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

Jasneet Kaur
Dept. ECE, McGill University
Montreal, QC, Canada
jasneet.kaur@mail.mcgill.ca

Gunter Mussbacher
Dept. ECE, McGill University
Montreal, QC, Canada
gunter.mussbacher@mcgill.ca

***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 as can be seen by recent examples involving Facebook or Delta Airlines. Therefore, it is important to help software developers incorporate human values considerations throughout the software development process. In this paper, we focus on domain modelling with class diagrams, an important technique for requirements engineering and early design activities. We propose a domain-specific language called HVT (Human Value Trigger) that enables the collection of human value issues including how to mitigate them. Practitioners may utilize this language to contribute more such examples to grow a catalogue of these past experiences over time. As a motivating example, 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 (i.e., model elements that may affect various human values). Furthermore as proof-of-concept, a prototype implementation addresses the need for human values to be integrated in domain models with the help of these collected past experiences by providing suggestions based on the model element type, name, and semantics based on synonyms. An analysis of eight synonym services is performed to find the optimal synonym service or combination of synonym services to use with the implementation.**

***Index Terms*—Human values, domain model, synonyms**

I. 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 [1]. 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, 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 [2]. The ticket reservation system of Delta Air Lines charged people evacuating from areas hit by Hurricane Irma five times more than the usual ticket price. It was seen as a breach of human values by the system [3]. 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 [4]. Consequently, it is important to incorporate human values considerations throughout software development.

However, considering human values in software engineering is challenging due to the lack of methods to track and incorporate those values in all phases of the software 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.* [5] highlight these incidents and express the necessity to integrate human values into software.

To incorporate human values in domain models[1], this paper proposes a domain-specific language called HVT (Human Value Trigger) which captures past experience, i.e., different examples for potential human value issues in domain models. This includes a detailed description that shows how the presence or absence of model elements in the domain model positively or negatively impacts the human values identified by Schwartz's taxonomy [6].

Furthermore, this paper proposes the Human Value Trigger System (HVTS) that provides suggestions for a domain model based on captured past experiences. The purpose of HVTS is to make the modeller think of human values through possible human value issues. The modeller still has to decide whether an issue actually exists. The proposed system takes two inputs, a domain model and a catalogue of human value issues capturing past experience. The catalogue is defined with HVT. The system iterates over the model elements of the catalogue 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 semantics based on synonyms. The system presents the matches as possible suggestions to the modeller. The modeller manually assesses the impact of the suggestions, using the impacts from the catalogue. The modeller then selects which suggestions to incorporate in the domain model, and the system integrates them into the domain model. To demonstrate the feasibility of HVTS, a prototype tool that implements the proposed system is developed as proof-of-concept.

[1]In this paper, domain model refers to a *system domain model* that specifies only those domain aspects that will be represented by a system and not an *exploratory domain model* that aims to capture the domain more broadly.

Furthermore, we analyze eight synonym services to find the optimal synonym service or combination of synonym services to use for HVTS during the matching process.

Section II explores background information and discusses related work, Section III introduces a motivating example, Section IV illustrates the metamodel used for HVT, Section V elaborates the analysis of synonym services, and Section VI concludes the paper and discusses future work.

II. Background and Related Work

This section provides brief background information on human values, domain modelling, and work done in the field of human values in software engineering.

Human values are valuable in our life as they help us to grow and develop. In 1992, Schwartz [6] [7] 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 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 control over others (power, achievement). Some but not all values defined by Schwartz match typical non-functional requirements. 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.

A **domain model** [8] 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 [9]. In UML, a class diagram is one way to describe the structure of the system. Basically, a domain model uses

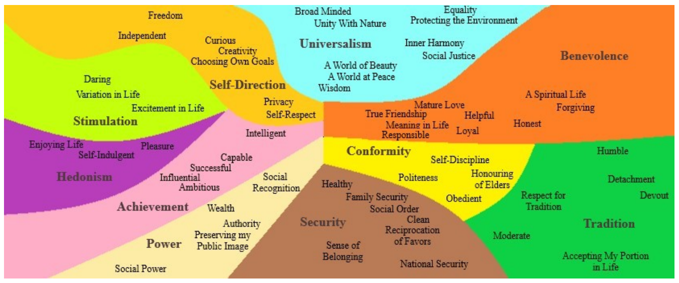

Fig. 1. Schwartz's Theory of Basic Human Values (adapted from source [7])

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) [10].

Human Values in Software Engineering. Waqar Hussain *et al.* propose the Value Design Hub (VDH) [3] which considers social values when creating design patterns. To carry out the valuefication of such patterns, this framework is created with the collaboration of software developers, users, and social scientists. Fundamentally, this study entails 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 [11]. This work focuses 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) [12] to determine the extent to which it encompasses fundamental human values. 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?"* [13] 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. Moreover, Jon Whittle *et al.* in their article *"A Case for Human Values in Software Engineering"* [1] 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 or 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 in a catalogue using the domain-specific language HVT. Also, it helps practitioners to address those values in a new system with similar situations.

Mussbacher *et al.* [14] 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). Our work provides tool support for the process described by Mussbacher *et al.* [14] by providing suggestions based on the collected past experiences.

To facilitate systematic integration, tracing, and evaluation of human values, Perera *et al.* [7] propose the Continual Value(s) Assessment (CVA) framework which uses goal modelling in combination with feature modelling. This study illustrates the thinking that links design choices to human values which ultimately ensures that the software development life cycle is satisfying the value needs of stakeholders.

The above stated work focuses 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 paper focuses on domain modelling and hence is complementary to the work by Perera *et al.* [7].

Perera *et al.* investigate the publication of Software Engineering conferences and journals (2015-2018) [15] for their relevance to different human values and concluded that only 16% of these papers include human values and 41% of these papers focus on security issues, i.e., very few publications directly addressed the majority of the human values.

Galhotra *et al.* [5] 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 development.

Rifat Ara Shams *et al.* investigate existing Bangladeshi agriculture mobile apps [16] to determine which user desired values are taken into account when creating apps. The result from this study gives guidance on the values to the developers that they should consider when creating these apps.

Hussain *et al.* conducted a case study [17] 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.

Nurwidyantoro *et al.* conducted an exploratory study [18] 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 *"human value dashboard"* to support the process. The participants acknowledges that this process will raise awareness of values among team members. 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" [19] 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, the authors introduce new methods, e.g., values companion, checklist, and value conversation, to address human values. Nurwidyantoro *et al.* present a case study to show that human values are present in software development artefacts [20] and reveal 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).

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

III. Motivating Example

This section 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. WhatsApp is a chat application that provides instant messaging and calling services to its user. Figure 2 exhibits all the key 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 [14]. 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

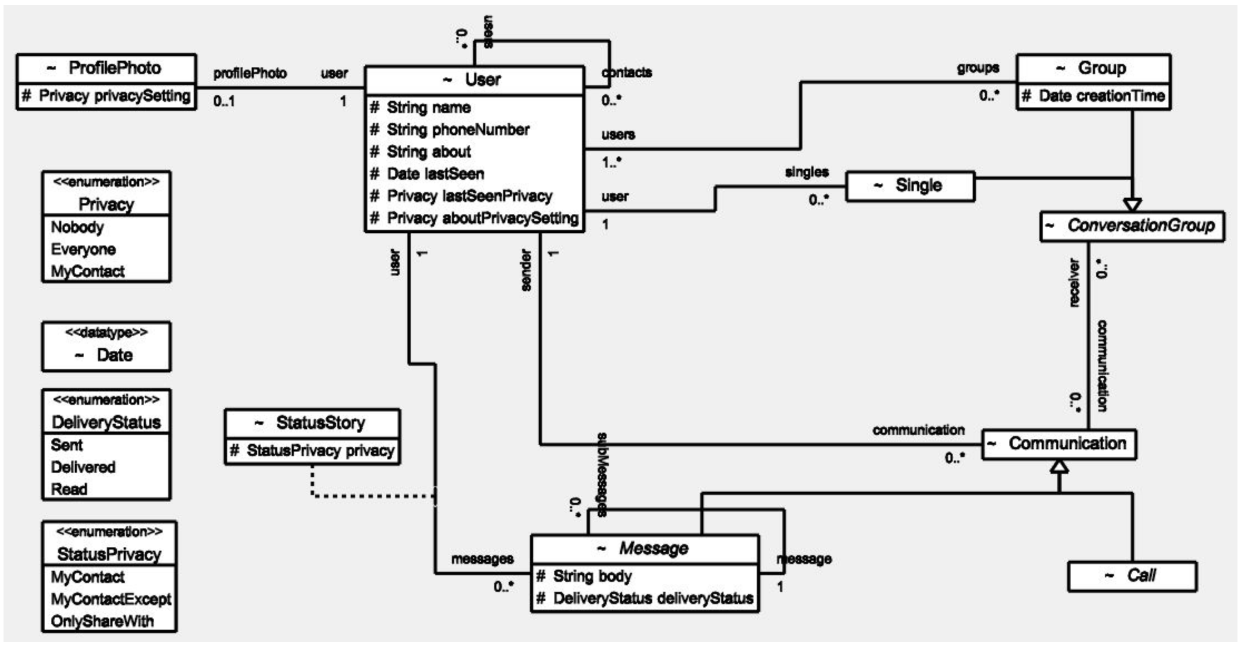

Fig. 2. WhatsApp Domain Model

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.

The domain model supports 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. Note that the subclasses for different types of messages and calls are not shown in Figure 2 due to space constraints. Privacy settings for the profile photo and the status stories shared by the user on their 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. Four scenarios are discussed here.

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 may lead to the circulation of rumors and ultimately may cause societal disruption or even a societal crisis. For example, elections may be manipulated by conveying false information. 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. To prevent this breach, one could add 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, one could add 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, one could add the option to create a whitelist of contact persons as a protection step.

iv) Lastly, people may get addicted to WhatsApp, doing the same thing repeatedly which may result 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, one could add 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.

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 in a catalogue 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 implications of human values they might have in the new 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 in a catalogue by capturing different examples for human values and model elements. For example, the above four scenarios where human values were disregarded could be the first four examples in a catalogue of potential human value issues. Based on this, we formalize a DSL that can express such detailed scenarios, the human values impacted by a scenario, and the involved model elements as explained in the next section. The DSL enables the specification of the catalogue.

IV. Metamodel and DSL

This section discusses the HVT metamodel in detail and explains the use of the Xtext language [21] to define the grammar for the proposed domain-specific language HVT.

The Human Value Trigger (HVT) metamodel as shown in Figure 3 defines the human values as well as the suggestions provided in response to the detection (trigger) 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.

The *Suggestion* class contains the different triggers collected based on past experiences and the suggested element that could be added to prevent an impact on various human values caused by the triggers. In other words, a suggestion groups all the detected human value issues (triggers) that could be addressed by a change to the domain model (i.e., the suggested model element). Each *Suggestion* has one 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 Section III, *"timeYouWantToSpend"* is the suggested model element (an attribute for the User Class in this case) to avoid a similar situation. The triggers of a *Suggestion* are alternatives, i.e., each one could lead to the addition of the suggested model element in the domain model.

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 the reasons on how presence or absence of the triggering element impacts different human values. Each *Example* refers to a detailed explanation for the trigger. Here *"People may get addicted to WhatsApp, doing the same thing repeatedly which may result 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* 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 a proper model element in the chosen domain modelling notation (i.e., the TouchCORE Class Diagram Metamodel (CDM) [22] for our prototype). 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"*. Furthermore, if we were to add these additional features, we would have to explicitly specify which elements are being matched and

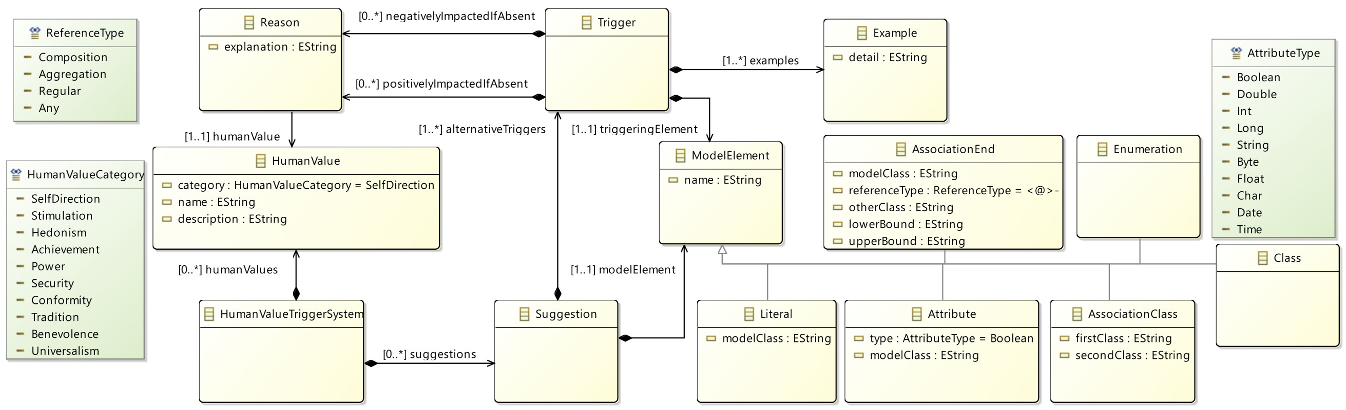

Fig. 3. Human Value Trigger Metamodel

which ones are not being matched for the analyzed domain model. Including only the required 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 3 is equivalent to the grammar definition specified with Xtext. This grammar includes the metamodel and the concrete syntax of a domain-specific language which is used by a modeller to produce or read HVT models (i.e., catalogues) 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 1 shows what scenario (iv) of the motivating example discussed previously looks like as an HVT model.

The grammar specifies a *HumanValueTriggerSystem* which contains first a list of *HumanValues* (two are shown in Listing 1 in Lines 1-2) and then a list of *Suggestions* (one shown in Lines 5-11 with one suggested *ModelElement* (Line 5) and one *Trigger* (Lines 6-11)). Several alternative triggers could be specified. The specification of a human value starts with the keyword “HumanValue” followed by the category which is separated from the name of the human value using the “.” as shown in Listing 1. This is followed by the description of the human value.

The suggested model element (Line 5) first specifies the kind of model element with the keyword ”Attribute”, ”AssociationEnd”, ”AssociationClass”, ”Class”, ”Enumeration”, or ”Literal”. This is followed by additional information as required for the kind of model element. E.g., the attribute’s type (Time), class (User), and name (timeYouWantToSpend) are specified for an Attribute in Line 5.

We use the keyword “Trigger” to represent a trigger in a suggestion. Line 6 shows that the trigger specifies a pattern to match which is “Class User”. Here, the User class will be matched to suggest possible recommendations for the analyzed domain model. If a match exists and the modeller agrees with the suggested model element ”Attribute Time ’User’.timeYouWantToSpend”, then the attribute will be added to the User class in the analyzed domain model.

```
1  HumanValue Universalism.BroadMindedOrTolerance
       'Liberal in views and reactions'
2  HumanValue Universalism.SocialJustice 'Everyone
        deserves equal economic, political, and
       social rights and opportunities'
3  //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)
4
5  Attribute Time 'User'.timeYouWantToSpend
6  Trigger Class User
7          Example 'People may get addicted to
       WhatsApp, doing the same thing repeatedly
       which may result in reduced (or no)
       interaction with other people'
8          negIfAbsent Creativity because 'People
       addicted to social media keep on doing the
       same thing every day, which kills
       creativity'
9          negIfAbsent Curious because 'People
       addicted to chatting may not read up on
       news or current topics or study for their
       course'
10         negIfAbsent AVariedLife because 'People
        addicted to WhatsApp are doing the same
       thing over and over again'
11 //Apart from the above mentioned values and
       reasons, there are a total of 16 values
       identified that may be impacted (not all
       shown for brevity)
```

Listing 1. Scenario (iv) as an HVT Model

The modeller bases their decision on one or more *Examples* from past experience (one shown on Line 7) and on one or more *Reasons* (three shown in Lines 8-10). A reason lists out the impacted value and a respective explanation in the format *“negIfAbsent <name of HumanValue> because <explanation of Reason>”*. 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 including model elements, triggers, examples, and reasons. The resolution of disagreements and conflicts about the content of the catalogue is out of scope and left for future work.

The human value trigger system proposed in this paper compares every CDM element of the analyzed domain model with the triggers collected from past experiences. For example, assume that the scenarios detailed in Section III including the one shown in more detail in Listing 1 were added to the catalogue in the past and that now the WhatsApp domain model is analyzed for potential human value issues. In that case, the analyzed domain model contains the class "User", which is automatically matched by the HVTS with the trigger specified in Listing 1 for the catalogue. For each matched trigger, its examples and reasons for the impacted values are presented to the modeller so that the modeller can better understand the suggested element and decide whether the potential human value issue is indeed one that needs to be addressed in the context of the analyzed domain model. If it is, the modeller selects the suggestion and the HVTS consequently integrates the suggested model element into the analyzed domain model. To improve the automated matching process, we support the semantic detection of human value issues based on synonyms. The next section provides an analysis of synonym services.

V. Analysis of Synonym Services

This section provides a detailed explanation of the analysis performed on eight synonym services in Section V-A and the analysis performed on their various combinations in Section V-B to determine the best combination for the HVTS. Without a synonym service, the HVTS uses a syntactic, name-based approach to match a model element in the analyzed domain model with a triggering model element in the catalogue. While a matching approach based on Levenshtein distance [23] catches small typos, it cannot match different words with the same meaning. With a synonym service, this matching process also considers synonyms for the names of model elements.

A. Analysis of Single Synonym Services

Let us consider an example domain model where the class "User" is referred as "EndUser". 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-based and AI-based services. Table I 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.

TABLE I
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	AI Chatbot

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 [24] [25]. 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. Note that some other existing synonym services such as dictionary.com and thesaurus.com are not used in this paper as they are not accessible by an API.

Based on our motivating examples (WhatsApp, Airline Reservation System [14]), 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 depends on the context in which the word is used. Hence, we are evaluating the synonym services for each word in the context of its domain. The words used are *price, person, user, airline, delivered, status, text, message, active, privacy, story*, and *phoneNumber*. The list of words includes 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.

For each synonym service, we investigate the results based on the number of suitable words returned as a synonym. Each word is categorized as a suitable synonym of the given word based on the interpretation done by the first author of this paper. The second author 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 suitable words *(Suitable words % = (Suitable words / (Suitable words + Not suitable 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 ground 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)*.

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 suitable while *"damage",* and *"term"* are counted as not suitable. Similarly, for *"user"* we get *"exploiter"* from the synonym service which is considered as not suitable for our analysis. For words such as *"airline", "delivered",* and *"phone number"*, there are no results returned from this synonym service. Table II shows the results for the given list of words for Wordnet. Here, the percentage of suitable words found is calculated based on the number of suitable 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 suitable words by taking into account the lowest and the highest number of suitable words found which is 0-6 (*"airline"* – 0, *"story"* – 6). Lastly, we determined the average of suitable words by considering the number of suitable 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 II
DETAILED RESULTS FOR WORDNET

Word	Suitable Words (SW)	Not Suitable Words (NSW)	# SW	# NSW	% SW
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	– –

Table III 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 suitable 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 (i.e., the matching process is automatic and should not yield many suggestions that are false positives for the analyzed domain model). 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 suitable 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.

However, 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 (the one who creates the catalogue, not the one who uses it to analyze a domain model) to specify the synonyms explicitly as done by Singh *et al.* [26]. In that case, the synonyms are not dynamically retrieved during the matching process but rather specified at catalogue creation time. The matching process would then only access the list of synonyms in the catalogue. To implement this manual approach of providing synonyms, Oxford would be a good option because we could just present the long list from Oxford (range of 0-12 and average of 3.5) 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 III, 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

TABLE III
RESULTS FOR ALL SYNONYM SERVICES

Synonym Service	Criterion 1	Criterion 2	Range of Suitable Words	Average of Suitable Words
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

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 with a range of 1-13 and an average of 5.75 for an interactive, manual synonym approach during catalogue creation 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 from ChatGPT.

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.

B. 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 III, Criterion 1 is more than 10% 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 IV. We investigate each pair-wise combination of these five synonym services.

As indicated in the last two rows of Table IV, the results for Criterion 1 and Criterion 2 do not surpass the results of Word Dictionary + Webster. Therefore, we disregard these two combinations. 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.

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 suitable words. Therefore, we move forward with the first combination (Word Dictionary + Webster).

When we compare the results from Word Dictionary + Webster with Oxford + Webster, it is very clear from Table IV that Criterion 1 is 52% more for the first combination. Consequently, Word Dictionary + Webster is the best choice from Table IV, 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 III and Table IV. Criterion 2 and the average of suitable 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 suitable 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 HVTS.

To assure us that a combination with three synonym services does not yield a better result, we investigate the results of all combinations of three synonym services. The results for the combinations indicate that while it is possible for Criterion 1 to reach 100%, it comes at the expense of Criterion 2 which is now deemed to be too low (i.e., 17%) for the results to be useful for our proposed HVTS.

TABLE IV
RESULTS FOR THE PAIR-WISE COMBINATION OF SYNONYM SERVICES

Synonym Service	Criterion 1	Criterion 2	Range of Suitable Words	Average of Suitable Words
O + WD	67%	50%	0-3	0.75
O + WE	37%	67%	0-7	1.75
O + WN	38%	42%	0-2	0.50
O + G	40%	42%	0-2	0.33
WD + WE	**89%**	**42%**	**0-2**	**0.66**
WD + WN	83%	42%	0-3	0.75
WD + G	50%	50%	0-2	0.33
WE + WN	26%	42%	0-4	0.50
WE + G	75%	33%	0-2	0.33
WN + G	89%	25%	0-2	0.33

G...GloVe, O...Oxford,
WD...Word Dictionary, WE...Webster, WN...Wordnet

VI. Conclusion and Future Work

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 matches against past experiences with human value issues. With the proposed HVT (a domain-specific language called Human Value Trigger), different examples for human value issues together with the model elements for which the issues manifest themselves are captured in a catalogue along with a detailed scenario that explains how the presence or absence of the model element impacts the values.

In this paper, 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. 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 for the automated matching process in HVTS.

In the future, the matching algorithm of HVTS could be improved so that it works for patterns of multiple elements as triggers and suggestions instead of single elements. Pattern matching and model querying technologies such as OCL could be used for that purpose. Additional examples from past experiences could be collected using the grammar specified by HVT over time. The individual impacts of triggers could be translated and combined into a goal model to help the modeller assess more holistically the applicability of the suggestions for the analyzed domain model. Moreover, a user study could be conducted to assess the usefulness of the proposed system, possibly comparing HVTS with catalogue-based approaches for non-functional requirements. In addition to domain models, potential human value issues could be collected for other key RE modelling techniques such as goal models and workflow models to address human values throughout the software development process.

REFERENCES

[1] J. Whittle, M. A. Ferrario, W. Simm, and W. Hussain, "A case for human values in software engineering," *IEEE Software*, vol. 38, no. 1, pp. 106–113, 2021.

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

[3] W. Hussain, D. Mougouei, and J. Whittle, "Integrating social values into software design patterns," in *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, 2018, pp. 8–14.

[4] BBC News, "Instagram vows to remove all graphic self-harm images from site," 2019. [Online]. Available: https://www.bbc.com/news/uk-47160460

[5] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: Testing software for discrimination," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 498–510. [Online]. Available: https://doi.org/10.1145/3106237.3106277

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

[7] H. Perera, G. Mussbacher, W. Hussain, R. Ara Shams, A. Nurwidyantoro, and J. Whittle, "Continual human value analysis in software development: A goal model based approach," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020, pp. 192–203.

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

[9] Object Management Group, "OMG® Unified Modeling Language® (OMG UML®)," Dec 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF

[10] N. S. B. Sani, "Lab 3: Introduction to domain modeling and class diagram," 2009-2010. [Online]. Available: https://norsamsiah.files.wordpress.com/2010/01/lab-003-domain-modeling1.pdf

[11] D. Mougouei, H. Perera, W. Hussain, R. Shams, and J. 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*, ser. ESEC/FSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 780–784. [Online]. Available: https://doi.org/10.1145/3236024.3264843

[12] H. Perera, W. Hussain, D. Mougouei, R. A. Shams, A. Nurwidyantoro, and J. Whittle, "Towards integrating human values into software: Mapping principles and rights of gdpr to values," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 404–409.

[13] J. Whittle, "Is your software valueless?" *IEEE Software*, vol. 36, no. 3, pp. 112–115, 2019.

[14] G. Mussbacher, W. Hussain, and J. Whittle, "Is there a need to address human values in domain modelling?" in *2020 IEEE Tenth International Model-Driven Requirements Engineering (MoDRE)*, 2020, pp. 73–77.

[15] H. Perera, W. Hussain, J. Whittle, A. Nurwidyantoro, D. Mougouei, R. A. Shams, and G. 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*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 409–420. [Online]. Available: https://doi.org/10.1145/3377811.3380393

[16] R. A. Shams, W. Hussain, G. Oliver, A. Nurwidyantoro, H. Perera, and J. 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*, ser. ICSE-SEIS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 53–62. [Online]. Available: https://doi.org/10.1145/3377815.3381382

[17] W. Hussain, H. Perera, J. Whittle, A. Nurwidyantoro, R. Hoda, R. A. Shams, and G. Oliver, "Human values in software engineering: Contrasting case studies of practice," *IEEE Transactions on Software Engineering*, vol. 48, no. 5, pp. 1818–1833, 2022.

[18] A. Nurwidyantoro, M. Shahin, M. Chaudron, W. Hussain, H. Perera, R. A. Shams, and J. 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)*, ser. ESEM '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3475716.3475770

[19] W. Hussain, M. Shahin, R. Hoda, J. Whittle, H. Perera, A. Nurwidyantoro, R. A. Shams, and G. Oliver, "How can human values be addressed in agile methods? a case study on safe," *IEEE Transactions on Software Engineering*, vol. 48, no. 12, pp. 5158–5175, 2022.

[20] A. Nurwidyantoro, M. Shahin, M. R. Chaudron, W. Hussain, R. Shams, H. Perera, G. Oliver, and J. Whittle, "Human values in software development artefacts: A case study on issue discussions in three android applications," *Information and Software Technology*, vol. 141, p. 106731, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584921001828

[21] Xtext, "Website." [Online]. Available: https://www.eclipse.org/Xtext/

[22] TouchCORE, "Website." [Online]. Available: http://touchcore.cs.mcgill.ca/

[23] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[24] GloVe: Global Vectors for Word Representation, "Website." [Online]. Available: https://nlp.stanford.edu/projects/glove/

[25] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[26] P. Singh, "Domain modeling mistake detection system," McGill University, Canada, 2022. [Online]. Available: https://escholarship.mcgill.ca/concern/theses/5x21tm741