Change Propagation of View Models by Logic Synthesis using SAT solvers

Oszkár Semeráth[1,2] Csaba Debreceni[1,2] Ákos Horváth[1] Dániel Varró[1,2]

[1] Budapest University of Technology and Economics,
Department of Measurement and Information Systems,
1117 Budapest, Magyar tudósok krt. 2.
{semerath,debreceni,ahorvath,varro}@mit.bme.hu

[2] MTA-BME Lendület Cyber-Physical Systems Research Group

Abstract

View models are key concepts of domain-specific modeling to provide task-specific focus (e.g., power or communication architecture of a system) to the designers by highlighting only the relevant aspects of the system under design. View models can be specified by unidirectional graph queries, and automatically maintained from the underlying source model using incremental transformation techniques. However, tracing back the consequence of a modification in the abstract view to the source model is a challenging task as several valid source changes may correspond to a single change in the view. Calculating these source changes requires complex logic analysis which has to take into account both queries defining the view and the structural well-formedness constraints of the source model. In this paper we outline a systematic technique to calculate valid source candidates by iterative calls to underlying SAT solvers.

1 Introduction

View models are key concepts of domain-specific modeling to provide task-specific focus (e.g., power or communication architecture of a system) to the designers by creating a model which highlights only the relevant aspects of the system under design and aids the detection of conceptual flaws. In [DHH+14], we proposed an approach to define view models in a highly automated way, based on unidirectional declarative graph queries [UBH+15] and using incremental transformation techniques [VB07].

However, view models created in this way are read-only representations, thus changes in the view model can not be directly propagated to a change in the source model. When a conceptual change is need to be done in a view model, the developer is forced to edit the source model, and manually check if the modified model corresponds to the expected view model. Additionally, there might be several other views which may be unintentionally changed, or structural well-formedness constraint may be violated by the manual modification on the source.

In: A. Anjorin, J. Gibbons (eds.): Proceedings of the Fifth International Workshop on Bidirectional Transformations (Bx 2016), Eindhoven, The Netherlands, April 8, 2016, published at http://ceur-ws.org

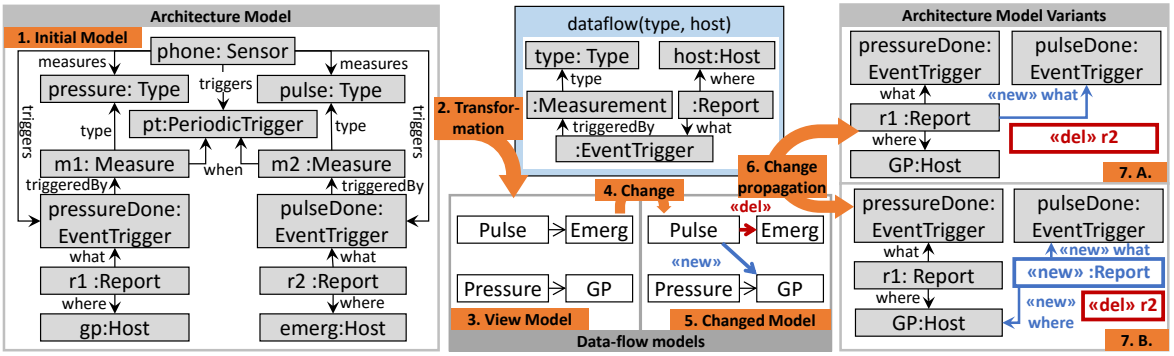

Figure 1: Telecare model

To overcome this problem we propose a technique to automatically deduce source model variant which are consistent to a changed view model. First (i) the unaffected partition of the source model is selected to restrict the impact of a modification, (ii) transforms the modified view model to a set of logic conditions according to the specification of the view defining graph queries and (iii) calculates possible candidate solutions by dedicated SAT solvers by combining the unaffected part, the conditions and the well-formedness constraints. Hence, this technique helps the developement process by allowing the enumeration on valid design decisions for a change in the view model. An initial implementation is carried out using Alloy Analyser [all] as the back-end solver to construct several solution candidates.

2 Change Propagation in Health Care Architecture Models

Our change propagation approach is illustrated in the context of the healthcare case study developed for the Concerto Artemis project [con]. As source model, a *health care architecture* metamodel is developed which consists of Sensor instances that measure specific Type of data. The execution of a measurement can be initiated by a PeriodicTrigger while an EventTrigger captures the end of a measure and can trigger Report operation to send the measured data to a specific Host. A domain-specific language may define several *well-formedness constraints (WF)* to ensure the correctness of the model under developement, which tipically defined in OCL invariants [OCL06], or as graph patterns [BHR+10]. For example, in the architecture level a WF specifies that all the measurements have to be reported. As view model, a *data-flow model* is created which connects measurement Types to the target Host.

Example. Our approach is illustrated on a pulse and blood pressure measurement environment controlled by a smart phone, which is depicted on the left side of Fig. 1 labelled as Initial Model in *Step 1*. The measurement is executed by the Sensors of a mobile phone. The phone measures two types of data: pulse and blood pressure. The measurements m1 and m2 are executed periodically, triggered by the the phone timer pt. The completion event of the measurements triggers the job for processing of the sensor data: pressureDone and pulseDone. The result of the measurement is collected to Reports r1 and r2, and sent to the different hosts. In our case study, the blood pressure is sent to the general practitioner (GP) of the patient for logging, and signs of hearth failure is sent to hospitals (modelled by emerg). In order to create a focused view on what information is sent to the hosts, a view model of the data-flow is created in step *Step 2*. Graph pattern dataflow(type, host) is presented in the middle-top of Fig. 1, which matches to a type-host pair if the corresponding measurement is triggered and reported to the host. Therefore the data-flow model labelled as View Model is created in *Step 3*: the pulse is sent to the emergency department, and the blood pressure is forwarded to the General Practitioner (GP).

Let us assume that the developer makes changes in the data-flow model depicted in *Step 4*: the data-flow from Pulse to Emergency is redirected to the General Practicioner (noted by «del» and «new»), which results in Changed Model in *Step 5*. Two possible resolution of the source (architecture) model is proposed. In *Step 7.A* the pulse and blood pressure is reported in the same message. Technically, the r2 report has been removed, therefore the communication with Emergency is terminated, and a relation is added to r1. In *Step 7.B* the results of the measurements are sent in two separate reports. Report r2 is also removed in this case, and a new report is added to the model that sends the measurement results to the new address. In both cases, the well-formedness constraints are satisfied.

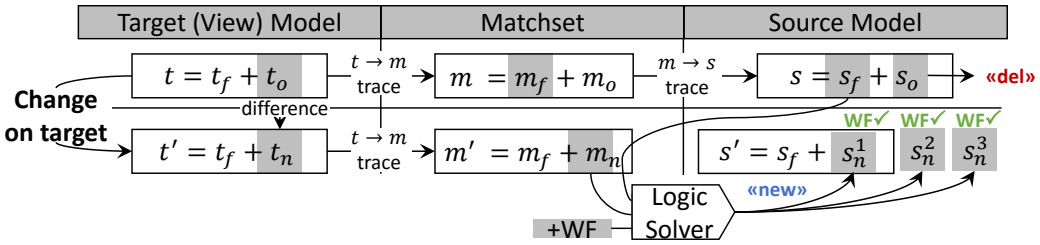

Figure 2: figure
Overview of the change propagation

Challenges. The main challenge of the case study is to propagate back the changes of the data-flow model (view model) introduced in *Step 4* to the system architecture (source model) in order to provide possible design candidates in *Step 6* that satisfies the following assumptions:

- A1 **Consistency:** Each source model candidates needs to have the same view model which is isomorphic to the modified view model.
- A2 **Well-formedness:** Generated models have to satisfy the well-formedness constraints of the domain. It also helps the developer detect infeasible requirement of a view model change.
- A3 **Localization:** A change can affect only a limited, well-defined part of the source model. For example, a sensor should not be changed upon rearranging the dataflow.
- A4 **Minimal Changes:** Solutions with minimal changes should be created. For example, in *Step 7.B*, report r2 is removed and a new report with the same role is created. From a practical perspective, it would be more reasonable to use existing model elements if possible.

3 Change Propagation by Logic Solvers

Overview. An overview of our approach is depicted in Fig. 2, which shows a *target (view) model* (t) that is derived from a *source model* (s). The view model is created based on the match set (m) of the view definition queries. The developer makes changes on t which leads to t'. The goal of our approach is to calculate a changed source model s' and maintain m'.

We assume that *traceability links* are built and maintained during the forward propagation of changes: $[t \to m]$ trace specifies which view model element is created from which match (as described in [DHH+14]), and $[m \to s]$ collects the source model elements whose change may invalidate the match (including the binding of symbolic parameters, and possible interpretation of internal variables). Upon a change from t to t', the model can be separated into two partitions: fixed model partition t_f is the part of t which remains unchanged, while old part t_o is changed to a new part t_n (cross-references are included in t_o and t_n). Consequently, a fixed subset of matches m_f, and the changed subset m_o can be calculated via the $[t \to m]$ trace. The source model is also partitioned to s_f and s_o using the $[m \to s]$ trace where s_f contains the objects whose are not associated to any objects or edges of the changing part of the view model t_n. In this paper, we associate an object to a pattern match if the object is bound to a parameter or inner variable of the pattern. This information is conceptually stored in the *traceability links*. In the future, more sophisticated calculation can applied to select the relevant part more precisely. t_n directly specifies the invalidated and newly introduced matches m_n. However, the changed subset of matches m_n specifies declarative structural constraints only, which have to be satisfied by the modified part.

Therefore the possible solutions s_n for a change in the target model are calculated based on (1) the unchanged part of the source model s_f, (2) the requirements defined by the changed matches m_n, and (3) additional domain-specific well-formedness constraints *WF*. All these constraints are transformed into a first-order logic problem to be solved by a *logic (SAT/SMT) solver*. The solver provides several possible valid solutions for s_n, which from the model developer may choose the most suitable one s_n^i. Finally, the obsolete part s_o is replaced with s_n^i.

In order to represent model generation tasks as logic problems, the structural constraints of the source meta-model are mapped to a set of formulae *Meta*. Additionally, the well-formedness constraints *WF* can be added to the logic problem to create valid solutions. Finally, [SBH+15] describes how graph patterns *Patterns* and requirements about the matches defined by the view model *Matches* can be translated to logic relations and assertions. Therefore, an s where $s \models Meta \wedge WF \wedge Patterns \wedge Matches$ is a valid source model, where view defining patterns has matches as the constraint *Matches* specifies, so *A1 Consistency* and *A2 Well-formedness* challenges are satisfied, similarly as [MC13] describes. However, generating full models from scratch causes scal-

ability issues (which is a common problem with logic solvers). Additionally, *A3 Localization* and *A4 Minimal Changes* assumptions are ignored, or have to be enforced by additional constraints. However, in our approach the generated source model is separated into a fix s_f and changing s_n part, where only the generation of the s_n part is issued to the solver, similarly as [SVV16] describes incremental model generation. Hence, the well-formedness and the view definitions in s_f can be excluded from the problem to simplify the problem and ensure *A3* and *A4* requirements, while creating problems proportional only to the size of the changing part.

Evaluation. By applying the method described in the previous section, valid health care system variations were synthesized. In the following we evaluate our solution with respect to the outlined challenges:

Challenge 1: Because the query specification is explicitly mapped to constraints on the model fragment to be generated, the correspondence is ensured.

Challenge 2: The well-formedness constraints can be directly added to the logic problem, therefore only valid models will be generated.

Challenge 3: Our solution selects the largest part in the source model where the change may be synthesised. While approach is useful for searching solutions which are hard to satisfy, or identify unsatisfiable changes, simple changes may have unnecessary side-effects in irrelevant model partitions.

Challenge 4: In the current state, our approach replaces the old partition with newly created objects, and does not takes the previous state of the modified part into account.

While challenge 1. and 2. is solved, our the current solution may underperform in aspects of 3. and 4. In the future, we plan to generate solutions in multiple steps, where the scope of the change is limited at first, and in each step extended with additional elements if the solver fails to create simple solutions. For example, in the first step the solver should try to set only attributes, then it can redirect edges, and finally create or delete objects. We expect that these steps can be used to prioritise better solutions.

4 Related Work

Most of the view model synchronization techniques use bijective transformations where the rules are pairs of functions reverse of each other such in lenses [FGM+07], injective functions [MHT04], ATL [XLH+07] or relational/declarative descriptions such as in TGG [HEO+13] or QVT [Ste10]. These transformations can provide incremental backward propagation of changes. However, well-formedness constraints have to be included in all transformation pairs separately and cannot guarantee the completeness and unambiguity of the solutions.

Other approaches, such as [CDREP10], allow non-injective transformation for target model derivation such as that use logical representation to calculate the reverse direction. [MC13] reuses the declarative rule definition of *QVT-R* and maps it to *Alloy* SAT solver. [CK13] formalise the problem to Linear Programming. However, these techniques aim to find valid source models from scratch, instead of valid model parts. Moreover, well-formedness constraints are not taken into account. [Het10] defines the *partial synchronization*, where the changing part of the model is defined by a minimal explanation of an *abductive reasoning* problem. In contrast to our work, [Het10] defines an operation-based approach to search for minimal set of operations on the source model.

We believe that our contribution is unique in the context of state-based model synchronization in the sense that uni-directional and non-injective derivation rules are reversed using a SAT-solver to find well-formed models solutions, where the rules are mapped to first-order logic constraints and only relevant model parts are extended instead of the recalculation of the whole source model.

5 Conclusion and Future Work

In this paper, we presented an approach for backward propagation of changes from view to source models by (i) selecting the impact of a modification, (ii) transforming the modified view model to logic conditions and (iii) calculating possible solution candidates by mapping the unaffected part, the conditions and well-formedness constraint to a SAT problem. Initial prototype is implemented which successfully generated design candidates for the case study using Alloy Analyzer as the back-end solver. As future work, we plan to (i) priorize the synthesized solutions and (ii) evaluate the performance of our approach on different measurement scenarios.

Acknowledgement

This paper is partially supported by the CONCERTO (ART-2012-333053) and the MTA-BME Lendület 2015 Research Group on Cyber-Physical Systems.

References

[all] Alloy Analyzer. `http://alloy.mit.edu/`.

[BHR+10] Gábor Bergmann, Ákos Horváth, István Ráth, Dániel Varró, András Balogh, Zoltan Balogh, and András Ökrös. Incremental evaluation of model queries over EMF models. In *Model Driven Engineering Languages and Systems, MODELS 2010*, pages 76–90, 2010.

[CDREP10] Antonio Cicchetti, Davide Di Ruscio, Romina Eramo, and Alfonso Pierantonio. Jtl: a bidirectional and change propagating transformation language. In *Software Language Engineering*, pages 183–202. Springer, 2010.

[CK13] Glenn Callow and Roy S Kalawsky. A satisficing bi-directional model transformation engine using mixed integer linear programming. 2013.

[con] CONCERTO ARTEMIS project. `http://concerto-project.org/`.

[DHH+14] Csaba Debreceni, Ákos Horváth, Ábel Hegedüs, Zoltán Ujhelyi, István Ráth, and Dániel Varró. Query-driven incremental synchronization of view models. In *Proceedings of the 2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, page 31. ACM, 2014.

[FGM+07] J Nathan Foster, Michael B Greenwald, Jonathan T Moore, Benjamin C Pierce, and Alan Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(3):17, 2007.

[HEO+13] Frank Hermann, Hartmut Ehrig, Fernando Orejas, Krzysztof Czarnecki, Zinovy Diskin, Yingfei Xiong, Susann Gottmann, and Thomas Engel. Model synchronization based on triple graph grammars: correctness, completeness and invertibility. *Software & Systems Modeling*, 14(1):241–269, 2013.

[Het10] Thomas Hettel. *Model round-trip engineering.* PhD thesis, Queensland University of Technology, 2010.

[MC13] Nuno Macedo and Alcino Cunha. Implementing qvt-r bidirectional model transformations using alloy. In *Fundamental Approaches to Software Engineering*, pages 297–311. Springer, 2013.

[MHT04] Shin-Cheng Mu, Zhenjiang Hu, and Masato Takeichi. An injective language for reversible computation. In *Mathematics of Program Construction*, pages 289–313. Springer, 2004.

[OCL06] *Object Constraint Language, v2.0*, May 2006.

[SBH+15] Oszkár Semeráth, Ágnes Barta, Ákos Horváth, Zoltán Szatmári, and Dániel Varró. Formal validation of domain-specific languages with derived features and well-formedness constraints. *Software & Systems Modeling*, 2015.

[Ste10] Perdita Stevens. Bidirectional model transformations in QVT: semantic issues and open questions. *Software & Systems Modeling*, 9(1):7–20, 2010.

[SVV16] Oszkár Semeráth, András Vörös, and Dániel Varró. Iterative and incremental model generation by logic solvers. *Fundamental Approaches to Software Engineering, 19th International Conference, FASE 2016*, 2016.

[UBH+15] Zoltán Ujhelyi, Gábor Bergmann, Ábel Hegedüs, Ákos Horváth, Benedek Izsó, István Ráth, Zoltán Szatmári, and Dániel Varró. EMF-IncQuery: An integrated development environment for live model queries. *Sci. Comput. Program.*, 98:80–99, 2015.

[VB07] Dániel Varró and András Balogh. The Model Transformation Language of the VIATRA2 Framework. *Science of Computer Programming*, 68(3):214–234, October 2007.

[XLH+07] Yingfei Xiong, Dongxi Liu, Zhenjiang Hu, Haiyan Zhao, Masato Takeichi, and Hong Mei. Towards automatic model synchronization from model transformations. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 164–173. ACM, 2007.