Modelling Data Interaction Requirements
A Position Paper

Sagar Sen, Jose Luis de la Vara, Arnaud Gotlieb
Certus V&V Center
Simula Research Laboratory
Lysaker, Norway
Email: {sagar,jdelavara,arnaud}@simula.no

Arnab Sarkar
Institute Of Engineering And Management
West Bengal University Of Technology
Kolkata, India
Email: ArnabSarkarCalcutta@gmail.com

***Abstract*—Data-intensive information systems constitute the backbone of e-commerce and e-governance services running worldwide. Structured data is a central artefact in these information systems. Requirements for structure in data are typically modelled in a database schema. However, information system behaviour is often a function of interactions that *cross-cut* database features such as field values in different tables. For instance, consultants at the Norwegian Customs and Excise reveal that taxation rules are triggered due to *data interactions* between 10,000 items, 88 country groups, and 934 tax codes. There are about 12.9 trillion possible three-wise interactions of which only about 220,000 interactions are used in reality as customs rules. Therefore, we ask, how can we model data interaction requirements to further *bound the input domain* of an information system? In this position paper, we address this question by modelling data interaction requirements using *classification tree models*. We also present different applications of data interaction requirements in the development of information systems.**

***Index Terms*—data requirements; data interactions; modelling the data perspective; classification tree model;**

I. Introduction

Data-intensive software systems are increasingly prominent in driving global processes such as scientific and medical research, e-governance, and social networking. Large amounts of data is collected, processed, and stored by these systems in *databases*. Requirements for developing these specific types of information systems have three principal perspectives: data, functional and behavioural according to Pohl and Rupp [1]. Modelling requirements for the *data perspective* is the overall area of this paper. In the data perspective, input and output data is usually modelled using conceptual models such as entity-relationship diagrams, database schemas, and class diagrams. These models can later be used for creation of a database. Nevertheless, these modelling languages do not support modelling of interactions between data elements that cross-cut several database tables and their fields.

We define the term *data interaction* as a set of values for fields in a database schema. These fields can be from any table of the schema. *In this paper, we aim to show the importance of modelling data interaction requirements and how we have started to deal with their modelling*. We will illustrate this with an industrial case study at the Norwegian Customs and Excise (NCE) department. The NCE uses the TVINN system to processes about 30,000 *customs declarations* a day. The TVINN system at the NCE has been in operation since 1988 and is a massive database application that processes declarations sent as standard EDIFACT messages [1]. These messages are encoded from information in an user interface for customs officers and companies concerned with import and export in Norway. A sophisticated batch application called EMIL processes about 30,000 declarations per day to notify customs officers about the correctness of the declarations. The correctness is verified based on a large set of well-formedness rules and customs laws and regulations. The customs laws and regulations are typically a three-wise function of interaction between fields values (from different tables) such as 10,000 items, 88 country groups, and 934 tax codes. This gives rise to about 12.9 trillion three-wise interactions of which only around 220,000 interactions are used in customs rules. Consequently, more than 99.99% of all three-wise interactions are irrelevant to the TVINN information system.

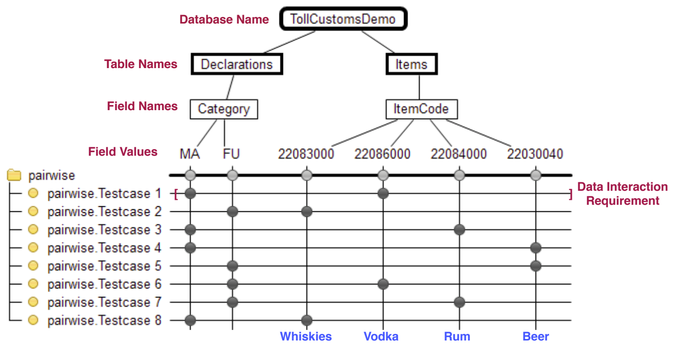

Fig. 1. Model of Data Interaction Requirements

Reducing the unrealistic possibilities in inputs is an essential requirement for a critical information system such as TVINN. A single error in the computation of tax due to incorrect choice of rules could lead to catastrophic consequences in terms of public image. Therefore, it is necessary to model data interactions to *bound the data* processed by the system.

We propose the use of *Classification Tree Models* or diagrams to model data interaction requirements. These requirements are specified on an existing *data model* such as

[1]http://www.unece.org/trade/untdid/welcome.html

a database schema. A classification tree model presents a high-level visual and structured presentation of set of choices in field values from different tables of a database. A set of chosen values for the different fields represents a data interaction requirement. We use the tool CTE-XL [2] to specify classification tree models that show data interactions. We illustrate an example classification tree model in Figure 1. The example model is used to model all two-wise or pairwise interactions between four imported alcohol types and the declaration categories MA (manual intervention) and FU (finalized declaration). The first data interaction requirement states that the information system rules database must contain an interaction between alcohol type vodka (22086000) and manual (MU) verification of a declaration. The paper presents several applications of modelling data interaction requirements with classification tree models.

The paper is organized as follows. In Section II, we present a detailed overview of the case study at the NCE. We use the case study to explain the general approach of modelling data interactions. In Section III, we present the data interaction modelling approach and its potential applications. Section IV discusses the related work, while we conclude in Section V.

II. Industrial Case Study: Norwegian Customs and Excise Department

We describe our industrial case study from the NCE as illustrated in Figure 2(a). As mentioned above, the system under study is TVINN. An overview of TVINN process flow is presented in Figure 2(a) and an official description is available on its website[2].

Customs officers and industries associated with import and export create declarations at Norwegian ports of entry. These declarations are encapsulated in the EDIFACT standard for business communication. A declaration is encapsulated as an EDIFACT CUSDEC message. These messages are sent to TVINN's central server where they are processed by a sophisticated batch application called EMIL. EMIL parses EDIFACT messages and verifies them against well-formedness rules. It then verifies if the declared amount is accurately computed based on a statistical value for an item. These rules depend on numerous factors such as (a) 260 countries of origin divided in 88 country groups. (b) over 160 currencies. (c) around 900 tax code groups, and (d) a list of more than 10,000 items. A declaration can be categorized into six different categories based on EMIL's computation, the simplest categories being *complete* and *reject*. The response from TVINN is sent back as an EDIFACT CUSRES message to customs officer or industry.

Rules in TVINN evolve on a regular basis (approximately, every six months), depending on new governmental policies, sanctions, and change in political parties. TVINN is also affected with time-bounded rules created by customs officers. These rules exist for a short period of time. For instance, a customs officer could decide to thoroughly check 20 trucks coming from a nation X in civil war and he/she would create a rule to check all the trucks from the nation for the following three hours. These kinds of rules are called *mask control* and will disappear after a fixed time limit. These rules can change on an everyday basis without anticipation, making TVINN a highly dynamic system.

TVINN is a complex and dynamic database application with a high number of requirements. Modelling and validating data interaction requirements in TVINN is the principal challenge presented in this paper. This activity, will help address the following challenges in the information system:

Data-processing Rule Validation Records obtained from real-world transactions, such as customs declarations, can validate a realistic subset of all possible *data processing rules*, such as taxation rules. However, they often do not cover combinations of values that are very rare or exceptional. Modelling data interactions will help create data records with specific combinations.

Very Large Set of Records: Accumulating information from real-world transactions can easily give rise to an ever-growing set of data records. Many of these records share similarities and hence are redundant for the purpose of validating requirements satisfaction in the information system. This means that the same data interactions may occur multiple times. Cost-effective system validation will require a selection of a minimal set of records that precisely satisfy all data interaction requirements. Models of data interactions can help identify such a minimal set.

Constantly Changing Rules: Information systems such as those in e-governance evolve on a regular basis. For instance, in the NCE system, customs rules change when sanctions are imposed on countries or significant changes happen in currency exchange rates. Customs rules process different combinations of data elements as the system evolves. Modelling data interaction requirements alleviates the task of setting new bounds to the input domain.

III. Modelling of Data Interaction Requirements

In this section, we present an approach to model data interaction requirements and introduce the applications of the approach. In Section III-A, we present foundational notions of database schemas and classification tree models. In Section III-B, we describe the modelling approach, and in Section III-D we present applications of data interaction requirement models.

A. Foundations

1) Database Schema: Data requirements are typically modelled using a data model such as a *database schema*[1]. It specifies the input domain of a database in an information system. We briefly describe the well-known concept of database schema. More information on them can be found in a standard database textbooks such as [3]. A database schema typically contains one or more *tables*. A table contains *fields* with a *domain* for each field. Typical examples for field types/domains are integer, float, double, string, and date. The

[2]https://fortolling.toll.no/Tvinn-Internett

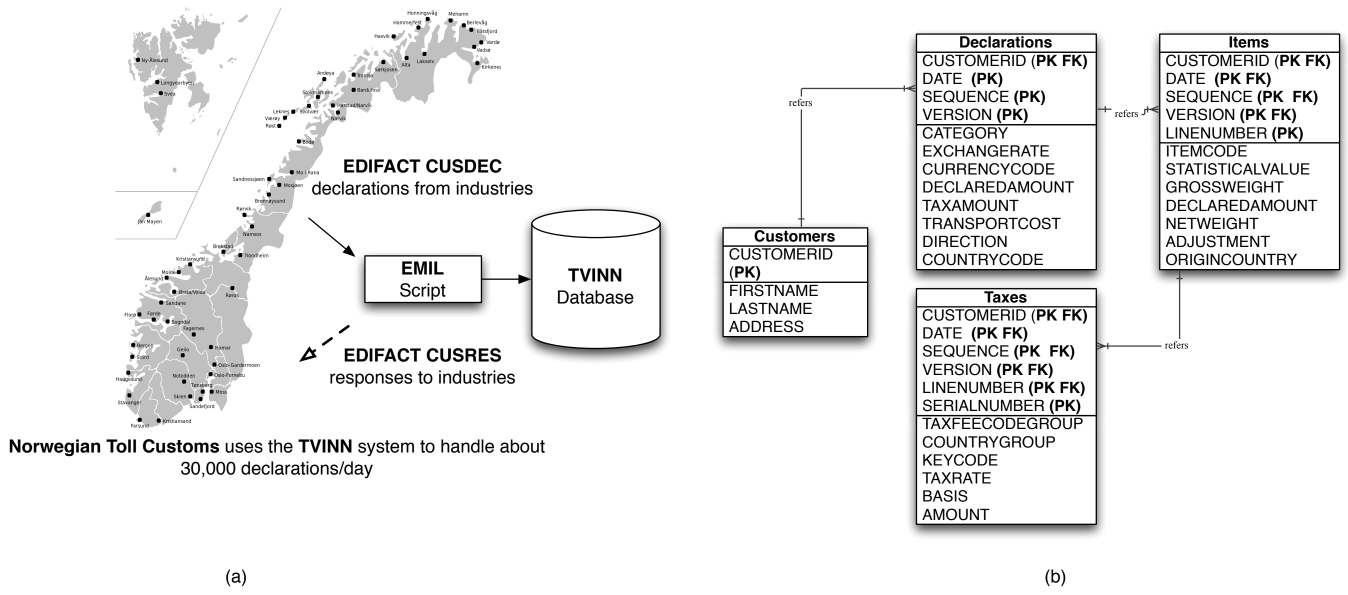

Fig. 2. (a) The TVINN System at Norwegian Customs and Excise (b) Subset of Database Schema of TVINN

value of each field must be in its domain hence maintaining *domain integrity* in a database. A table contains zero or more *records*, which is a set of values for all its fields within their domain. A table may also contain one or more fields that are referred to as *primary keys*, which identify each record. In addition, each table may refer to primary keys of other tables via *foreign keys*. The value of foreign keys must match the value of a primary key in another table. This is known as a *referential integrity* constraint. We refer to the combined concepts of *referential integrity* and *domain integrity* as *data integrity*. Records in a database must satisfy data integrity as specified by its database schema. Databases can be queried using Structured Query Language (SQL) queries. We use queries to create views and to select and count number of records. In this paper, we use the database schema, shown in Figure 2(b) for the NCE case study.

2) Classification Tree Model: We use the *classification tree models* to graphically represent data interaction requirements. The models are created using the tool CTE-XL tool[2]. CTE-XL is an editor based on the classification-tree method, as an approach to category-partition validation that uses a descriptive tree notation. This tool can scale up to the complexity of input domains such as the one of the Norwegian Tax Department [4]. CTE-XL has the notion of *compositions, classifications and classes* to model variability in a database record. The variability is typically modelled as a tree. CTE-XL also allows the specification of *dependency rules* as boolean constraints to constrain selection of classes across the tree. It provides all features necessary to model a constrained domain. In Figure 1, the top-level composition is the name of the database, the second-level compositions are table names, the third-level classifications are field names and finally in the fourth-level we have classes representing values that go into fields. Data interaction requirements can be specified manually in CTE-XL by clicking on a button for each field value in a requirement. We specify *data interaction requirements* in CTE-XL as *test cases*. CTE-XL can also be used to automatically generate interaction requirements that cover all pair-wise or three-wise interactions between classes of choice. For instance, in Figure 1, we present all pair-wise interactions between two declaration categories and four item codes.

B. Modelling Data Interactions

We assume that a modeller is knowledgeable about the database schema of the information system for which data interaction requirements are specified. The modeller specifies data interaction requirements as a classification tree model. An example is illustrated in Figure 1. The different elements of the classification tree model for data interactions are as follows:

Root Composition for Database: It is an identifier for a database on a server. Software that analyzes the classification tree model can identify a concrete database using this identifier. In Figure 1, this is represented by the composition TollCustomsDemo.

Compositions for Tables: The root composition can contain several compositions representing identifiers for tables. In Figure 1, this is represented by the compositions Declarations and Items from the schema shown in Figure 2(b). All or only a subset of tables maybe be specified depending on the use of the model.

Classifications for Fields: Fields in tables are classifications in the third level of the classification tree. For instance, in Figure 1, we use the fields Category and ItemCode. All or only a subset of fields for a table maybe be specified in the model.

Classes for Field Values: The different values for fields are classes in the model. For instance, in Figure 1, the fields values MA and FU are associated with the classification for the field Category. Field values are unique and an interaction can have exactly one possible field value.

Interactions as Test Cases in Groups: Interactions between field values across different tables are represented as test cases in a classification tree model. For instance, in Figure 1, pairwise.TestCase1 represents the interaction {MA,2208600}. This is the interaction between the an ItemCode for alcohol type Vodka and manual processing of a declaration. One may envisage the use of such an interaction to generate customs rules. Interactions in CTE-XL can either be generated automatically such that all pairs or three-wise interactions between two or three classes are covered. Requirements engineers can also manually specify them. Test cases or interactions can be divided into groups to represent data interaction requirements for different facets of the information system.

The tool CTE-XL also allows specification of additional boolean constraints between classes, compositions, and classifications to limit the number of interactions or test cases.

C. Data Interaction Requirements to Software Artefacts

The data interaction requirements expressed as test cases in a classification tree model can be transformed to different types of software artefacts. We present some of the artefacts that could be put to use.

Code Generation for Data Processing: Data interaction requirements in a CTE-XL model can serve as input to generate boilerplate code for data processing in many imperative languages including PL/SQL, Java. Generated code can be in the form of condition-action expressions. Conditions are data interaction requirements for fields cross-cutting a database schema and actions could for instance be database updates. In the Norwegian Customs case study a condition could amount to a customs law between country group, and item code, tax fee code to produce a taxation rate on goods. We may also envisage generating code for data selection and filtration in very large databases.

Test Case Generation: Interactions in a CTE-XL model can be used to generate input test cases. For instance, they can be used to generate SQL queries that verify the integrity of data extracted from a variety of fields in a database. A create view SQL query for instance can be used to extract data from multiple interacting fields. A select count query in SQL can count the frequency or absence of interactions in a database. Questions such as, does my database contain an interaction that is not valid may be answered with such queries.

Visualizing Requirements Coverage: Data in databases may require to maintain certain properties. Such properties may be specified by a data interaction requirements. We can imagine extracting data interactions and visualizing them in concise manner using graphing libraries such as Google charts. These charts will help observe which interaction requirements are satisfied and which are not. This is particularly a challenge for Big Data where millions of records need to be analysed for requirements coverage.

D. Applications

We list out some of the applications of modelling data interactions requirements with our approach:

Modelling Bounds on Data: Information system data is typically bound by a data model. The data model such as a database schema specifies domains on fields and constraints such as unique key to ensure data integrity. However, data interactions that cross-cut a data model sometimes need to be restricted or bounded. We illustrate this using the customs rules. Some interactions never occur in reality. A model of data interaction requirements can prevent fields (in different tables) from taking on values that are not valid. These models can be specified by a domain expert. For instance, a custom rule that sets tax on an item with currency code NOK and origin country China is unlikely.

Inputs for Data Processing Rules: We can use a classification tree model to automatically generate code templates for rules to process data in an information system. There is also the need to maintain traceability between the model and the code.

Surgical Knowledge for Requirements Engineers: Requirements engineers can use a data interaction model to probe an information system's database like a surgeon. Requirements for different facets of an information system can be specified in different models. The information in the model can be used to both synthesize code and perform different types of analysis on the information system.

Testing for Coverage and Robustness: All feasible T-wise interactions between field values in a database can be used to verify coverage in a large database. For instance, one may wonder if a very large database exported from TVINN covers all customs rules. The set of 220,000 three-wise interactions between item codes, country groups, and tax types can be modelled in a classification tree model and hence be used to check database coverage.

Visualizing Interactions and their Coverage: We may inverse the problem an use the classification tree model to visualize interactions between field values present in an existing database. Are all interactions in real-world transactions valid given a set of boolean constraints? This is one possible question we may address by visualizing interactions extracted from a database.

IV. Related Work

Although modelling of requirements in general and of data requirements in particular have been important research topic for the last two decades (see for instance [1]), modelling of data interaction requirements has received very little attention.

Modelling such requirements can be essential for critical information systems such as TVINN.

We present *classification tree models* as a visual language to model data interactions. However, there are other languages that could be used to model these interactions. Feature modelling [5][6] could be a successful alternative to classification tree models, and, tool support exists for automated analysis of feature models [7][8]. These tools, can also be used to generate configurations (or valid interactions) that satisfy constraints in a feature model [9] [10] [11]. However, our principal reason to use classification tree models for data interactions are: (a) the rigid semantic of classification tree models ensures exactly one class per classification, which is ideal to represent unique variations in field values in a database; (b) convenient visual interface to specify and observe data interaction requirements, and; (c) past experience of using CTE-XL for large industrial case study at the Norwegian Tax department [4]. CTE-XL could visually represent contents of 4 million customer records without crashing. In [12], the authors illustrate how annotated graphs and entity-relationship diagrams can be used to merge multiple views of database system. This mechanism can be seen as a possible approach to merge interactions across views in a database in future work.

Classification tree models have also been recently used in the industrial context [4]. In [4], the authors used classification trees to model functional requirements in a database application. They look at the inverse problem of finding database records that satisfy functional requirements specified at a high-level. These database records were, used as test cases to find regression faults in the Norwegian Taxation information system SOFIE using their regression testing tool DART.

V. Conclusion

Requirements in data-intensive software systems have three principal perspectives *data*, *function*, and *behavior*. The requirements for the data perspective of information systems specify the input/output domain, and, database schemas are typically used to model them.

In this paper, we take modelling of the data perspective a step further by modelling data interaction requirements: *interactions between field values across tables in a data model*. Our motivation emerges from importance of data interactions in many critical informations systems, especially for e-governance.

We have presented the use of the classification tree models to specify data interaction requirements for a large, complex system of the Norwegian Customs and Excise department. Models of data interaction requirements can be transformed to different kinds of software artefacts such as boilerplate code for data-processing rules, test cases, and code to generate visualizations. The models of data interaction requirements have several potential applications in developing software for information systems such as bounding input/output data that cross-cut the database structure, generating combinatorial interaction tests, and generating code for data processing rules. These are open challenges for the software engineering community in general. From a requirements engineering perspective, we believe that maintaining *traceability* between field values in information system databases and the data interaction models could be a direction for future research.

Acknowledgment

We thank the Norwegian Customs and Excise department for their trustful interactions with us. In particular we would like to thank Atle Sander, Astrid Grime and Katrine Langset for their valuable inputs. We thank the Norwegian Research Council for without their generous support it would not have been possible to setup such a close Industry-Academia collaboration for high impact software engineering research.

References

[1] K. Pohl and C. Rupp, *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant*, 1st ed. Rocky Nook, 2011.

[2] E. Lehmann and J. Wegener, "Test case design by means of the cte xl," in *Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000)*, 2000, pp. 1–10.

[3] C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston, MA: Pearson Addison-Wesley, 2004.

[4] E. Rogstad, L. Briand, R. Dalberg, M. Rynning, and E. Arisholm, "Industrial experiences with automated regression testing of a legacy database application," in *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance*, ser. ICSM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 362–371. [Online]. Available: http://dx.doi.org/10.1109/ICSM.2011.6080803

[5] D. Batory, "Feature models, grammars, and propositional formulas," in *SPLC*, 2005, pp. 7–20.

[6] K. Czarnecki, S. Helsen, and U. Eisenecker, "Formalizing Cardinality-based Feature Models and their Specialization," *Software Process Improvement and Practice*, vol. 10, no. 1, pp. 7–29, 2005.

[7] M. Antkiewicz and K. Czarnecki, "Featureplugin: feature modeling plug-in for eclipse," in *OOPSLA workshop on eclipse technology eXchange*. New York, NY, USA: ACM, 2004, pp. 67–72.

[8] S. Segura, "Automated Analysis of Feature Models using Atomic Sets," in *[13]*, 2008.

[9] G. Perrouin, S. Oster, S. Sen, J. Klein, B. Baudry, and Y. Le Traon, "Pairwise Testing for Software Product Lines: Comparison of Two Approaches," *Software Quality Journal*, vol. 20, no. 3-4, pp. 605–643, Apr. 2012. [Online]. Available: http://hal.inria.fr/hal-00805856

[10] A. Hervieu, B. Baudry, and A. Gotlieb, "Pacogen: Automatic generation of pairwise test configurations from feature models," in *Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium*, 2011, pp. 120–129.

[11] G. Perrouin, S. Sen, J. Klein, B. Baudry, and Y. Le Traon, "Automated and scalable t-wise test case generation strategies for software product lines," in *International Conference on Software Testing (ICST'10)*, Paris, France, 2010.

[12] M. Sabetzadeh and S. Easterbrook, "View merging in the presence of incompleteness and inconsistency," *Requir. Eng*, pp. 174–193, 2006.

[13] D. Benavides, A. Ruiz-Cortés, D. Batory, and P. Heymans, "First international workshop on analysis of software product lines (aspl'08)," in *SPLC*. Washington, DC, USA: IEEE Computer Society, 2008, p. 385.