles and data subject rights. In their research, the authors applied GDPR rights to understand GDPR principles and then matched these principles to the widely recognized Schwartz theory of basic human values. They demonstrate that GDPR covers a variety of values such as power, security, and universalism etc. and can be utilized to incorporate concrete definitions to human values in the context of software.

Jon Whittle in his paper *"Is Your Software Valueless?"* [39] talks about ignorance of human values such as compassion and justice in software engineering. This article further emphasizes on how values of the software developer community do not align with broader values. Consequently, the author discusses how the current state of different software methods, such as agile development methods and user stories, could easily be modified to ensure that end user values are considered.

Jon Whittle *et al.* in their article *"A Case for Human Values in Software Engineering"* [40] emphasize on the significance of human values in engineering by discussing some preliminary ideas on how the not-for-profit industry incorporates human values. The first insight highlights the need for practical definitions for human values to work with projects. To accomplish this goal, the authors suggested to consider the Schwartz taxonomy and then generate value portraits which encapsulates the meaning of values in the context of the project. This study also underlines the method to provide value-based reasoning for requirements / design choices which further assist team members to recall their choices. Lastly, the authors talk about the need to consider these documented values throughout the lifecycle of software development.

Our research work is highly motivated by the above stated work as we became aware of the significance of human values in software engineering. Our work helps practitioners in capturing the implications for human values based on past experience using the domain-specific language HVT. Also, it helps practitioners to address those values in a new system with similar situations.

Mussbacher *et al.* [25] offer preliminary evidence that a domain model indeed incorporates human values. They propose enhanced guidelines for domain modelling to perform human value analysis to further analyze domain models to demonstrate how existence or absence of elements can have considerable impact on the human values. To identify the domain model elements, the authors explore 58 values compiled by the Schwartz taxonomy to describe the positive or negative influence of these elements on values. They contend that human value enriched modelling is useful to prevent system rejection and detrimental societal effects.

With respect to the above stated work, we also investigate an existing system to compile the list of issues concerning human values using the Schwartz taxonomy. To capture these experiences, we create a domain-specific language called HVT (Human Value Trigger) using a framework called

Xtext which captures different examples for each human value and the model element type including the detailed scenario which identifies the impact occurred with the presence or the absence of the element. Our work provides tool support for the process described by Mussbacher *et al.* [25] by providing suggestions based on the collected past experiences.

To facilitate systematic integration, tracing, and evaluation of human values, Perera *et al.* [31] propose the Continual Value(s) Assessment (CVA) framework which uses goal modelling in combination with feature modelling. This methodology consists of four primary sections. The initial component includes evaluation of the satisfaction of identified stakeholder values, which further helps in understanding the consequences of human values on system. The next component in this study consists of developing an initial feature model for the system. The subsequent component involves identifying connections between features and stakeholder values, which demonstrates the influence of various design decisions after considering the human values at various levels of the software development life cycle (SDLC). The final step determines the extent to which human values are satisfied at various stages of development. Basically, this study illustrates the thinking that links design choices to human values which ultimately ensures that the SDLC is satisfying the value needs of stakeholders.

The above stated work focusses on goal modelling as it handles the positive and negative interaction between different needs. They extended this technique with value-based goals with the help of a feature model that represents design choices whereas this thesis focusses on domain modelling and hence is complementary to the work by Perera *et al.* [31].

Perera *et al.* investigate the publication of Software Engineering conferences and journals (2015-2018) [30] for their relevance to different human values and concluded that only 16% of these papers include human values and 41% of these papers focuses on security related issues implying that very few publications directly addressed the majority of the human values.

Galhotra *et al.* [14] propose a testing-based method to measure the discrimination that may occur in software. In their study, they evaluate twenty software systems and conclude that discrimination is incorporated in software even though fairness is the main goal of developers. The authors further express the necessity to consider fairness testing during software development.

Rifat Ara Shams *et al.* investigate existing Bangladeshi agriculture mobile apps [35] to determine which user desired values are taken into account when creating apps. To conduct this case study, authors manually evaluated user reviews to learn about values user wanted from those apps by classifying them based on Schwartz's theory for human values. During analysis, they explore the values that are missing in the existing apps as well as the extent to which the desired values exist in the available apps. After completing the evaluation, they conclude that only eleven out of twenty-one desired values are reflected in the chosen applications and the remaining values are missing. The result from this study gives guidance on the values to the developers that they should consider when creating Bangladeshi agriculture apps.

Hussain *et al.* conducted a case study [20] to understand changing software development practices by corporations to accommodate human values properly while designing software. This work outlines the relationship between developer's knowledge about values and the company's culture and the level at which values are considered during the development process. Further, this paper discusses the difficulties faced by developers to accommodate them throughout the process. For example, in their research, authors interviewed practitioners from two firms about their elicitation and analysis techniques. The methodology used by the consultants from "Koala" firm revolves around user participation, personas, and prototypes that indicate stakeholder values. Moreover, to persuade clients to consider essential human values they present the consequences of ignoring these values on corporate outcomes. Whereas the "Wallaby" firm practices simply focus on prioritized in-person interaction and iterative development to incorporate values into software. Furthermore, *"participants in Wallaby were more inclined to believe that the delivery of an efficient (time-saving, automated) system addresses user's values automatically as the technology itself makes life easier and meaningful"*. Clearly, these strategies are considered by the team members in early phases of a project to capture user values and emotions and their perspective and understanding significantly influence their considered approach.

Nurwidyantoro *et al.* conducted an exploratory study [26] to extract human values from software development artifacts and use them to address human values throughout software development. To accomplish this task, the authors conduct interviews with software practitioners and develop a prototype called as *"human value dashboard"* to support the process. The participants acknowledges that this process will raise awareness of values among team members. For example, the statement given by P10-Project manager *"So at the design level, we can target those areas where the accessibility issues can be pointed out so that the things are more planned accordingly in* *terms of accessibility”* indicates one of the benefits as it will assist with decision making based on identified values. Moreover, this study concluded *“requirement document”* and *“issue discussion”* as the most appropriate approach for employing artefacts as a source of value identification in the dashboard.

Hussain *et al.* investigate one of the agile methodologies “Scaled Agile Framework” [21] to introduce human values in all software development phases. Their study highlights existing artefacts including user stories, personas, roles, ceremonies, practices, and culture that can be modified to serve as a potential intervention point for incorporating values. Furthermore, authors introduce new methods, e.g., values companion, checklist, and value conversation, to address human values in software.

To better understand the values significant to Bangladeshi female farmers, Rifat Ara Shams et al. conducted a survey using Portrait Values Questionnaire (PVQ) [36] which helps developers to embrace such values during app development. This ensures that the apps developed are aligned with the values of their users. This research demonstrates how social and cultural norms influence the level of importance of different values among different demographics and concludes that Conformity and Security are the most essential values and Power, Hedonism, and Stimulation are the least important.

Nurwidyantoro *et al.* present a case study to show that human values are present in software development artefacts [27]. The authors chose 1097 issues collected from the issues in GitHub of three open-source Android projects: Signal, K-9, and Focus. A pilot study was then performed on the issues collected to determine if human values are present and to discover new human values which are not yet considered in existing software engineering human values’ models. To make sure that the analysts involved for pilot analysis share the same understanding for human values, the concepts and definitions from Schwartz’s theory are considered. The authors develop two definitions based on their results from the pilot study. As a first definition, an observation (issue) can be considered as a value theme if it indicates that there is a feeling of liking or disliking by a contributor towards the application. Second, a value theme is considered present when such a theme is discovered in issue discussions. The main research is conducted after the pilot study and 20 value themes are discovered. It reveals that out of twenty values identified, ten theme values (including conformity, pleasure, dignity, inclusiveness, sense of belonging, freedom, independence, wealth, privacy, and security) directly correspond to Schwartz's human values, while the other ten are more technical and termed as system value themes (trust, correctness, compatibility, portability, reliability, efficiency, energy preservation, usability, accessibility, and longevity). The results conclude that human values can be discovered in almost one-third of the studied issues in Signal, K-9, and Focus apps and automated detection tools should be developed to analyze presence of human value themes in software development projects.

With respect to the above stated work, our work focuses on providing tool support for the detection of human value issues in domain models and provides suggestions based on the captured past experiences.

7.2 Pattern Matching

We investigate some approaches proposed on matching of the class diagram and patterns matching in query languages because extensive research has been done in the field of matching which determines similarity/dissimilarity or extracts data using graphical matching or queries. Bian *et al.* [4] report on the matching of class diagrams in the context of grading student solutions. They present an automatic grading algorithm that compares student responses with template models using syntactic, semantic, and structural matching. Moreover, their algorithm provides the grades based on the outcomes of matching of the model elements. Similarly, Singh *et al.* [37] propose an algorithm to detect mistakes in a class diagram by comparing the student solution with an instructor solution. Their approach compares the model elements based on the information like name, attributes, type of model elements, and relationships and supports matching based on the synonyms of the elements. Moreover, their algorithm provides detailed information about the detected mistake and its related elements. Additionally, Boubekeur *et al.* [5] propose a feedback algorithm that provides progressive feedback for the detected mistakes in a class diagram based on the knowledge of the student. With respect to above stated work, the HVTS also requires matching of the model elements based on the syntactic detection as well as synonym-based semantic detection. Furthermore, pattern matching may be performed using OCL [16] and QVT [17]. OCL is a formal language used to specify constraints and rules on UML models such as preconditions and postconditions, ensuring that the models remain consistent and correct. Pattern matching can be used to define more complex constraints and other expressions by using patterns to describe the structure of objects in the UML models. Whereas QVT is a standard set of languages used to specify transformations. Pattern matching in QVT is used as it provides a way to match patterns within models and apply transformations to those patterns. Clark [8] proposes an extension to OCL object navigation for pattern matching and introduces declarative patterns that can be used to match object structures instead of using lengthy repeated navigation statements. This paper analyzes OCL extensions with elaborated examples and syntax definitions. Li *et al.* [22] present a graphical model query using semantics and pattern matching of QVT relations. Further, their work proposes a method for mapping a query's selection criteria into XSLT to make the execution easier. In addition, they develop a tool to design queries and generate XSLT code automatically. We acknowledge that a lot of work has been done in the area of matching diagrams using graphical and query languages. For greater control over the matching process, we opt for a custom-built solution for our proof-of-concept implementation.

7.3 Summary

This chapter provides an insight of the work done in the domain of human values in software engineering. Furthermore, various approaches for matching of class diagrams are reviewed. This chapter discusses the related work done by other authors and states how the proposed HVTS is different from the existing approaches.

The next chapter concludes this thesis and talks about the future improvements to the HVTS.

8
Conclusions

Thesis Statement. Lack of consideration for human values when developing software often leads to social repercussions. The proposed Human Value Trigger System aims to address the need for human value consideration during domain modelling by providing suggestions based on collected past experiences.

Human values play a significant role in decision making in users, practitioners, and organizations. Users expect software that considers human values. The Human Value Trigger System (HVTS) aims to reduce the ignorance of human values during domain modelling by guiding software practitioners. The HVTS incorporates human values by providing suggestions for a domain model based on past experiences. With the proposed HVT, a domain-specific language called Human Value Trigger, different examples for each human value and various types of model elements are captured along with a detailed scenario that explains how the presence or absence of the model element impact the values.

In this thesis, we specify the domain-specific language HVT that captures examples from past experiences. We explain the language and the metamodel for HVT in greater detail. We investigate the domain model for the WhatsApp System considering all the values in Schwartz's taxonomy to motivate our approach to include human values-based elements in our domain modelling. Moreover, we present in detail the implementation of the matching algorithm of HVTS. Furthermore, an analysis of eight synonym services including dictionaries and thesauri as well as NLP-based and AI-based services is performed to find the optimal synonym service or combination of synonym services to use with the implementation. To validate the system, we test the HVTS using tests with different human value trigger files and domain models, boosting our confidence in the matching capability of HVTS. However, there is still potential for advancement in this research area.

In the future, the matching algorithm of HVTS could be improved so that it works for patterns of multiple elements as Trigger and Suggestion instead of single elements. This could be difficult to achieve with a "from scratch" solution as presented in this thesis as this will require pattern matching for all trigger elements to add the suggestion in the domain model. Instead, pattern matching and model querying technologies such as OCL and QVT could be used. Additional examples from past experiences could be collected using the grammar specified by HVT over time. Moreover, a user study could be conducted to assess the usefulness of the proposed system. Furthermore, other key RE modelling techniques such as goal models and state models could be analyzed to utilize a similar approach to address human values throughout the process of software development.

Bibliography

[1] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, P. Naur, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. Revised report on the algorithmic language ALGOL 60. *The Computer Journal*, 5(4):349–367, 01 1963. ISSN 0010-4620. doi: 10.1093/comjnl/5.4.349. URL `https://doi.org/10.1093/comjnl/5.4.349`.

[2] N. Baker. Molly russell: Instagram bans graphic selfharm images after suicide of uk teen, 2019. URL `https://www.sbs.com.au/news/molly-russell-instagram-bans-graphic-self-harm-images-after-suicide-of-uk-teen`.

[3] Thorsten Berger. A new dsl textbook in town! Keynote, Educators Symposium at 2022 ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS'22).

[4] Weiyi Bian, Omar Alam, and Jörg Kienzle. Automated grading of class diagrams. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 700–709, 2019. URL `https://doi.org/10.1109/MODELS-C.2019.00106`.

[5] Younes Boubekeur. A learning corpus and feedback mechanism for a domain modeling assistant, McGill University, Canada, June 2022. URL `https://escholarship.mcgill.ca/concern/theses/9593v1553`.

[6] Lethbridge T. C. and Laganière Robert. *Object-oriented software engineering : practical software development using uml and java.* 2nd edition. McGraw Hill / Europe, Middle East and Africa, 2004.

[7] C. Cadwalladr and E. Graham-Harrison. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach, 2018. URL `https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election`.

[8] Tony Clark. Ocl pattern matching. *OCL@MoDELS, CEUR*, (1092:33–42), 01 2013.

[9] Benoit Combemale, Robert France, Jean-Marc Jézéquel, Bernhard Rumpe, James Steel, and Didier Vojtisek. *Engineering Modeling Languages: Turning Domain Knowledge into Tools.* Chapman and Hall/CRC, 2020.

[10] Eclipse Foundation. Eclipse Modeling Framework (EMF). URL `https://www.eclipse.org/modeling/emf/`.

[11] GloVe: Global Vectors for Word Representation. Website. URL `https://nlp.stanford.edu/projects/glove/`.

[12] Eclipse Foundation. Ecore. URL `https://wiki.eclipse.org/Ecore`.

[13] Context free grammar. Website. URL `https://en.wikipedia.org/wiki/Context-free_grammar`.

[14] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. Fairness testing: Testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 498–510, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450351058. doi: 10.1145/3106237.3106277. URL `https://doi.org/10.1145/3106237.3106277`.

[15] Object Management Group. The MetaObject Facility Specification™ (MOF™), April 2002. URL `https://www.omg.org/mof/`.

[16] Object Management Group. Object constraint language, February 2014. URL `https://www.omg.org/spec/OCL`.

[17] Object Management Group. Mof query/view/transformation, June 2016. URL `https://www.omg.org/spec/QVT/1.3/About-QVT/`.

[18] Object Management Group. OMG® Unified Modeling Language® (OMG UML®), Dec 2017. URL `https://www.omg.org/spec/UML/2.5.1/PDF`.

[19] Waqar Hussain, Davoud Mougouei, and Jon Whittle. Integrating social values into software design patterns. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pages 8–14, 2018. doi: 10.1145/3194770.3194777.

[20] Waqar Hussain, Harsha Perera, Jon Whittle, Arif Nurwidyantoro, Rashina Hoda, Rifat Ara Shams, and Gillian Oliver. Human values in software engineering: Contrasting case studies of practice. *IEEE Transactions on Software Engineering*, 48(5):1818–1833, 2022. doi: 10.1109/TSE.2020.3038802.

[21] Waqar Hussain, Mojtaba Shahin, Rashina Hoda, Jon Whittle, Harsha Perera, Arif Nurwidyantoro, Rifat Ara Shams, and Gillian Oliver. How can human values be addressed in agile methods? a case study on safe. *IEEE Transactions on Software Engineering*, 48(12):5158–5175, 2022. doi: 10.1109/TSE.2022.3140230.

[22] Dan Li, Xiaoshan Li, and Volker Stolz. Model querying with graphical notation of qvt relations. *SIGSOFT Softw. Eng. Notes*, 37(4):1–8, jul 2012. ISSN 0163-5948. doi: 10.1145/2237796.2237808. URL `https://doi.org/10.1145/2237796.2237808`.

[23] Davoud Mougouei, Harsha Perera, Waqar Hussain, Rifat Shams, and Jon Whittle. Operationalizing human values in software: A research roadmap. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, page 780–784, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355735. doi: 10.1145/3236024.3264843. URL `https://doi.org/10.1145/3236024.3264843`.

[24] MT4j. Multitouch for java (mt4j) framework. URL `https://sites.google.com/site/gmitresearch/mt4j`.

[25] Gunter Mussbacher, Waqar Hussain, and Jon Whittle. Is there a need to address human values in domain modelling? In *2020 IEEE Tenth International Model-Driven Requirements Engineering (MoDRE)*, pages 73–77, 2020. doi: 10.1109/MoDRE51215.2020.00015.

[26] Arif Nurwidyantoro, Mojtaba Shahin, Michel Chaudron, Waqar Hussain, Harsha Perera, Rifat Ara Shams, and Jon Whittle. Towards a human values dashboard for software development: An exploratory study. In *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ESEM '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386654. doi: 10.1145/3475716.3475770. URL https://doi.org/10.1145/3475716.3475770.

[27] Arif Nurwidyantoro, Mojtaba Shahin, Michel R.V. Chaudron, Waqar Hussain, Rifat Shams, Harsha Perera, Gillian Oliver, and Jon Whittle. Human values in software development artefacts: A case study on issue discussions in three android applications. *Information and Software Technology*, 141:106731, 2022. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2021.106731. URL https://www.sciencedirect.com/science/article/pii/S0950584921001828.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[29] Harsha Perera, Waqar Hussain, Davoud Mougouei, Rifat Ara Shams, Arif Nurwidyantoro, and Jon Whittle. Towards integrating human values into software: Mapping principles and rights of gdpr to values. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 404–409, 2019. doi: 10.1109/RE.2019.00053.

[30] Harsha Perera, Waqar Hussain, Jon Whittle, Arif Nurwidyantoro, Davoud Mougouei, Rifat Ara Shams, and Gillian Oliver. A study on the prevalence of human values in software engineering publications, 2015 – 2018. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 409–420, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371216. doi: 10.1145/3377811.3380393. URL https://doi.org/10.1145/3377811.3380393.

[31] Harsha Perera, Gunter Mussbacher, Waqar Hussain, Rifat Ara Shams, Arif Nurwidyantoro, and Jon Whittle. Continual human value analysis in software development: A goal model based approach. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 192–203, 2020. doi: 10.1109/RE48521.2020.00030.

[32] Nor Samsiah Binti Sani. Lab 3: Introduction to domain modeling and class diagram, 2009-2010. URL https://norsamsiah.files.wordpress.com/2010/01/lab-003-domain-modeling1.pdf.

[33] Matthias Schöttle, Nishanth Thimmegowda, Omar Alam, Jörg Kienzle, and Gunter Mussbacher. Feature modelling and traceability for concern-driven software development with touchcore. In *Companion Proceedings of the 14th International Conference on Modularity*, MODULARITY Companion 2015, page 11–14, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450332835. doi: 10.1145/2735386.2735922. URL https://doi.org/10.1145/2735386.2735922.

[34] Shalom H. Schwartz. *An Overview of the Schwartz Theory of Basic Values.* Online Readings in Psychology and Culture, 2(1), 2012. URL https://doi.org/10.9707/2307-0919.1116.

[35] Rifat Ara Shams, Waqar Hussain, Gillian Oliver, Arif Nurwidyantoro, Harsha Perera, and Jon Whittle. Society-oriented applications development: Investigating users' values from

bangladeshi agriculture mobile applications. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*, ICSE-SEIS '20, page 53–62, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371254. doi: 10.1145/3377815.3381382. URL `https://doi.org/10.1145/3377815.3381382`.

[36] Rifat Ara Shams, Mojtaba Shahin, Gillian Oliver, Waqar Hussain, Harsha Perera, Arif Nurwidyantoro, and Jon Whittle. Measuring bangladeshi female farmers' values for agriculture mobile applications development. *CoRR*, abs/2012.01268, 2020. URL `https://arxiv.org/abs/2012.01268`.

[37] Prabsimran Singh. Domain modeling mistake detection system, McGill University, Canada, 2022. URL `https://escholarship.mcgill.ca/concern/theses/5x21tm741`.

[38] TouchCORE. Website. URL `http://touchcore.cs.mcgill.ca/`.

[39] Jon Whittle. Is your software valueless? *IEEE Software*, 36(3):112–115, 2019. doi: 10.1109/MS.2019.2897397.

[40] Jon Whittle, Maria Angela Ferrario, Will Simm, and Waqar Hussain. A case for human values in software engineering. *IEEE Software*, 38(1):106–113, 2021. doi: 10.1109/MS.2019.2956701.

[41] Niklaus Wirth. The programming language pascal. *Acta informatica*, 1(1):35–63, 1971.

[42] Xtext. Website. URL `https://www.eclipse.org/Xtext/documentation/301_grammarlanguage.html`.

A

Grammar Definition for Human Value Trigger (HVT) DSL

Listing A.1: Grammar Definition for Human Value Trigger in Xtext

```
grammar org.xtext.example.hvt.HumanValueTrigger with org.eclipse.xtext.common.Terminals

generate humanValueTrigger "http://www.xtext.org/example/hvt/HumanValueTrigger"

HumanValueTriggerSystem:
  (humanValues+=HumanValue)*
  (suggestions+=Suggestion)*
;

Suggestion:
  (modelElement=ModelElement)
  (alternativeTriggers+=Trigger)*
;

ModelElement:
  (Attribute | AssociationEnd | AssociationClass | Class | Enumeration | Literal )
;

Attribute:
  'Attribute' type=AttributeType modelClass=STRING '.' name=ID
;

enum AttributeType:
  Boolean='Boolean' | Double='Double' | Int='Int' | Long='Long' |
  String='String' | Byte='Byte' | Float='Float' | Char='Char' |
  Date='Date'| Time='Time'
;

AssociationEnd:
  'AssociationEnd' modelClass=STRING '.' name=ID referenceType=ReferenceType otherClass=STRING '['
    lowerBound=STRING '..' upperBound=STRING ']'
;
```

```

enum ReferenceType:
  Composition='<@>-' | Aggregation='<>-' | Regular='--' | Any='?-'
;

AssociationClass:
  'AssociationClass' name=ID firstClass=STRING '--' secondClass=STRING
;

Class:
  'Class' name=ID
;

Enumeration:
  'Enumeration' name=ID
;

Literal:
  'Literal' modelClass=STRING '.' name=ID
;

Trigger:
  'Trigger' (triggeringElement=ModelElement) (examples+=Example)*
  ('posIfAbsent' positivelyImpactedIfAbsent+=Reason)*
  ('negIfAbsent' negativelyImpactedIfAbsent+=Reason)*
;

Reason:
  (humanValue=[HumanValue])
  'because' explanation=STRING
;

HumanValue:
  'HumanValue' category=HumanValueCategory '.' name=ID description=STRING
;

enum HumanValueCategory:
  SelfDirection='SelfDirection' | Stimulation='Stimulation' | Hedonism='Hedonism' |
  Achievement='Achievement' | Power='Power' | Security='Security' | Conformity='Conformity' |
  Tradition='Tradition' | Benevolence='Benevolence' | Universalism='Universalism'
;

Example:
  'Example' detail=STRING
;
```

B

HVTF for all WhatsApp Scenarios Covered in Chapter 3

Listing B.1: Human Value Trigger File for WhatsApp Scenarios

```
HumanValue Universalism.BroadMindedOrTolerance 'Liberal in views and reactions'
HumanValue Universalism.SocialJustice 'Everyone deserves equal economic, political, and social
    rights and opportunities'
HumanValue Universalism.Wisdom 'The quality of having experience, knowledge, and good judgment'
//Apart from the above mentioned values, there are a total of 58 values defined according to
    Schwartz's taxonomy of human values (not all shown for brevity).

Attribute Boolean 'Message'.inappropriate
Trigger Attribute String 'Message'.body
        Example 'For the Group class, various WhatsApp groups exist and anyone knowing of their
    existence can share information. So, misinformation could potentially be shared by members from
     one group to another which leads to the circulation of rumors sometimes and ultimately may
    cause disruption in the society or even a crisis in the society.'
        negIfAbsent Privacy because 'Misinformation shared in the groups can effect the privacy as
    for example wrong information shared about the person health can effect the privacy.'
        negIfAbsent Freedom because 'Misinformation shared can cause confusion which ultimately
    effects freedom.'
        negIfAbsent ChoosingOwnGoal because 'Misinformation shared in the groups can influence the
    decision making as for example in case of elections'
        negIfAbsent EnjoyingLife because 'Wrong information conveyed to the people can cause
    unwanted stress or initiate overthinking.'
        negIfAbsent Successful because 'Wrong information transmission can effect the reputation.'
        negIfAbsent Influential because 'Misinformation can effect the mindset of an individual.'
        negIfAbsent SocialPower because 'Wrong information conveyed through messages could lead to
    massive destructions like riots.'
        negIfAbsent Authority because 'Misinformation shared could pressurize authorities to take
    severe actions.'
        negIfAbsent PreservingMyPublicImage because 'Misinformation can effect the public image of
    a person, e.g., fake news about the celebrity.'
        negIfAbsent Healthy because 'Misleading content can cause damage to a person both mentally
    or physically.'
        negIfAbsent FamilySecurity because 'Sharing of misleading sensitive information could put
    an individual into a critical situation.'
        negIfAbsent SocialOrder because 'Misleading information can cause chaos in the social
    circle of an individual.'
```

```
        negIfAbsent NationalSecurity because 'Misinformation can lead to disorder between two nations.'
        negIfAbsent SelfDiscipline because 'A frequent misinformation can lead to distractions from the goals of an individual.'
        negIfAbsent Politeness because 'Misleading information can cause unnecessary debates between people or communities.'
        negIfAbsent Honest because 'People unknowingly share the wrong information and are often wronged by others due to this.'
        negIfAbsent Responsible because 'Wrong information unknowingly shared by the users but nobody takes the responsibility of the chaos it can cause.'
        negIfAbsent AWorldOfPeace because 'Misinformation can lead to crisis between the nationalities'
        negIfAbsent BroadMindedOrTolerance because 'Wrong information influences the thinking and the mindset of an individual.'

AssociationEnd 'ProfilePhoto'.sharedWith -- 'User' [ '0'..'*' ]
Trigger Enumeration Privacy
        Example 'For the ProfilePhoto class, we have only options like Everyone, My Contacts, and Nobody to secure the privacy of the profile picture. But sometimes people need to save a random contact number to have one-time contact with another person. There is a minute possibility that the profile photo can be saved, e.g., by taking the screenshot which can be misused'
        negIfAbsent Privacy because 'Profile photo can be saved by taking the screenshot which can be misused.'
        negIfAbsent SelfRespect because 'Misusage of those photos can directly influence self respect.'
        negIfAbsent PreservingMyPublicImage because 'Photos saved as screenshot can be used to defame the person by editing those photos.'
        negIfAbsent SocialRecognition because 'Misusage of those photos can deeply impact the recongnition received by an indivdual.'
        negIfAbsent Healthy because 'Misusage of those photos can cause stress and anxiety to an individual.'
        negIfAbsent ASpiritualLife because 'Thinking about the consequences of misusage of the picture can have the severe impact on the peace of mind'
        negIfAbsent SocialJustice because 'Consequences of misusage of the picture can have severe impact on the jusctice received by an individual.'

AssociationEnd 'User'.whitelistContacts -- 'User' [ '0'..'*' ]
Trigger AssociationEnd 'User'.contacts -- 'User' [ '0'..'*' ]
        Example 'If a random person somehow has the contact number of another person, then that random person can send messages to the other person. While the receiver has the option to block the sender after receiving the message, that message can have an impact on the receiver in various ways depending upon the type of information being shared in the message.'
        negIfAbsent Privacy because 'Continuous messages from a random person can affect the private life of an individual.'
        negIfAbsent Pleasure because 'Messages from a random person can cause unnecessary stress which further impacts pleasure.'
        negIfAbsent EnjoyingLife because 'Messages from a random person could affect the perspective of an individual towards life and another individual.'
        negIfAbsent SelfRespect because 'Random messages from a random person could also affect the self respect of an individual and cause sense of insecurity.'
        negIfAbsent PreservingMyPublicImage because 'Any random person could easily tarnish the public image of an individual by broadcasting bogus messages over the WhatsApp.'
        negIfAbsent ASpiritualLife because 'Messages from a random person could cause sense of restlessness amongst the individual.'

Attribute Time 'User'.timeYouWantToSpend
Trigger Class User
        Example 'People get addicted to WhatsApp, and they do the same thing repeatedly which results in reduced (or no) interaction with other people'
        negIfAbsent Creativity because 'People are addicted to social media so they keep on doing
    the same thing every day, which kills creativity'
        negIfAbsent Curious because 'People get addicted to chatting so they do not study news or
    current topics or study their course.'
        negIfAbsent AVariedLife because 'People are addicted to WhatsApp and doing the same thing
    over and over again.'
        negIfAbsent SelfIndulgent  because 'People are addicted to WhatsApp not knowing which
    person is sitting next to them.'
        negIfAbsent Healthy because 'People spend more time facing health issues like migraine
    problem.'
        negIfAbsent SelfDiscipline because 'People spend more time on WhatsApp due to which people
    forget to do important work.'
        negIfAbsent Politeness because 'There is a change in style of conversation; basically
    WhatsApp changed the way people talk mostly on instant messages or on call'
        negIfAbsent HonoringOfElders because 'WhatsApp changed the way people talk mostly on
    instant messages or on call so people forget the right way.'
        negIfAbsent Obedient  because 'Due to constant usage people tend to lose track of time.'
        negIfAbsent Humble  because 'Due to excessive use there is a change of behavior among the
    users.'
        negIfAbsent RespectForTradition  because 'People are so involved in using the WhatsApp that
     they forget about the customs and tradition.'
        negIfAbsent Moderate  because 'Due to addiction people tend to go to extremes.'
        negIfAbsent Forgiving  because 'As people spend more time on WhatsApp the same things come
    up again and again.'
        negIfAbsent Loyal  because 'Due to addiction people tend to talk more than needed, which
    may lead to miscommunication among users.'
        negIfAbsent TrueFriendship  because 'Due to spending more time on WhatsApp people ignore
    their relationship with the people around them.'
        negIfAbsent MatureLove  because 'Spending more time on WhatsApp can impact relationships by
     decreasing the amount and quality of time people spend together.'

Attribute Boolean 'User'.virtualAssistantIsRequired
Trigger Class User
        Example 'Currently there is no way for people with disabilities to use the WhatsApp
    especially if the user is visually impaired'
        negIfAbsent Creativity because 'User with disabilities are not provided with any assistance
     to use WhatsApp with ease'
        negIfAbsent Freedom because 'Absence of assistance for disabled also puts restriction
    towards freedom of speech for disabled users.'
        negIfAbsent ChoosingOwnGoal because 'Absence of assistance for disabled also makes it
    difficult to navigate various options.'
        negIfAbsent AnExcitingLife because 'As a new trend emerges for people to connect with each
    other via WhatsApp, absence of assistance causes major blockage to disabled users to connect
    with one another.'
        negIfAbsent EnjoyingLife because 'It is impossible for visually impaired people to enjoy
    the perks of interaction using the app.'
        negIfAbsent Authority because 'Disabled users are not provided with any authorization over
    the data/info transfered.'
        negIfAbsent SenseOfBelonging because 'As disabled users find it difficult to use WhatsApp
    as a regular tool, a sense of belonging is always missing.'
        negIfAbsent EqualityForAll because 'Due to absence of features for disabled user, it would
    be impossible for such users to follow up with trends of WhatsApp as compared to normal users.'
        negIfAbsent SocialJustice because 'Absence of assistance for disabled makes them feel
    excluded'
```