TranspLanMeta: A Metamodel for TranspLan Modeling Language

Deniz Cetinkaya[1][a] and Mahmood Hosseini[2]

[1]*Department of Computing & Informatics, Bournemouth University, Poole, U.K.*

[2]*JPMorgan and Chase, Bournemouth, U.K.*

Keywords: Requirements Engineering, Transparency Requirements, Requirements Modeling, Transparency, TranspLan, Metamodeling, Domain-specific Modeling.

Abstract: Transparency and transparent decision making are essential requirements in information systems. To this end, a modeling language called TranspLan has been proposed. TranspLan is a domain-specific modeling language which is designed for the purpose of analysing and modeling transparency requirements in information systems. This paper presents a metamodel for transparency requirements modeling. We are introducing a model-driven approach to TranspLan language specifications to facilitate the use of the language more efficiently in real life cases. Metamodeling is an effective method for formally defining domain specific languages and moving from specifications to computer-aided modeling. In this paper, we propose a metamodel for TranspLan modeling language which is called as TranspLanMeta. The metamodeling process helps us to transfer TranspLan language specifications into a machine-readable format. The metamodel has been developed with GME (Generic Modelling Environment), which is a configurable toolkit for creating domain-specific modeling and program synthesis environments. By developing TranspLanMeta with GME, an automatically-generated modeling tool for TranspLan language is provided as well. In this way, an effective approach for accelerating software development is followed and the auto-generated modeling editor is used to define various models. This work provides a formal and practical solution for transparency modeling and a well-defined basis for using transparency requirements models in the further steps of the business process.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Transparency and transparent decision making are becoming one of the important non-functional requirements in information systems. In the light of new transparency rules and regulations, the need for tools and languages that enable transparency modeling is increasing. TranspLan is a domain-specific modeling language which is designed for analysing and modeling transparency requirements in information systems (Hosseini, 2016). The language was originally defined as a set of specifications without any tool support. Introducing a metamodel and tool support for TranspLan will open new ways for its usage and practicality. Hence, we are introducing a model-driven development (MDD) approach to TranspLan language specifications to facilitate the use of the language more efficiently in real-life cases in this paper.

MDD is a software development approach which aims to face the challenges of software development process through representing the essential aspects of a system as models and generating the final source code from these models (semi-)automatically (Rodrigues da Silva, 2015; Sommerville, 2016). In MDD literature, the most common method to define modeling languages is metamodeling. Metamodeling is an effective and practical approach for defining domain-specific modeling languages. Model-driven approaches in software and systems engineering use well-defined modeling languages to specify models and apply model transformations to automatically generate new models or source code from an initial design model (Cetinkaya et al., 2015).

In this paper, we propose a metamodel for TranspLan modeling language. The metamodel has been developed with GME (Generic Modelling Environment), a configurable toolkit for creating domain-specific modeling and program synthesis environments. A modeling editor for transparency modeling is automatically generated and configured by using the metamodel. In this way, an effective approach for developing a computer-aided, formal and practical solution for transparency modeling is followed and

[a] https://orcid.org/0000-0002-1047-0685

Cetinkaya, D. and Hosseini, M.
TranspLanMeta: A Metamodel for TranspLan Modeling Language.
DOI: 10.5220/0010202201470154
In *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2021)*, pages 147-154
ISBN: 978-989-758-487-9
Copyright © 2021 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

the auto-generated modeling editor is used to define various models. As an example, a sample model for defining transparency requirements for an email service provider is shown in this paper. This work helped us to transfer TranspLan language specifications into a machine-readable format and it provided a sound basis for using transparency requirements models in the further steps of the business process.

The paper is structured as follows: Section 2 introduces TranspLan modeling language. Section 3 presents the metamodel for TranspLan, which we call TranspLanMeta. Section 4 explains a case example. Section 5 discusses the benefits and challenges of this work. Finally, Section 6 presents the conclusions and the future work.

2 TranspLan: A TRANSPARENCY MODELING LANGUAGE

TranspLan is a domain-specific modeling language for the identification, capturing, and analysis of transparency requirements of stakeholders in an information system. The design of TranspLan is based on four proposed transparency reference models (Hosseini et al., 2017) as below:

1. **Transparency Actors Wheel:** which illustrates transparency stakeholders and the information flow between them
2. **Transparency Depth Pyramid:** which illustrates the depth and meaningfulness of information
3. **Transparency Achievement Spectrum:** which illustrates the steps necessary to achieve a useful transparency
4. **Transparency Information Quality:** which illustrates the information quality dimensions for transparency provision

These reference models are in turn based on an extensive literature study on transparency in different fields of study such as politics, economy, and journalism (Hosseini, 2016). TranspLan, as a modeling language, was first proposed in (Hosseini et al., 2016). Then, another study was made on TranspLan and its evaluation, which was later published in (Hosseini et al., 2018). This study also tweaked the modeling language, and more specifically, its visual diagram, for a better and clearer visual and semantic representation.

TranspLan consists of StakeHolders' Information Exchange Layout Diagram (also called Shield diagram) for the visual representation of information exchanges amongst stakeholders and their transparency requirements. TranspLan is also accompanied by two descriptive specification models for information elements and stakeholders, called INFOrmation eLEment Transparency Specification (also called Infolet specification) and Stakeholders' Information Transparency REQuirements Specification (also called Sitreq specification), respectively. These specification models explain the information elements and the stakeholders with their elicited transparency requirements in the Shield diagram.

2.1 Modeling Constituents and Representations

The TranspLan language is mainly built based on three different constituents: *stakeholders*, *information elements*, and the *relationships* between stakeholders and information elements. Relationships can be decomposed using *decomposition relations*. An *Information Exchange (IEX)* is a combination of all these constituents and illustrates the flow of information amongst different stakeholders. These constituents are described as follows.

- **Stakeholders** are the people, departments, organisations, etc., which are involved in providing, receiving, or requesting transparency in any information exchange amongst stakeholders. When categorising stakeholders, they are commonly represented as one entity, e.g., Student or Finance Department. However, the exchanged information within an information exchange system may concern all the stakeholders within that system (referred to as *All* in TranspLan), and it may also concern the public audience.
- **Information elements** are pieces of information exchanged amongst stakeholders. Stakeholders' transparency requirements affect the way information elements should be formed and presented to other stakeholders. An Information Element (IE) has a type, which is related to their transparency meaningfulness. These types can be *data* type, *process* type, or *policy* type. Examples of information elements are privacy policy statements, mortgage documents, and application forms.
- **Stakeholder-information relationships** exist between stakeholders and information elements, and they describe how the IE is associated with the stakeholder. The *production* relationship denotes that the stakeholder produces the IE for other stakeholders. The *obligation* relationship denotes that the stakeholder provides the IE based on coercive supply or requests the IE based on legal demands. The *optionality* relationship denotes that

the stakeholder provides the IE based on voluntary supply or requests the IE based on personal demands. The *restriction* relationship denotes that the IE should not be available to the stakeholder. The *undecidedness* relationship denotes that the relationship between the stakeholder and the IE is not known or decided yet. For example, a bank (i.e., the stakeholder) must provide a mortgage guide (i.e., information element) to their customer (i.e., the other stakeholder). Therefore, there will be two relationships, one provision relationship between the bank and the mortgage document and one obligation document between the mortgage document and the bank customer.

- **Decomposition relations** exist between relationships and can be one of the following: *AND* relation, *OR* relation, and *XOR* (exclusive or) relation.
- **Information exchanges** illustrate the flow of information from an information provider to an information receiver or requester. For example, the bank, their customer, and the mortgage document together constitute an information exchange between these two stakeholders. An information exchange system is a collection of all information exchanges in an information system.

2.2 TranspLan Mathematical Definition

The *TranspLan* language and its constituents can be defined using the ordinary mathematical language as follows:

[Information element] Let $IE = \{ie_1, ie_2, ..., ie_m\}$ be the set of information elements, and IE_Label and IE_Name be sets of unique labels and names respectively. Every $ie_i \in IE$ can be defined as follows:

$IE = \{ie \mid ie = (ietype, ielabel, iename, ieused) \wedge ietype \in IE\_type \wedge ielabel \in IE\_label \wedge iename \in IE\_name \wedge ieused \subset ielabel\}$

$IE\_type = \{data, process, policy\}$

[Stakeholder] Let $S = \{s_1, s_2, ..., s_n\}$ be the set of stakeholders, $IE = \{ie_1, ie_2, ..., ie_m\}$ be the set of information elements, and $R = \{r_1, r_2, ..., r_l\}$ be the set of stakeholder-information relationships. The set of stakeholders and two subsets of S, called PS and RS, can be defined as follows:

$S = \{s \mid s \text{ is a stakeholder}\}$

$PS = \{s \mid s \in S \wedge ie \in IE \wedge (s, ie, production) \in R\}$

$RS = \{s \mid s \in S \wedge ie \in IE \wedge rt \in \{obligatory, optional, restricted, undecided\} \wedge (s, ie, rt) \in R\}$

[Stakeholder-information relationship] Let $R = \{r_1, r_2, ..., r_l\}$ be the set of relationships where each relationship is between stakeholder $s_i \in S$ and information element $ie_j \in IE$. Every $r_i \in R$ can be defined as follows:

$R = \{r \mid r = (s, ie, rtype) \wedge s \in S \wedge ie \in IE \wedge rtype \in R\_type\}$

$R\_type = \{production, obligatory, optional, restricted, undecided\}$

[Decomposition relation] Let $Rel = \{rel_1, rel_2, ...rel_k\}$ be the set of relations where each relation is between two or more relationships $R_1, R_2, ..., R_j \in R$. Every $Rel_i \in Rel$ can be defined as follows:

$Rel = \{rel \mid rel = (r_1, r_2, .., r_j, reltype) \wedge r_1, r_2, .., r_j \in R \wedge reltype \in Rel\_type\}$

$Rel\_type = \{and, or, xor\}$

[Information exchange] Let $IEX = \{iex_1, iex_2, ..., iex_t\}$ be the set of information exchanges amongst stakeholders where one stakeholder $s \in PS$ produces some information elements $IESet \subset IE$ that is received or requested by a group of other stakeholders $RSSet \subset RS$ and $s \notin RSSet$. Every information exchange iex_i can be defined as follows:

$IEX = \{iex \mid iex = ((s_i, ie_i, r_i), (s_j, ie_i, r_j)) \wedge s_i \in PS \wedge s_j \in RS \wedge r_i = production \wedge (s_i, ie_i, r_i), (s_j, ie_i, r_j) \in R\}$

2.3 Shield Diagram

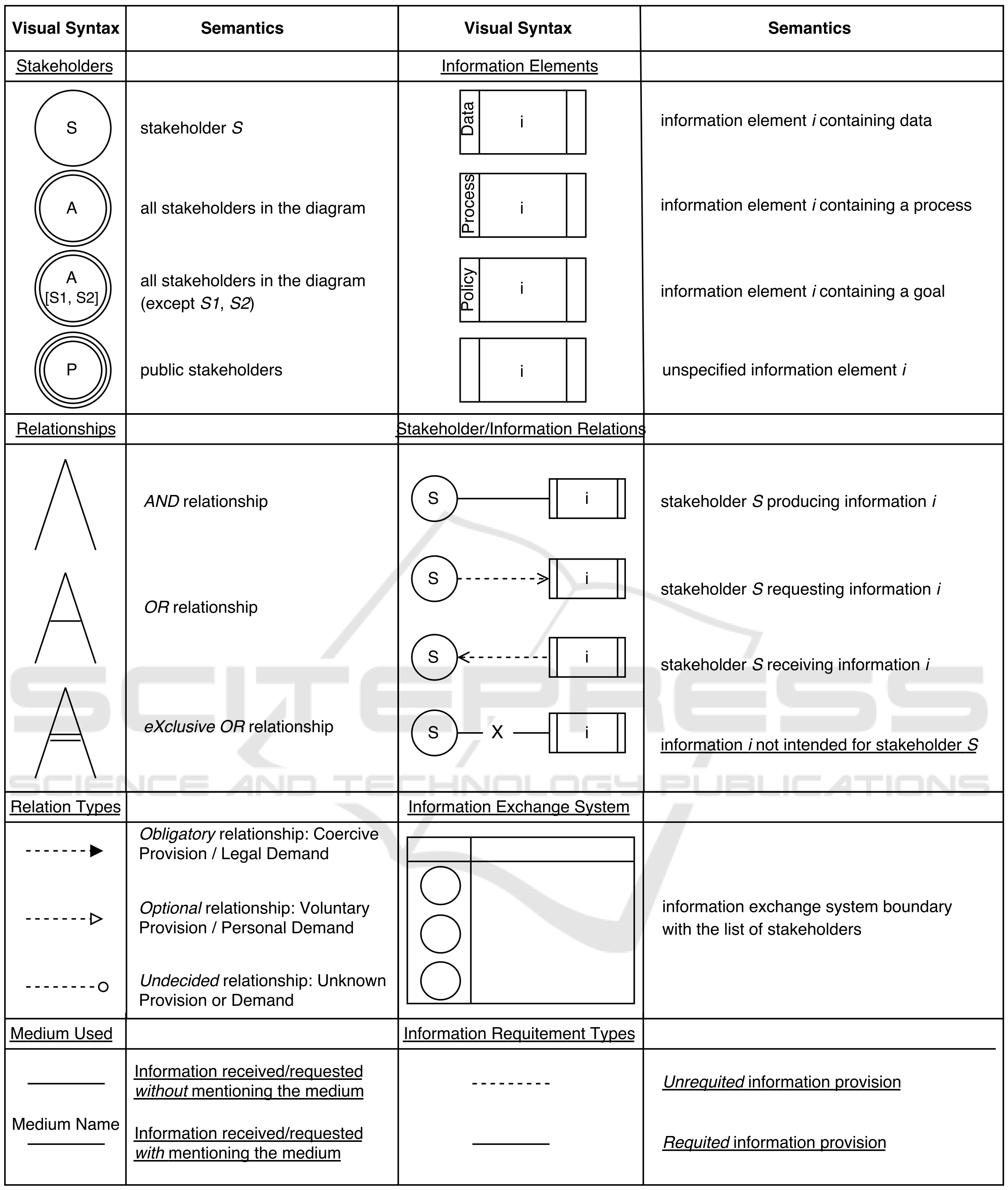
The Shield diagram is the graphical representation of the TranspLan language. The constituents of the TranspLan language can be illustrated in the Shield diagram as follows.

2.3.1 Stakeholders

Stakeholders are captured in one of the four following ways: (1) One stakeholder; (2) All stakeholders within an information exchange system (for the purpose of facilitating a more efficient, clutter-free visual design); (3) All stakeholders further enriched by the *exclusion* notation (for referring to those stakeholders who are excluded from the information exchange); (4) Public stakeholders (for referring to the public, i.e., all stakeholders inside and outside the information exchange system under study) (See Figure 1).

2.3.2 Information Elements

Information elements capture the type of IE (i.e., data, process, and policy), a unique IE label, its name, and a list of all the other IE tags which use, partly or completely, the current IE (See Figure 1).



Visual Syntax	Semantics	Visual Syntax	Semantics
Stakeholders		Information Elements	
S	stakeholder *S*	Data i	information element *i* containing data
A	all stakeholders in the diagram	Process i	information element *i* containing a process
A [S1, S2]	all stakeholders in the diagram (except *S1*, *S2*)	Policy i	information element *i* containing a goal
P	public stakeholders	i	unspecified information element *i*
Relationships		Stakeholder/Information Relations	
	AND relationship	S i	stakeholder *S* producing information *i*
	OR relationship	S i	stakeholder *S* requesting information *i*
		S i	stakeholder *S* receiving information *i*
	eXclusive OR relationship	S X i	information *i* not intended for stakeholder *S*
Relation Types		Information Exchange System	
	Obligatory relationship: Coercive Provision / Legal Demand		information exchange system boundary with the list of stakeholders
	Optional relationship: Voluntary Provision / Personal Demand		
	Undecided relationship: Unknown Provision or Demand		
Medium Used		Information Requitement Types	
	Information received/requested *without* mentioning the medium		*Unrequited* information provision
Medium Name	Information received/requested *with* mentioning the medium		*Requited* information provision

Figure 1: Building blocks of *Shield* and their interpretations (Hosseini et al., 2018).

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

2.3.3 Stakeholder-information Relationships

Stakeholder-information relationships capture whether the stakeholder is producing the information, or receiving the information on an obligatory or optional basis. They also capture currently unknown relationships and restricted relationships. Arrows are intentionally chosen to be dotted to emphasise that information flow may or may not serve its transparency purpose because its usefulness must be decided through complicated procedures and involvement with stakeholders (See Figure 1).

2.3.4 Decomposition Relations

Decomposition relations describe the relationship amongst relationships and can be *AND* (the default relation), *OR*, and *XOR* (See Figure 1).

2.3.5 Information Exchange System

An information exchange system captures the following four parts: its name, its description and extra notes, a list of all stakeholders in the information exchange system including two pre-defined *All* and *Public* stakeholders, and all the information exchanges amongst the stakeholders (See Figure 1).

3 TranspLanMeta: A METAMODEL FOR TranspLan LANGUAGE

TranspLan specification defines an efficient language to represent transparency requirements. Its formal definition provides a sound framework for transforming the requirements into further models. Introducing model-driven approaches and defining a metamodel for TranspLan language will improve its usability. In this section, we introduce a metamodel for the TranspLan language given in Figure 2.

Metamodeling is the process of complete and precise specification of a modeling language in the form of a metamodel (Cetinkaya et al., 2015). Once the metamodel is defined, then it can be used to specify models in that language. Metamodeling environments, sometimes called as domain-specific modelling environments, provide powerful tools to define metamodels as well as to develop modeling environments for those new modeling languages (Emerson et al., 2008). A model is an abstraction of a system, which shows the main properties of the system and excludes unnecessary details, which in turn leads to reducing the complexity of the system (Syriani et al., 2013). A modeling language consists of an abstract syntax, concrete syntax, and semantics (Atkinson and Kühne, 2002; Cetinkaya et al., 2015).

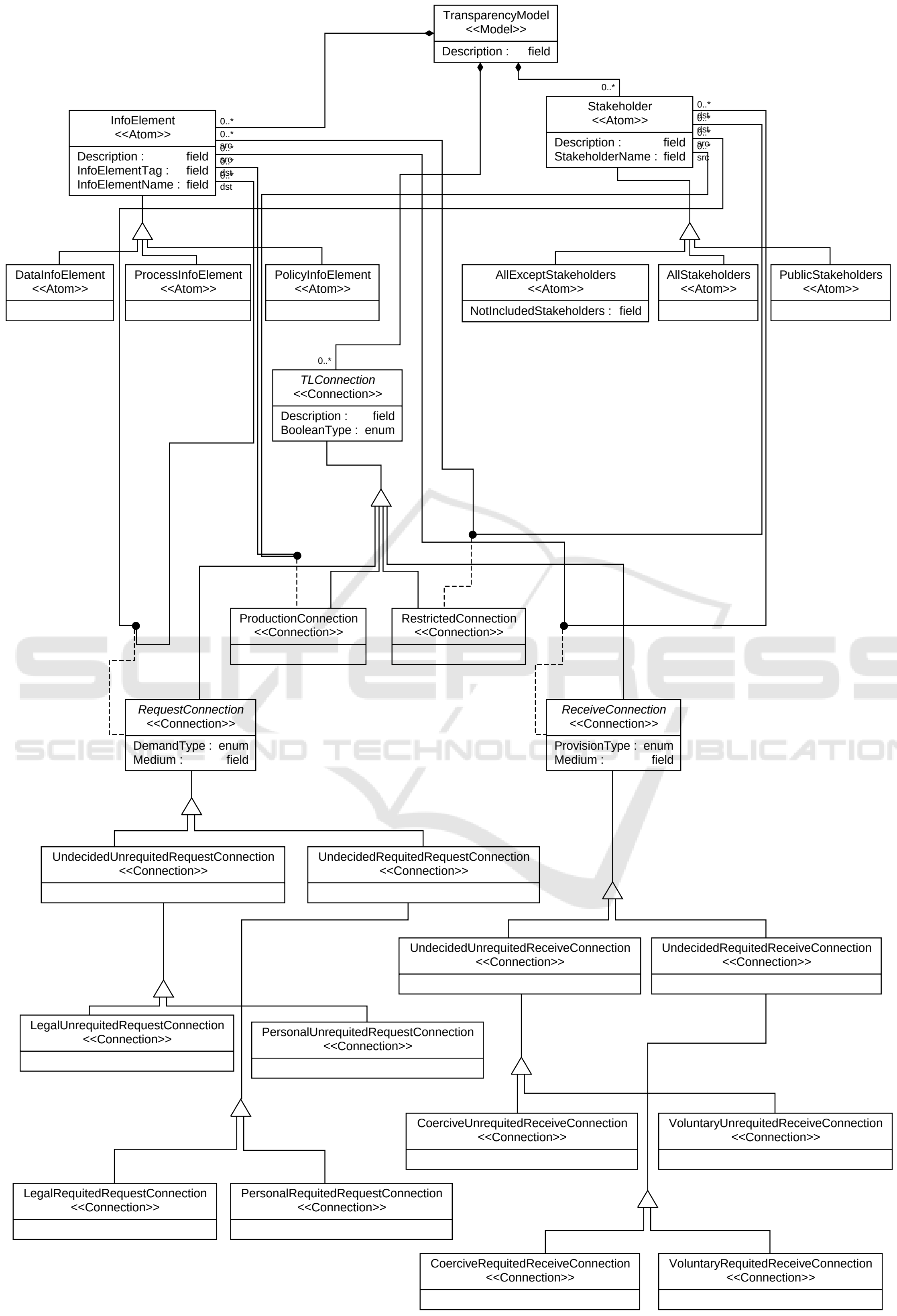
The metamodel, namely TranspLanMeta, is defined with GME (Generic Modelling Environment), which is a configurable metamodeling toolkit for defining domain-specific modeling languages and creating modeling environments (Ledeczi et al., 2001; Davis, 2003). In GME, metamodels are specified as modeling paradigms that represents the application domain. The metamodels contain all the syntactic and semantic elements to construct the models with domain specific concepts as well as the relationships among those concepts. Besides, the rules for the composition of the concepts or modeling elements during the construction of the models are defined. Then, the metamodels can be used to automatically generate the target domain-specific modeling environment.

During the development of the metamodel, the building blocks of the Shield are defined within the metamodeling environment. First, stakeholders and information elements are specified as atomic modeling elements and then the relationships are introduced as connections. Overall model is specified as a coupled modeling element and called as *Transparency-Model*. Figure 2 shows the metamodel which is using the notation of GME. Atomic elements are shown as <<*Atom*>>and coupled elements are shown as <<*Model*>>in the metamodel.

Connections are shown as <<*Connection*>>in the metamodel. Different relation types are defined as different connections whereas they are grouped with inheritance relation. ProductionConnection is defined from Stakeholder to InfoElement. RestrictedConnection is defined from InfoElement to Stakeholder. Request and Receive connections has six different subtypes to better express the variations. Similarly, stakeholder and information element types are defined as different atomic elements and they are grouped with inheritance as well. We preferred this approach rather than defining a type attribute because with specific modeling elements and connections we could handle the variations in a more effective and practical way. The resulting modeling editor became more usable and we could easily identify the type of the elements.

We applied a prototyping approach where the deliverables were built up incrementally. The modeling editor provides a diagram view as well as an XML based representation. We defined attributes for some elements, as well as different decoration types for better usability and to be compliant with the language definition. The resulting editor is a full functional drag and drop style modeling editor. Some attributes are assigned default values, for example default information requitement type is *unrequited information provision* since if the type is *requited* then this should be defined by the modeler specifically. Figure 3 shows a screenshot of the modeling editor.

OCL (Object Constraint Language) based axiom checking supports the model validation. Figure 4 illustrates the constraint definition and usage. Figure 4.a shows how the constraints are attached to metamodel in the metamodeling stage and Figure 4.b shows an example constraint definition by updating the attributes of the constraint. Figure 4.c presents the output when the constraint is triggered in the modeling stage. The example is *NoEmtpyName* constraint



SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

Figure 2: TranspLan language metamodel.

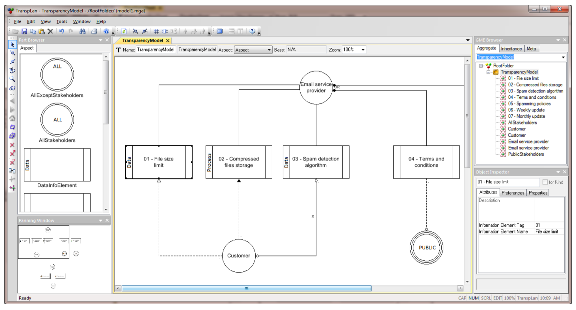
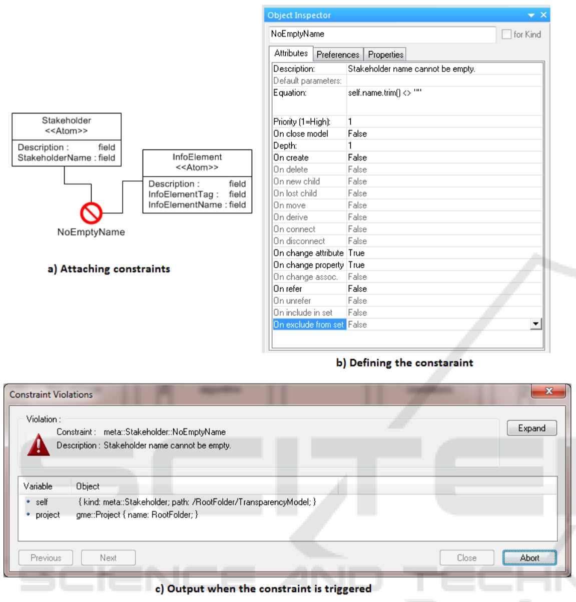


Figure 3: Screenshot of the TranspLan modeling editor.



Object Inspector

NoEmptyName for Kind

Attributes | Preferences | Properties

Description:	Stakeholder name cannot be empty.
Default parameters:	
Equation:	self.name.trim() <> ""
Priority (1=High):	1
On close model	False
Depth:	1
On create	False
On delete	False
On new child	False
On lost child	False
On move	False
On derive	False
On connect	False
On disconnect	False
On change attribute	True
On change property	True
On change assoc.	False
On refer	False
On unrefer	False
On include in set	False
On exclude from set	False

Constraint Violations

Violation :
Constraint : meta::Stakeholder::NoEmptyName
Description : Stakeholder name cannot be empty.

Variable	Object
self	{ kind: meta::Stakeholder; path: /RootFolder/TransparencyModel; }
project	gme::Project { name: RootFolder; }

Previous Next Close Abort

Figure 4: An example constraint for TranspLanMeta.

and checks that a Stakeholder cannot have an empty name. Similarly, we are checking that tag numbers are unique with OCL. Some of the rules are guaranteed by using the metamodel such as each information element has one type. We are currently working on the constraints and improving the metamodel due to some of the OCL rules are partially running. However, the mechanism has been tested and works properly, we only need to revise the OCL rules.

4 CASE EXAMPLE

The auto-generated modeling editor is used to define various models and tested with different small and mid scale scenarios. As an example, a sample transparency model for defining the transparency requirements for an email service provider is shown in this paper which was previously presented in (Hosseini et al., 2018). We have chosen the same example to be able to compare the outputs. Figure 5 shows the sample model. The modeling editor provides an environment to be able to exactly replicate the previously drawn models. The metamodel covers all of the core modeling elements and relationships. Table 1 shows how the TranspLanMeta metamodel covers the concepts of the TranspLan language.

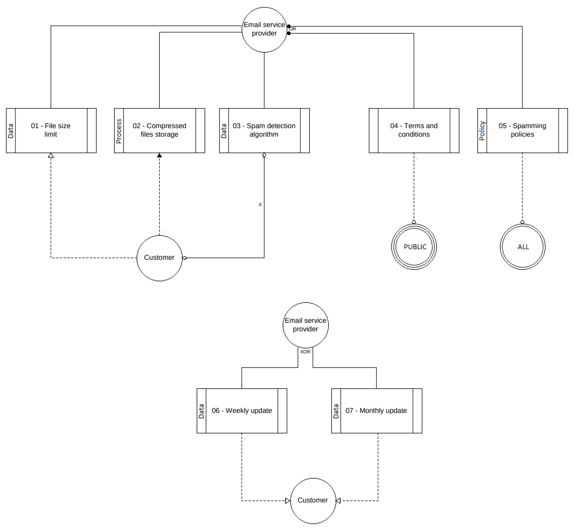


Figure 5: Sample model developed with the TranspLan modeling editor.

Table 1: Language constructs vs. TranspLanMeta elements.

Language construct	Metamodel element
Stakeholder	Stakeholder (Atomic)
Information element	InforElement (Atomic)
Producing information	ProductionConnection
Requesting information	RequestConnection
Receiving information	ReceiveConnection
Restricted information	RestrictedConnection

All stakeholder types are implemented with different icons and various information element types are implemented with the data type as a vertical text on the left. All relationship types are implemented as well, with their associated line styles such as dashed or solid as well as arrow headings. The relationships are checked for their directions, for example a production relation is only possible from a Stakeholder to Information Element.

5 DISCUSSION

In this section, we would like to discuss the benefits of TranspLanMeta and challenges in our work. First of all, this work helped us to transfer the TranspLan language specification into a machine readable format. We followed a well-established model driven approach and effectively developed a practical solu-

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

tion for transparency modeling. The auto-generated modeling editor is used to test both the language specification and the metamodel.

During the metamodeling process, we needed to make some decisions. In many cases, the decision making process was straightforward and supported by literature. But, in a few occasions we needed to try all alternatives and choose the best option. For example, making all relation types as different connection classes was initially seemed to be irrelevant. However, after some tests and modeling exercises we preferred this approach rather than defining a type attribute for the base class. Because with specific connections, the resulting modeling editor became more usable and we could easily identify the type of the elements. In this way, we could decorate different line types associated with different relation types such as dashed or solid, empty or arrow headed, etc.

Additionally, we would like to mention that we highlighted the ambiguity problem about AND, OR, and XOR relationships present in the language specification. Currently, this issue is explained in (Hosseini et al., 2018) and it is assumed to use priority between logical operators as 1)XOR 2)OR and 3)AND. Although, this assumption solves the issue temporarily, we need a better mechanism to represent the mathematically equivalent relations. In TranspLanMeta, we solve this ambiguity problem with grouping the relations with an attribute. This is equal to using parentheses and resulting mathematical equation can be automatically derived.

6 CONCLUSION

This paper presented a metamodel for transparency modeling and a modeling tool with a formal basis. The metamodel is based on TranspLan language, which is proposed for defining and analysing transparency requirements in information systems (Hosseini et al., 2016). We applied a prototyping approach where the deliverables were built up incrementally. The modeling editor provides a diagram view as well as an XML based representation. OCL based axiom checking supports further model validation.

As a future work, we will develop a plug-in attached to our metamodel to automatically generate (fully or partially) Sitreq and Infolet information. In addition, we would like to improve TranspLan language specifications to resolve the ambiguity in logical priorities by using the findings throughout the metamodeling process.

REFERENCES

Atkinson, C. and Kühne, T. (2002). Rearchitecting the UML infrastructure. *ACM Trans. on Modeling and Computer Simulation*, 12(4):290–321.

Cetinkaya, D., Verbraeck, A., and Seck, M. D. (2015). Model continuity in discrete event simulation: A framework for model-driven development of simulation models. *ACM Transactions on Modeling and Computer Simulation*, 25(3):17:1–17:24.

Davis, J. (2003). GME: The generic modeling environment. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, pages 82–83. ACM.

Emerson, M., Neema, S., and Sztipanovits, J. (2008). Metamodeling languages and metaprogrammable tools. In *Handbook of Real-Time and Embedded Systems*, pages Chapter 33, 1–16. Taylor and Francis Group.

Hosseini, M., Shahri, A., Phalp, K., and Ali, R. (2016). A modelling language for transparency requirements in business information systems. In *Int. Conference on Advanced Information Systems Engineering*, pages 239–254. Springer.

Hosseini, M., Shahri, A., Phalp, K., and Ali, R. (2017). Four reference models for transparency requirements in information systems. *Requirements Engineering*, pages 1–25.

Hosseini, M., Shahri, A., Phalp, K., and Ali, R. (2018). Engineering transparency requirements: A modelling and analysis framework. *Information Systems*.

Hosseini, S. M. M. (2016). *Engineering of transparency requirements in business information systems*. PhD thesis, Bournemouth University.

Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., and Volgyesi, P. (2001). The generic modeling environment. In *Int. Workshop on Intelligent Signal Processing (WISP'01)*. IEEE.

Rodrigues da Silva, A. (2015). Model-driven engineering. *Comput. Lang. Syst. Struct.*, 43(C):139–155.

Sommerville, I. (2016). *Software Engineering*. Pearson, 10th edition.

Syriani, E., Gray, J., and Vangheluwe, H. (2013). *Domain Engineering: Product Lines, Languages, and Conceptual Models*, chapter Modeling a Model Transformation Language, pages 211–237. Springer Berlin Heidelberg.