




The Influence of Granularity of Transactions on Performance in VITRUVIUS

Thomas Weber 
thomas.weber@kit.edu
Karlsruhe Institute of
Technology
Karlsruhe, Germany

Benedikt Jutz 
benedikt.jutz@kit.edu
Karlsruhe Institute of
Technology
Karlsruhe, Germany

Zenon Zacouris 
zenon.zacouris@tum.de
Technische Universität
München
München, Germany

Abstract

The development of cyber-physical systems involves managing interconnected models, where changes in one model may require updates in others to maintain consistency. Our research evaluates the performance of a tool for multi-model consistency management, VITRUVIUS, focusing on how transaction size and meta-model topology influence runtime in managing these models. Our results indicate that the number of model deltas within a transaction can lead to a degradation in system performance by a factor of over 300 in the worst-case scenario. Additionally, consistency preservation rules significantly slow down performance, with scenarios involving a higher fan-out degree (number of directly connected models) leading to greater performance reductions than those involving a chain of metamodels.

Index terms— Transaction, VITRUVIUS, Granularity, Performance

1 Introduction

The development of cyber-physical systems (CPS), such as electric cars, requires collaboration among multidisciplinary teams to create functional products. Hence, a change in one model typically requires adjustments in related models. For example, changing the tire size may require adjustments to the chassis to maintain compatibility. Failure to update the chassis can lead to inconsistencies, resulting in a non-buildable vehicle [5].

To address these challenges of multi-model consistency management, VITRUVIUS [3] was developed. VITRUVIUS implements a (virtual) Single Underlying Model (V-SUM) that combines the information for the development of CPS. This V-SUM encapsulates all system aspects through interconnected models, offering three key advantages of the models over a monolithic approach: 1) they are easier to build, 2) they are easier to evolve, and 3) they often already exist. Each user interacts with a specific view, a subset of the entire system. Interconnected models are linked by Consistency Preservation Rules (CPRs), which automatically update related models in response to changes. For

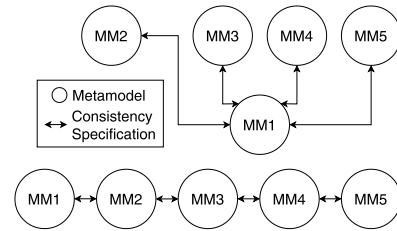


Figure 1: The fan-out and the chain topology

instance, if a user alters tire size, CPRs will adjust the chassis size and other related components. We define models using the Eclipse Modeling Framework (EMF) [1], and the CPRs are specified using the Reactions [3] language.

To apply changes to a V-SUM, VITRUVIUS uses deltas. A delta is a modification to a model, such as adding, updating or removing model elements, their attributes, or their references. Currently, VITRUVIUS does not allow for concurrent execution of updates. We borrow the concept of database transactions to work towards concurrent updates in VITRUVIUS, while ensuring consistency across all models. Transactions are a unit of database processing, which consist of one or more database access operations. We will therefore refer to a model transaction as a unit of V-SUM processing, consisting of one or more deltas. To define model transactions, we use the *delta metamodel* implemented in VITRUVIUS. The *delta metamodel* defines all possible deltas, i.e., changes, we can perform on an EMF model, such as replacing a reference.

Especially in the context of the development of cyber-physical systems, e.g., as researched in the *Convide* project [7], the performance of a tool like VITRUVIUS becomes crucial. Our research investigates the performance of VITRUVIUS with respect to granularity of model transactions. For this purpose, we focus on two aspects of granularity: transaction size and metamodel topology. Figure 1 shows the two topologies used. In a fan-out topology, one metamodel is connected to several other metamodels, which are not connected further. In a chain topology, metamodels

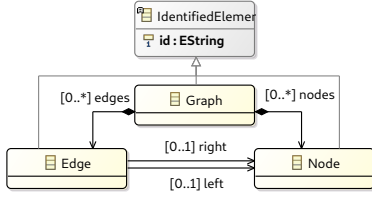


Figure 2: Metamodel of the graph models used for the experiments.

are connected pairwise by consistency specifications, forming a chain. We answer three research questions:

- RQ1** How does the size of a transaction affect the runtime of delta application?
- RQ2** How does the fan-out degree (the number of metamodels connected to the modified metamodel) affect runtime?
- RQ3** How does the length of a chain of metamodels impact runtime?

The remainder of this paper is organized as follows: Section 2 discusses the experiment design, Section 3 presents the results, Section 4 analyzes the findings, Section 5 addresses threats to validity, Section 6 discusses related work, and Section 7 concludes the paper.

2 Experiment Design

To conduct the experiments, we used the simple graph metamodel illustrated in Figure 2. The metamodels for the fan-out and chain scenarios are identical, but with different namespaces in order to implement Reactions for them. In this metamodel, a **Graph** consists of **Nodes** and **Edges**, and an **Edge** references a **left** and a **right Node**. All three metaclasses are of type **IdentifiedElement** and therefore have an **id**. To execute the tests, we used a development version of VITRUVIUS, which is available in our replication package¹ for installation. The experiments were run on a laptop with an Intel Core i7-1260P and 32GB of RAM with Windows 11, OpenJDK 17.0.12 and Eclipse Modeling Tools 2024-06.

Delta Acquisition We employed two different approaches to calculate the necessary deltas. For the experiments, we constructed an empty **Graph**, which results in three deltas (one for creating, one for inserting and one for setting its **id**). Then, depending on the number of deltas required, we add a **Node**, and derive the deltas necessary to add this node. The derivation is necessary because we add the node in a view of the system, which is a state and not a delta-based representation. This also results in three additional deltas.

We derive the deltas per **Node** to ensure they are applicable with the minimum granularity of three to

| # repetitions | 10000 | 1000 | 100 | 10 | 10 |
|---------------|-----------------------|----------------------|--------------------|-------------------|-------------------|
| # deltas | transaction size | | | | |
| | 3 | 30 | 300 | 3000 | 30000 |
| 3 | 0.64 (0.59) | - | - | - | - |
| 30 | 3.53 (1.31) | 0.76 (0.55) | - | - | - |
| 300 | 42.67 (6.43) | 5.09 (2.24) | 1.86 (0.74) | - | - |
| 3000 | 1493.70 (48.04) | 166.60 (6.99) | 25.20 (1.99) | 11.20 (1.83) | - |
| 30000 | 204820.4 (3107.53) | 18417.20 (102.93) | 1993.00 (35.57) | 335.00 (14.21) | 130.30 (11.60) |

Table 1: Granularity: Runtime {mean(deviation)} in ms for the granularity of a transaction as the number of deltas it contains.

| # repetitions | 10000 | 1000 | 100 | 10 | 1 |
|---------------|----------------|----------------|-----------------|---------------------|------------------|
| # meta-models | # deltas | | | | |
| | 3 | 30 | 300 | 3000 | 30000 |
| 1 | 0.46 (0.54) | 0.58 (0.52) | 1.44 (0.59) | 11.00 (3.41) | 121 (0.00) |
| 2 | 0.78 (0.51) | 1.33 (0.56) | 6.63 (1.29) | 104.70 (4.71) | 8926 (0.00) |
| 3 | 1.59 (0.70) | 2.72 (0.98) | 12.11 (1.44) | 310.70 (14.03) | 31989 (0.00) |
| 4 | 2.34 (0.92) | 5.09 (1.47) | 30.38 (5.76) | 852.30 (102.04) | 57171 (0.00) |
| 5 | 2.48 (1.01) | 5.79 (1.61) | 37.79 (8.16) | 1232.90 (114.85) | 103044 (0.00) |

Table 2: Fan-Out: Runtime {mean(deviation)} in ms for different transaction sizes and changing numbers of fan-out connected metamodels.

be able to define transactions with a minimum size of three deltas. Computing the deltas for multiple added nodes at once results in a delta sequence that contains transactions of size 3 that are not applicable to the transaction, but globally for the whole delta sequence. Because we set the minimum granularity to three, we are not using **Edges** in this experiment, because an **Edge** creates 5 deltas. Thus, we could not split, e.g., the delta sequence with 30 deltas into 10 transactions with 3 deltas each. For the fan-out and chain scenario, we create a graph that results in the required amount of deltas and add it to a view, which triggers the consistency preservation, also creating **Edges** in the models.

3 Results

This section answers the three proposed research questions proposed using the data in Table 1, Table 2, and Table 3. The comparison of transaction sizes and the overall number of deltas is presented in Table 1. For both the fan-out scenario, as illustrated in Figure 1, as well as the chain scenario, shown in Figure 1, we used the graph metamodel and isomorphic identical graph

¹<https://zenodo.org/doi/10.5281/zenodo.13643245>

| # repetitions | 10000 | 1000 | 100 | 10 | 1 |
|---------------|----------------|----------------|-----------------|-------------------|-----------------|
| # meta-models | # deltas | | | | |
| | 3 | 30 | 300 | 3000 | 30000 |
| 3 | 1.03 (0.42) | 3.33 (0.99) | 18.05 (4.38) | 388.30 (37.84) | 17115 (0.00) |
| 4 | 1.96 (0.87) | 4.30 (1.66) | 24.22 (5.81) | 691.80 (56.12) | 34821 (0.00) |
| 5 | 1.53 (0.57) | 3.45 (0.80) | 25.33 (1.56) | 793.00 (26.03) | 58995 (0.00) |

Table 3: Chain: Runtime {mean(deviation)} in ms for different transaction sizes and changing numbers of chain-like connected metamodels. The first two rows are identical to the fan-out scenario and thus omitted.

metamodels. The only exception is the namespace URI to tell them apart. This means that the CPRs connecting the metamodels simply map an element to an identical element in another metamodel, e.g., a `Node` from the `graph` metamodel into a `Node` from the `graph1` metamodel. The results for the fan-out scenario are presented in Table 2, and the results for the chain scenario are presented in Table 3, which are discussed in the next section.

4 Discussion

As the size of individual transactions increases, resulting in a reduced number of transactions for a given total number of deltas, system performance improves, illustrated, e.g., in the last row in Table 1. Using 10000 transactions with three deltas each requires over 300 times the execution time of a single transaction with 30000 deltas. We presume this is caused by the overhead required to process each transaction. Using fewer but bigger transactions as opposed to many smaller ones can improve performance, which answers **RQ1**. Table 2 and Table 3 present the answers to **RQ2** and **RQ3**, respectively. In both cases, the performance deteriorates as the number of metamodels connected through consistency preservation rules increases. Our experiments showed that a fan-out scenario with four metamodels performs worse than the chain scenario with five metamodels.

5 Threats to Validity

The data collected from the experiments has several weaknesses, which we discuss here in accordance with [2]. We conducted these experiments to gain preliminary knowledge about factors that may influence the performance of model transactions. A threat to the external validity of the discussion of **RQs 2** and **3**, is the focus on only two extremes: the fan-out and the chain scenario. In realistic scenarios, we will likely encounter a mixture of both. Further threats to external validity are 1) the small number of experiments, 2) using only a single metamodel with a single CPR configuration (although duplicated for more than one metamodel), which produces isomorphic models,

which is also unlikely to happen in practice, and 3) a limited number of experiments. We plan to address these threats by extending our evaluation in future work.

6 Related Work

There are other related approaches to manage consistency, e.g., *ComprehensiveSystems* [4] or *DesignSpaces* [6], where the latter also provides some performance evaluations. The works do not reason about the influence of transaction sizes on the performance of consistency preservation in general but for specific use cases, or do not consider it at all.

7 Conclusion

We presented our investigation into the influence of transaction size and the metamodel topology on the performance of model transactions in *VITRUVIUS*. Our data indicates, that the transaction size influences the performance of *VITRUVIUS*, resulting in a reduction of throughput by a factor of over 300 in one of the experiments conducted. While consistency preservation rules are essential for working with multiple related models, they have a negative effect on performance. Regarding the connections between metamodels, connecting metamodels in a chain has shown to be more efficient than in a fan-out topology. Since we are working on an initial version of a model transaction definition, for future work, we will research additional properties of model transactions, to refine our definition. Moreover, we will develop guidelines for users of *VITRUVIUS* to enhance performance.

Acknowledgements

Authors B. Jutz and Z. Zacouris are funded and T. Weber is supported by the DFG - SFB 1608 - 501798263.

References

- [1] D. Steinberg et al. *EMF: eclipse modeling framework*. Pearson Education, 2008.
- [2] P. Runeson and M. Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical software engineering* 14 (2009), pp. 131–164.
- [3] H. Klare et al. “Enabling consistency in view-based system development—the vitruvius approach”. In: *Journal of Systems and Software* 171 (2021), p. 110815.
- [4] P. Stükel et al. “Comprehensive systems: a formal foundation for multi-model consistency management”. In: *Formal Aspects of Computing* 33.6 (2021), pp. 1067–1114.
- [5] I. David, H. Vangheluwe, and E. Syriani. “Model consistency as a heuristic for eventual correctness”. In: *Journal of Computer Languages* 76 (2023), p. 101223.
- [6] L. Marchezan et al. “Fulfilling industrial needs for consistency among engineering artifacts”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2023, pp. 246–257.
- [7] R. Reussner et al. “Consistency in the View-Based Development of Cyber-Physical Systems (Convide)”. In: *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 2023, pp. 83–84.