TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Professur für Softwaretechnologie

Master Thesis

User-Driven Constraint Modelling for Entity Models at Runtime

Anton Skripin
Born on: 10.03.1998 in Novouralsk, Russia

04.11.2023

Supervisor
Dr.-Ing. Sebastian Götz
Supervising professor
Prof. Dr. rer. nat. habil. Uwe Aßmann

Statement of authorship

I hereby certify that I have authored this document entitled *User-Driven Constraint Modelling for Entity Models at Runtime* independently and without undue assistance from third parties. No other than the resources and references indicated in this document have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present document. I am aware that violations of this declaration may lead to subsequent withdrawal of the academic degree.

Dresden, 04.11.2023

Anton Skripin

Abstract

Here should go my abstract

Contents

1 Introduction

This chapter concisely introduces the background of this master's thesis. Section 1.1 reflects the key motivation by providing a chronological overview of a modeling evolution, starting with the basic notion of a model and finishing with the modern challenges in models@run.time. Section 1.2 familiarizes a reader with the main contributions and research questions, the answers to which should be found in the scope of this work.

1.1 Motivation

The formalized notion of a model emerged from the model theory. According to the model theory and omitting all complexity, a world consists of objects. In turn, objects are endowed with properties. A model represents an original whose properties are described as mapping to the image in a model. [CK90] It is worth noting that not every model is an abstraction of a real object but can be an abstraction of another model. An intuitive example of such a sequenced model is a map application, the domain model of which cannot be a direct model of our planet. Furthermore, as stated by [Hel+16], depending on the purpose of a model it serves, it can be either descriptive or prescriptive. The former means abstracting a real object. Hence, an origin stems from an actual entity. The letter ones comprise the specification of a real entity to be constructed. Deviation from a prescriptive model specification indicates an error. Thus, in such models, an origin originates from the specification of the created entity.

Modern software systems are non-trivial and consist of numerous artifacts. Thus, requirements are collected from involved stakeholders and reflected in design and system architecture. The source code then manifests all the stages before. Finally, the documentation for a system is created either manually or automatically to maintain the origin of software knowledge among involved parties. One vital concept to grasp, however, is that nowadays, a final software product is much more than just a program code. All system artifacts are necessary to produce and support a system during its software lifecycle. If the final software lacks at least one of the elements mentioned earlier, it cannot be regarded as software but just as some purpose-specific script.

Nevertheless, what overarching role do models play during the development of software? First, regardless of the design paradigm for the development of software systems, a model is a link between a client and a developer that serves as an intermediate component every involved party can understand. Secondly, models cooperatively with documentation help keep a system's essence during its evolution.

One of the most natural ways to present and manage something complex is to depict it through modeling. Models can have different purposes. They can be applied to depict a desired structure or behavior of a system before development. Besides, models are perfect for deriving system attributes during or after development to grasp its functioning better. Thus, an object-oriented data model [Day90] must bridge a semantic gap between the real world and relational tables. On the other hand, relational models [Cod07] are highly used in database management to help experts characterize and handle data stored in a database. Being one of the most common and adopted modeling languages, UML (Unified Modeling Language) [Rum05] finds itself at the heart of Model Driven Architecture (MDA). [Sol+00]

Driven by models, MDA aims to facilitate the design and development of modern systems via weaving, traceability, transformation, and automation techniques [Sol+00]. The models produced within MDA showcase the development cycle of a system employing structural or behavioral models. As mentioned by [FR07], these models form a layer of abstraction above the code level. However, in recent years the demand is rising to casually connect a model with its running system. A potential area of application is safety-critical systems. They must guarantee high availability and responsibility even in the presence of errors. Coupling a model with a system state would allow the system to adapt its behavior without downtime. Runtime models also highly impact developers and end-users. Consequently, software engineers would benefit from this concept by being able to fix design flaws or introduce new components at runtime. Furthermore, models@run.time would give end-users the advantage of observing a system state dynamically. [BBF09]

Despite a rich scope of application and increasing potential for use in the future, models are often not explicit enough to provide full expressiveness and manifest all functional and non-functional requirements. One of the ways to circumvent such a limitation is to use a constraint modeling language to enrich a model with functional and non-functional requirements. Up until nowadays, the research community has been looking for various means to provide full expressiveness while modeling. However, the current state of research does not offer a bridge between the power of constraint languages and dynamic and variant-aware models at runtime. Does it even make sense to enrich runtime models with constraining capabilities? If the answer is yes, how must a software system look at the end where constraints enrich the concept of models@run.time?

1.2 Problem Space

1.3 Objective

This thesis attempts to fill the current research gap in the area of runtime models. Therefore, this thesis aims to research and evaluate the viability and possibility of using constraints for models at runtime by end-users. Subsequently, this work summarizes state-of-the-art regarding various available languages and modeling techniques that should be a prerequisite for overcoming current limitations in this field. Finally, this thesis introduces a framework that allows end-user to impose constraints on a runtime variant-aware model.

Furthermore, the main contributions of this master's thesis to the research community are listed below:

1. Introduced the concept of end-user-driven constraint modeling for models at runtime as a specification for restricting variant-aware models by users with no preliminary background in software engineering.

2. Provided a solution to defining such constraints by end-users and keeping the constraints valid in the presence of concurrently existing entities with different variants.

3. Implemented a framework primarily focusing on the novel concept of end-user-driven constraints at runtime. While many frameworks allow users to define constraints in a quite intuitive manner, none of them, to the best of our knowledge, tackles the challenges

of shifting the definition and validation of constraints from the design to the runtime state of a system.

Finally, the answers to the following primary research questions are provided as a result of this work:

1.
2.
3.

List of Figures

Bibliography

[BBF09] Gordon Blair, Nelly Bencomo, and Robert B France. “Models@ run. time”. In: *Computer* 42.10 (2009), pp. 22–27.

[CK90] Chen Chung Chang and H Jerome Keisler. *Model theory*. Elsevier, 1990.

[Cod07] Edgar F Codd. “Relational database: A practical foundation for productivity”. In: *ACM Turing award lectures*. 2007, p. 1981.

[Day90] Umeshwar Dayal. “Queries and views in an object-oriented data model”. In: *Proceedings of the Second International Workshop on Database Programming Languages*. 1990, pp. 80–102.

[FR07] Robert France and Bernhard Rumpe. “Model-driven development of complex software: A research roadmap”. In: *Future of Software Engineering (FOSE’07)*. IEEE. 2007, pp. 37–54.

[Hel+16] Rogardt Heldal et al. “Descriptive vs prescriptive models in industry”. In: *Proceedings of the acm/ieee 19th international conference on model driven engineering languages and systems*. 2016, pp. 216–226.

[Rum05] James Rumbaugh. *The unified modeling language reference manual*. Pearson Education India, 2005.

[Sol+00] Richard Soley et al. “Model driven architecture”. In: *OMG white paper* 308.308 (2000), p. 5.