Balancing Model Usability and Verifiability with SBVR and Answer Set Programming

Deepali Kholkar
deepali.kholkar@tcs.com

Dushyanthi Mulpuru
dushyanthi.mulpuru@tcs.com

Vinay Kulkarni
vinay.vkulkarni@tcs.com

TCS Research
Pune, India

Abstract

Model driven engineering (MDE) approaches necessitate verification and validation (V&V) of the models used. Balancing usability of modeling languages with verifiability of the specification presents several challenges. We present an approach and early results using the Semantics of Business Vocabulary and Business Rules (SBVR) standard as modeling notation and answer set programming (ASP) logic paradigm for verification of domain models as an attempt to address these challenges, illustrated in the problem context of regulatory compliance for business enterprises with an example from a real-life regulation.

1 Introduction

MDE seeks to enable development of large, complex systems using models as basis, necessitating V&V of the models. The vocabulary and business rules of the problem domain need to be captured as the first step in model building. It is desirable that models are populated directly by subject matter experts (SMEs) to minimize errors and provide them with greater control over the specification. Usability and verifiability are therefore two key requirements from modeling languages. In practice, we find that modeling languages that are easy to use by SMEs are less formal, therefore harder to verify e.g. Unified Modeling Language (UML), while formal modeling languages such as Alloy, Kodkod, CSP, and SAT specification languages that have automated solvers available are hard for SMEs to use. Models facilitate V&V since they can be used to generate the formal specification for validating themselves that can be solved using off-the-shelf solvers as in [GD17, ABGR07, CCR08, FSB04, FF08].

In our earlier work we built a generic MDE framework for rule checking that we applied to the complex real-world problem of regulatory compliance by enterprises [RSKK17]. Our framework illustrated in Figure 1 comprises three parts: a) automated extraction from natural language (NL) text of regulations to aid SMEs in authoring a model of regulation rules [RSKK17], b) model-based generation of formal logic specification of rules [KSK17] and extraction of relevant enterprise data, and c) automated checking of extracted data for rule compliance using an off-the-shelf rule engine. V&V of the models used was not dealt with in this work, and is the focus of the current paper.

Direct derivation of formal logic rules from NL regulation text is not feasible [LN13], therefore controlled natural language (CNL) and a rule model are selected as logical intermediate steps. Attempto Controlled English is a CNL that translates directly to formal specification while we need the intermediate model for step b). We choose the SBVR modeling standard by Object Management Group (OMG) to build the rule model since SBVR a) is an expressive notation for modeling business vocabulary and rules of any domain and b) provides a CNL interface called SBVR Structured English (SBVR SE) usable by SMEs to populate the model that is grounded in first-order logic (FOL).

Automated validation of models created using a flexible CNL notation such as SBVR SE however poses several problems. SAT solvers used in existing model validation approaches [GD17, ABGR07, FSB04, FF08] use programming paradigms that are much less expressive than SBVR and do not support aggregates, relations, or functions. Alloy, Kodkod, and Object Control Language (OCL) support encoding of aggregates, relations and constraints but not of rules with modalities such as obligations and permissions, and default behavior with exceptions, all of which frequently occur in real-world problem contexts such as regulation,

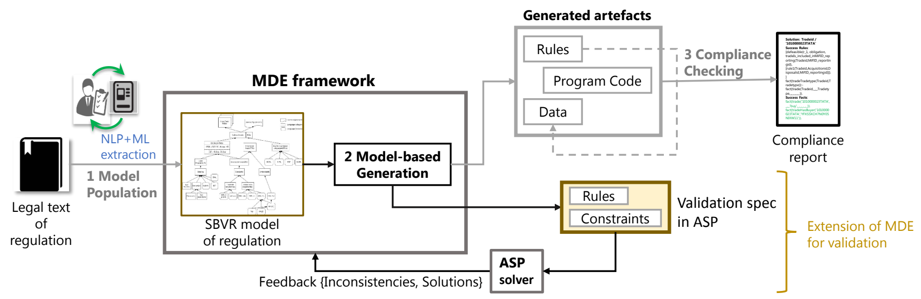

Figure 1: Extension of our MDE Framework for Model Validation

policy or contract compliance. SBVR supports these features, however validating an SBVR model using any of these languages requires implementation in terms of the language primitives, a non-trivial task. To overcome these limitations, we propose a V&V approach for domain models built in SBVR using the ASP logic programming paradigm detailed below.

2 Our V&V Approach for Domain Models

We extend our earlier developed MDE framework to generate the formal logic specification for V&V of the SBVR model in the ASP paradigm, as shown in Figure 1. Model verification seeks to check that the model does what it is expected to do, usually through an implementing program[1]. We propose inconsistency checking, scenario generation, and testing using a standard set of test cases and test data as verification techniques in our approach.

The objective of model validation is to ascertain that the model is a faithful representation of the problem space it represents[1]. We implement validation by having SMEs certify scenarios generated from the model for validity and coverage. We briefly touch upon the SBVR and ASP technologies before explaining the approach using an illustration from the KYC regulation for Indian banks. KYC is an anti-money laundering regulation that lays down guidelines for acceptance of new customers including risk categorization and identification procedures.

2.1 SBVR

SBVR is a fact-oriented modeling notation [Nij07] that encodes *rules* as *logical formulations* over *fact types* that are relations between *concepts*[2]. Rules are stated in SBVR SE to populate the model. Listing 1 shows a few SBVR SE rules from the KYC example. Rules r6 and r7 define obligations on the bank to open an account for a customer. Rule r6 is populated in the SBVR model as a *rule*, *meant by* an *obligationformulation* embedding an *implication logical formulation* over fact types *bank opensAccountFor customer* and *bank verifiesIdentityOf customer* that relate concepts *bank* and *customer*.

Listing 1: Section of SBVR SE Rules from the KYC Example

```
rule r1 It is necessary that customer @is
    highRiskCust || customer @is lowRiskCust
rule r2 It is necessary that customer @has
    sourceOfWealthEasilyIdentified if customer
     @is lowRiskCust
rule r4 It is necessary that customer @is
    salaried || customer @is govtDept if
    customer @has
    sourceOfWealthEasilyIdentified
rule r6 It is obligatory that bank
    @opensAccountFor customer if bank
    @verifiesIdentityOf customer
rule r7 It is obligatory that bank not
    @opensAccountFor customer if customer
    @approaches bank && customer @is
    bannedCustomer
rule r8 It is necessary that bank
    @verifiesIdentityOf customer if customer
    @approaches bank && customer @submits
    document
```

2.2 ASP

We select ASP for automated verification since it a) is a powerful, highly expressive logic programming notation supporting aggregates, functions, and optimizations and b) maps directly onto SBVR's fact-oriented model and underlying first-order logic formalism.

Rules in SBVR are translated as rules in ASP of the form *Head :- Body*, where *Body* is the antecedent that implies the consequent *Head*. Statement-type rules without antecedents are translated as constraints i.e. rules with empty head, as *:- Body*. Relations in SBVR become predicates in ASP, with related concepts as arguments referred by their ids e.g. *bank @opensAccountFor customer* becomes *opensAccountFor(BankId,CustomerId)*. Our model-based generator uses Eclipse Modeling Framework (EMF) generated model traversal functions to traverse the SBVR model and translate it to ASP rules and constraints. Listing

[1]http://www.inf.ed.ac.uk/teaching/courses/pm/Note16.pdf
[2]https://www.omg.org/spec/SBVR/1.3

2 shows a fragment from the ASP program generated for the KYC example.

Listing 2: ASP Fragment for KYC Example
```
highRiskCust(CustomerId) v lowRiskCust(
    CustomerId) :- customer(CustomerId).
sourceOfWealthEasilyIdentified(CustomerId) :-
    lowRiskCust(CustomerId).
salaried(CustomerId) v govtDept(CustomerId) :-
    sourceOfWealthEasilyIdentified(CustomerId
    ).
opensAccountFor(BankId, CustomerId) :-
    verifiesIdentityOf(BankId, CustomerId).
-opensAccountFor(BankId, CustomerId) :-
    approaches(CustomerId, BankId),
    bannedCustomer(CustomerId).
verifiesIdentityOf(BankId, CustomerId) :-
    approaches(CustomerId, BankId),submits(
    CustomerId, DocumentId).
submits(CustomerId, DocumentId) :-salaried(
    CustomerId),letterOfIdentityFromCorporate(
    DocumentId).
```

ASP supports encoding of extended logic programs with choice, cardinality, classical negation, and disjunction in the head of rules, not supported by normal logic programs, enabling verification of these features that are supported in SBVR SE. Disjunction in rule heads is a powerful feature used to model mutually exclusive paths, such as high and low risk categorization in rule r1 and and two types of low risk customers in rule r4 in Listing1. Classical negation allows explicit statement or derivation of negative knowledge, e.g. rule r7 in Listing1.

ASP solvers perform automated analysis by *grounding* the program i.e. generating variable-free rules given a set of ground facts and using techniques similar to SAT solving to generate solutions or answer sets of the program.

2.3 V&V using ASP

We use the DLV system [LPF$^+$06] as the solver for our generated ASP programs. The solver performs consistency checking of the rule base, indicating conflicting rules or constraints and generating answer sets for paths where no conflicts exist. Answer sets by definition are minimal, i.e. no answer set can be contained in another [Lif08], and represent the minimal set of unique scenarios for the input model. These are presented to SMEs to check whether the generated scenarios are valid and whether all important scenarios have been covered.

The KYC program of Listing2 generates four answer sets corresponding to two risk categories and four customer types. Listing3 shows ground facts provided for a single customer with id 301 and a sample answer set for salaried customer.

Listing 3: Input Ground Facts and Sample Answer Set
```
% Ground Facts:
customer(301).
approaches(301,201).
letterOfIdentityFromCorporate(401).
%bannedCustomer(301).
% Solutions:
{customer(301), approaches(301,201),
    letterOfIdentityFromCorporate(401),
    lowRiskCust(301),
    sourceOfWealthEasilyIdentified(301),
    salaried(301), submits(301,401),
    verifiesIdentityOf(201,301),
    opensAccountFor(201,301)}
```

Although minimal, number of answer sets can grow combinatorially with conditions and ground instances. To constrain number of scenarios, only critically important conditions are modeled as disjunctions that create independent scenarios for each condition, while non-critical conditions such as id proof submitted by a customer e.g. letterOfIdentityFromCorporate are modeled as normal clauses and minimal number of ground facts are provided to prevent combinatorial explosion.

Another verification step is testing the ASP program using a set of manually created positive and negative test cases with test data provided as ground facts and checking generated scenarios against expected results. When test data in the form of ground fact bannedCustomer(301) is added, rule r7 becomes true, conflicting with rule r6 and displays the inconsistency as shown in Listing 4.

Listing 4: Inconsistencies Shown by ASP Solver
```
% Facts derived as true in every answer set:
-opensAccountFor(201,301).
% Inconsistencies:
:- opensAccountFor(201,301).
```

Inconsistencies need to be corrected in the model and solutions re-generated iteratively until no anomalies remain.

2.4 Discussion

In our experience of using our framework over two years on sections from three real-life regulations, we trained several groups of SMEs who found SBVR SE to be an easily usable notation to understand and specify rules without requiring knowledge of underlying technologies. Using a general-purpose CNL and model saves us the effort of implementing a DSL, also keeps the approach generic and usable for any problem domain. CNL being more flexible, verification is harder than in case of a DSL. Our SBVR SE language implementation therefore places some restrictions on sentence structure and keywords for ease of verification, taking care not to compromise on feature support.

It is desirable that V&V ensures that specified rules form a correct inferencing hierarchy. Work to supplement the approach described here with static analysis of the model to flag rules that are not explicated further as possible cases of missing rules, check for cycles and reachability is ongoing. However the same clause may get specified in multiple ways. The extra work to be performed for verification is compensated by other advantages of the SBVR-ASP combination mentioned earlier, also minimal translation effort from SBVR to ASP. Not having to implement an input DSL that reduces the development and testing effort on language engineering and model transformations. SBVR helps identify and map to actual data if available, else generate mock data in our framework [RSKK17].

Having so far worked with partial regulations, we plan to encode a few complete regulations that specify rules at different levels of abstraction in order to test adequacy of the SBVR and ASP notations.

3 Conclusion

Usability and verifiability of modeling languages are both critical in any MDE approach, however often conflict in practice. We proposed using CNL as a stepping stone to populate models and reviewed the challenges faced in verification of such a model using commonly used model checking languages. We described our V&V approach using SBVR and ASP to address these challenges and illustrated it with an example from the KYC regulation for Indian banks. Ongoing investigations include managing conflicts between rules using priorities and automated generation of positive and negative ground data for verification.

4 References

References

[ABGR07] Kyriakos Anastasakis, Behzad Bordbar, Geri Georg, and Indrakshi Ray. Uml2alloy: A challenging model transformation. In *Model Driven Engineering Languages and Systems, 10th International Conference, MoDELS 2007, Nashville, USA, September 30 - October 5, 2007, Proceedings*, 2007.

[CCR08] Jordi Cabot, Robert Clarisó, and Daniel Riera. Verification of UML/OCL class diagrams using constraint programming. In *First International Conference on Software Testing Verification and Validation, ICST 2008, Lillehammer, Norway, April 9-11, 2008, Workshops Proceedings*, 2008.

[FF08] S. Feja and D. Ftsch. Model checking with graphical validation rules. In *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008)*, 2008.

[FSB04] F. Fleurey, J. Steel, and B. Baudry. Validation in model-driven engineering: testing model transformations. In *Proceedings. 2004 First International Workshop on Model, Design and Validation.*, 2004.

[GD17] Martin Gogolla and Khanh-Hoang Doan. Quality improvement of conceptual UML and OCL schemata through model validation and verification. In *Conceptual Modeling Perspectives.*, pages 155–168, 2017.

[KSK17] Deepali Kholkar, Sagar Sunkle, and Vinay Kulkarni. Towards automated generation of regulation rule bases using MDA. In *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2017, Porto, Portugal, February 19-21, 2017.*, 2017.

[Lif08] Vladimir Lifschitz. What is answer set programming? In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence,AAAI 2008, Chicago, Illinois, USA, July 13-17*, 2008.

[LN13] François Lévy and Adeline Nazarenko. Formalization of natural language regulations through SBVR structured english - (tutorial). In *Theory, Practice, and Applications of Rules on the Web - 7th International Symposium, RuleML 2013, Seattle, WA, USA, July 11-13, 2013. Proceedings*, pages 19–33, 2013.

[LPF$^+$06] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[Nij07] Sjir Nijssen. SBVR: Semantics for business. 2007.

[RSKK17] Suman Roychoudhury, Sagar Sunkle, Deepali Kholkar, and Vinay Kulkarni. From natural language to SBVR model authoring using structured english for compliance checking. In *21st IEEE International Enterprise Distributed Object Computing Conference, EDOC 2017, Quebec City, QC, Canada, October 10-13*, 2017.