JOT
JOURNAL OF
OBJECT TECHNOLOGY

Journal of Object Technology | **RESEARCH ARTICLE**

Temporal Models on Time Series Databases

Alexandra Mazak[*], **Sabine Wolny**[†], **Abel Gómez**[‡], **Jordi Cabot**[§], **Manuel Wimmer**[†], **and Gerti Kappel**[**]

[*]Institute for Subsurface Engineering - Digital Transformation in Tunnelling, Montanuniversität Leoben, Austria
[†]CDL-MINT, Johanens Kepler University Linz, Austria
[‡]Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC), Spain
[§]ICREA, Spain
[**]Business Informatics Group, TU Wien, Austria

ABSTRACT With the emergence of Cyber-Physical Systems (CPS), several sophisticated runtime monitoring solutions have been proposed in order to deal with extensive execution logs. One promising development in this respect is the integration of time series databases that support the storage of massive amounts of historical data as well as to provide fast query capabilities to reason about runtime properties of such CPS.

In this paper, we discuss how conceptual modeling can benefit from time series databases, and vice versa. In particular, we present how metamodels and their instances, i.e., models, can be partially mapped to time series databases. Thus, the traceability between design and simulation/runtime activities can be ensured by retrieving and accessing runtime information, i.e., time series data, in design models. On this basis, the contribution of this paper is four-fold. First, a dedicated profile for annotating design models for time series databases is presented. Second, a mapping for integrating the metamodeling framework EMF with InfluxDB is introduced as a technology backbone enabling two distinct mapping strategies for model information. Third, we demonstrate how continuous time series queries can be combined with the Object Constraint Language (OCL) for navigation through models, now enriched with derived runtime properties. Finally, we also present an initial evaluation of the different mapping strategies with respect to data storage and query performance. Our initial results show the efficiency of applying derived runtime properties as time series queries also for large model histories.

KEYWORDS Runtime Models, Query Languages, Model-Based Analysis, Temporal Modeling, Time Series Databases.

1. Introduction

With the emergence of Cyber-Physical Systems (CPS) and sophisticated runtime monitoring infrastructures, time series databases (Bader et al. 2017) are nowadays frequently applied to store historical data of systems as well as to provide powerful analysis by dedicated query languages.

At the same time, Model-Driven Engineering (MDE) (Brambilla et al. 2017) approaches are a promising line for dealing with the complexity of designing CPS. However, in recent years the scope of MDE has been also extended to runtime aspects of CPS (Mazak & Wimmer 2016; Benelallam et al. 2017; Cruz, Sadovykh, Truscan, Brunelière, et al. 2020; Bencomo et al. 2019; Gogolla et al. 2019; Kästner et al. 2018).

Several approaches for dealing with runtime data in models have been proposed which are often referred to temporal models in analogy to temporal databases (Gómez et al. 2018; Wolny et al. 2018; Bill et al. 2017). Temporal models go beyond representing and processing the current state of systems. By this, they extend research done in the last decades where several dedicated mappings from design models to different database technologies following different data paradigms have been proposed, e.g., see (Gogolla 2005). However, currently there is a lack of approaches which deal with the explicit mapping of design models to time series databases which can be considered as a special type of temporal databases (Schmidt et

JOT reference format:
Alexandra Mazak, Sabine Wolny, Abel Gómez, Jordi Cabot, Manuel Wimmer, and Gerti Kappel. *Temporal Models on Time Series Databases*. Journal of Object Technology. Vol. 19, No. 3, 2020.
http://dx.doi.org/10.5381/jot.2020.19.3.a14

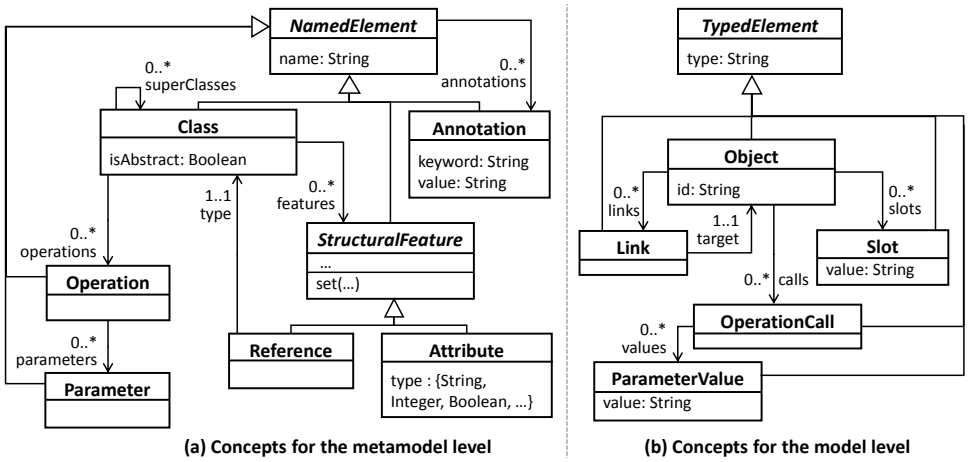

Figure 1 Excerpt of Ecore: (a) concepts for defining metamodels and (b) concepts for defining models.

al. 1995; Böhlen et al. 2018). Such mappings are required to further close the gap between design time modeling activities and simulation/runtime monitoring activities (Gogolla et al. 2019), which employ time series analytics. For instance, time series representations and analytics are foreseen in the development of SysML v2 (Wolny et al. 2020) in order to deal with additional activities in engineering technical systems such as computing different key performance indicators for running systems by applying aggregation functions such as mean, max, mode, etc.

To tackle this limitation, we propose in this paper a novel partial mapping from metamodels and their instances, i.e., the models, to time series databases. The partial mapping deals with the fact that most often not all model elements contribute to a time series, and thus, only those elements which have a runtime history are explicitly mapped to time series structures. Therefore, we propose a dedicated profile for extending the metamodels with appropriate annotations to drive and optimize the generation of model-based time series database connectors. In addition, we propose a Model-to-Time Series (M2TS) mapper that allows to inject data to time series databases from model changes as well as to extract data from the time series databases by model-based queries in OCL (Cabot & Gogolla 2012). By providing these features, we allow for model simulation runs which may be analysed by time series analytics as well as allow for model-based runtime monitoring of systems reporting their changes and states to time series databases. We demonstrate both scenarios by a production system case study and evaluate in particular two mapping strategies with respect to the required data storage as well as query answering performance.

The reminder of this paper is structured as follows. The foundations of this work are introduced in Section 2, namely MDE, in particular, metamodeling, and time series databases. A model to time series mapping approach is proposed in Section 3 incorporating two mapping strategies, which are subsequently evaluated in Section 4 by a case study based on a productions system demonstrator project. Section 5 presents research work marrying MDE-based approaches with temporal aspects concerning linking, versioning, languages, analytics, etc., before we conclude the paper in Section 6 with some directions for future research.

2. Background

In this section, we describe the background of this work, i.e., (meta)modeling and time series databases.

2.1. Metamodeling

Model-driven Engineering (MDE) considers models as first class citizens (Bézivin et al. 2014). A model is used to describe an abstraction of reality for a specific purpose. The basis of such models are modeling languages which are defined by their metamodels. Metamodels are used to describe the abstract syntax of modeling languages. Models created by using a modeling language are instances of the metamodel, and thus, conform to it (Bézivin et al. 2014). One of the best known modeling languages (amongst others) is the Unified Modeling Language[1] (UML) which bases on the Meta Object Facility[2] (MOF) standard. The advantages of UML are platform independence as well as adaption and extension capabilities for users to meet their own requirements for a specific purpose. UML offers a wide range of views and different types of diagrams to represent the structure and behavior of a system to be modeled. One example of a metamodeling language which is based on a core subset of UML and MOF is Ecore from the Eclipse Modeling Framework[3] (EMF). Since Ecore supports the key concepts of using models as input to development and integration tools, it is one of the most widely used languages for code generation and model serialization for data interchange.

In our approach, we focus on Ecore. For illustrating metamodels and models we employ as concrete syntax UML class diagrams and UML object diagrams, respectively. Figure 1 shows excerpts of (a) Ecore's concepts for defining metamodels and (b) Ecore's concepts for representing instances of the metamodels, i.e., models. Models are represented by object graphs and consist of `objects` (instances of `classes`), `slots`

[1] https://www.uml.org
[2] https://www.omg.org/mof
[3] https://www.eclipse.org/modeling/emf

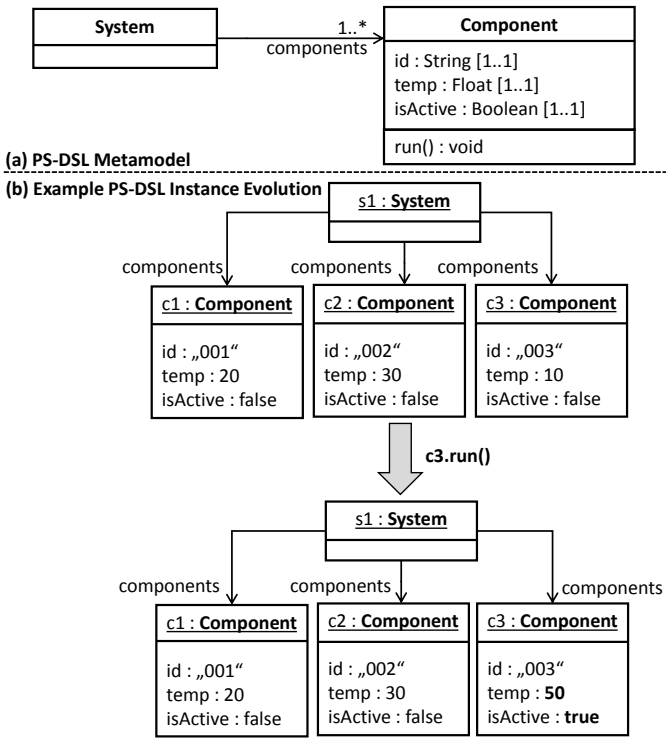

Figure 2 Example: (a) PS-DSL metamodel and (b) model instance evolution.

for storing values (instances of `attributes`), `calls` for executing operations (instances of `operations`) with particular values (instances of `parameters`), and `links` between objects (instances of `references`).

UML object graphs have to conform to the given UML class diagrams. For instance, this means that if an object is existing in the object graph, a corresponding concrete class must exist in the metamodel which act as type for the object.

Additionally, with Ecore, metamodel elements might be annotated with further information (so-called `annotations`), e.g., for tagging elements for particular platforms or purposes as we will see also later in the context of this work.

On the basis of these two metamodels, Figure 2 shows an example of an excerpt of a Production System Domain Specific Language (PS DSL) and an example instantiation of it. The DSL in Figure 2(a) shows that a `system` consists of various `components`. Each component has an unique `id`, a temperature value (`temp`), a property for showing if the component is active or not (`isActive`) and a method `run()` which is active when the component is busy processing an order.

Based on this discussed metamodel, Figure 2(b) shows an instance of a particular system. This system `s1` consists of three different components (`c1`-`c3`) with different property values. If c3 is starting to work (`c3.run()`), the property values of `c3`, more specific the temperature value (`temp`) and the `isActive` value, are changing. Thus, the system state is changing over time and in the shown example only snapshots of the system at a specific point in time are represented (Gogolla et al. 2014).

2.2. Time Series Database

A time series (TS) is a sequence of data points acquired by repeatedly measuring certain parameters (e.g., temperature) over time. The measured values are stored together with the timestamps at which the measurements are taken (Jensen et al. 2017). Although the measurements are usually performed at regular intervals (default in milliseconds), regularity is not a mandatory requirement. The increased interest on this data is in particular the result of the ongoing development in the CPS domain with its IoT technologies as described in the introduction, in which the number of sensors that regularly measure defined conditions is constantly increasing, e.g., for an efficient runtime monitoring.

Time series databases are used for storing, processing, querying as well as analyzing this data generated over time (Bader et al. 2017). Such data consists of timestamps, corresponding values, and optional tags which can consist of names and values (both mostly alphanumeric). Queries can be executed for timestamps or intervals without having to model the data into another structure (Bader et al. 2017). Since the TSDB is not only used for simply collecting data, the term “Time Series Database” (TSDB) is synonymous to the term “Time Series Database Management System” as a kind of software with specialized functions such as compressing or aggregating time series data (Kholod et al. 2017). As mentioned above such time series data is metering from a lot of different sensors. For storing these large amount of data with sufficiently high performance, TSDBs provide the relevant scalability (Jensen et al. 2019).

The level of granularity depends on the type of time series data and the requirements for data analysis, especially since not every time series has to be measured at the same level of detail in order to gain valuable insights of the monitored system (Bader et al. 2017). As an example, the half-hourly measurement of temperature in several rooms of an office building can be mentioned. In this example the granularity is 30 minutes. The values of a tag called “room” can then further specify to which room of the house the measured temperature (value of the time series) belongs.

Time series data differs from other data sets in that it is usually added as a new entry in a TSDB, and therefore, already stored entries are not overwritten (Kholod et al. 2017). Exceptions may only caused by the correction of faulty data, e.g., due to delayed measurements or a failure of sensors. Therefore TSDBs allow the recording and analysis of massive historical data, e.g., for anomaly detection or predictive analytics (Mazak et al. 2018). Thus, any changes over time can be traced nearly in a seamless manner. The storage of time series data, the analysis, and the monitoring of any changes over time provide a great deal of informative added value compared to other types of data, which can only represent a current status (Kholod et al. 2017). In our approach, we use InfluxDB[4] an open source TSDB by which we can continuously store and query data independently of another DBMS (Bader et al. 2017). For querying, it provides a SQL-like language, and for storing it provides rules for (long-

[4] https://www.influxdata.com/products

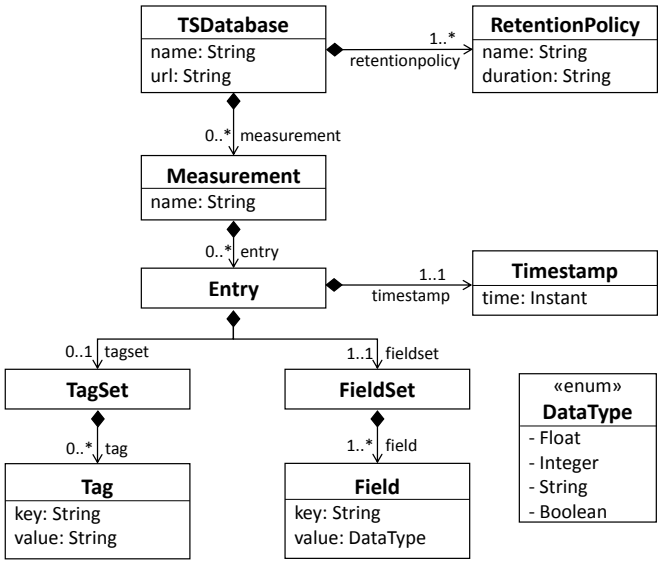

Figure 3 TSDB metamodel.

term) data storage. For instance, InfluxDB enables flexible data aggregations based on the timing factor and running calculations of functions (e.g., average temperature per hour).

Figure 3 shows a metamodel of the TSDB. The `TSDatabase` has a specific `name` and consists of various `Measurements`. Each measurement must consist of a `Timestamp` and a `FieldSet` where the different time series values are stored. Optionally, the measurement can have some additional meta-information stored in a `TagSet`. For instance, InfluxDB implements this metamodel and its line protocol informs the database of the measurement, tag set, field set, and timestamp. Listing 1 shows the structure of the line protocol, with first its `measurement`, followed by a optional `TagSet`, followed by a `FieldSet` with at least one field and optionally a `timestamp`. If no timestamp is specified, the current system time is taken by default.

```
<measurement>{,<tag_key>=<tag_value>}␣<field_key>
=<field_value>{,<field_key>=<field_value>}␣[
<timestamp>]
```

Listing 1 Example of the line protocol of InfluxDB.

3. Mapping Models to Time Series Representations

In order to allow an integration of time series storage and analysis in a model-based manner, in this section we present the design rationale for our approach before we outline two mapping strategies from object-oriented models (as described in Section 2) to TSDB.

3.1. A Polyglot for Combining Models with Time Series Databases

For combining models, especially EMF-based models, with TSDB, we aim for a polyglot solution where the static information resides in the model as it is already available, e.g., by

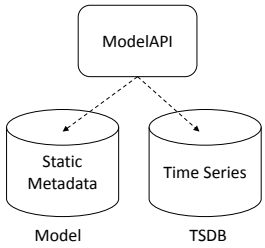

Figure 4 Polyglot solution for models on TSDB.

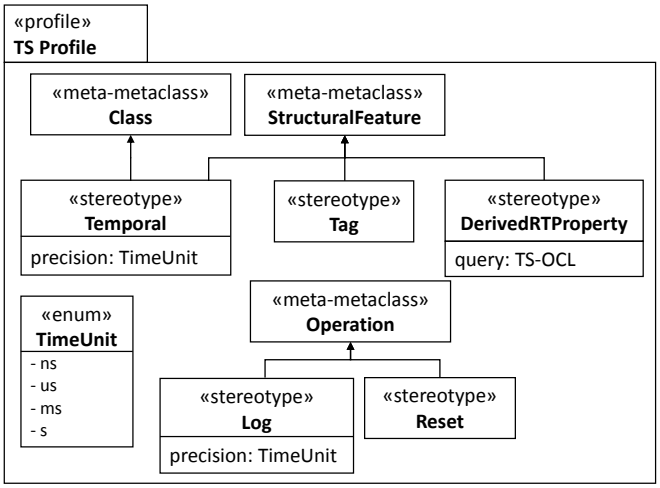

Figure 5 Time Series Profile based on TemporalEMF (Gómez et al. 2018)

XMI or other model persistence mechanisms, and only the time-sensitive information is stored in the TSDB (see Fig. 4).

These two storage parts are combined by a common `ModelAPI`, which then can be accessed and used by various applications. This unifying API abstracts implementation details and allow for a similar way of working with models as it is provided by EMF out-of-the-box. In particular, we reuse as much as possible and only extend those parts which are really required. As a result, the model is applied as close as possible to the EMF standard and the required information for the TSDB can be attached in a light-weight manner. In order to achieve such unifying API with a polyglot there are various requirements that must be fulfilled. First of all, there has to be a built-in mechanism that determines which information from the model should be transferred to the TSDB and stored there. Second, there should be as well a procedure that extracts data from the TSDB by querying and displaying it back in the model. Third, our temporal extensions should not hinder or pollute the use of models and they should be still manipulated as before. The main goal is to embed this process into a conceptual schema to avoid hard coding the functionalities again and again for different cases. In the following subsections, we describe the design choices of the polyglot.

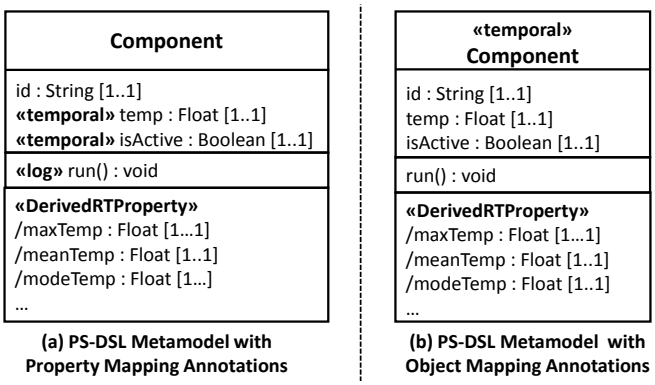

Figure 6 Annotated example metamodel: (a) single property mappings vs. (b) complete object mapping.

3.2. Time Series Profile

In a first step, we propose a profile (realized with EMF Annotations) for extending existing metamodels by time series aspects (cf. Figure 5). The profile defines different kinds of stereotypes for `Classes`, `StructuralFeatures` as well as `Operations`. The stereotype `Temporal` indicates that these elements (classes or structural features such as attributes and references) are temporal features that should be recorded as time series in a TSDB. With the use of `precision`, the accuracy of the recordings can be defined from nanoseconds (`ns`) to seconds (`s`). The stereotype `Tag` should be used for features that represent important metadata of time series features. For instance, the `id` of the room where the temperature is measured. The stereotype `DerivedRTProperty` marks features as derived runtime properties. Derived properties are features where the feature value is computed based on other feature values. Our stereotype is used for defining properties that get their values during runtime based on runtime data stored in the TSDB. For instance, the average temperature of a specific room over a whole day. We introduce this custom stereotype for our derived runtime properties since we use an OCL dialect that need further processing before execution (cf. Section 3.4). Additionally, there are two stereotypes for operations: stereotype `Log` is for logging the start and the end of an operation and stereotype `Reset` is used to refresh the system state, e.g., for a new simulation run.

In Figure 6, we show an example application for the previously presented metamodel in Section 2. In particular, we show on the left hand side the usage of the profile to map on a fine grained level two properties as temporal while on the right hand side we configure the whole class as temporal. These two usages are reflecting the two mapping strategies which are explained next.

3.3. Mapping Strategies

Based on the afore presented profile, we now establish the mapping between models and the TSDB. By this mapping, (i) the traceability between design and runtime activities should be enabled, (ii) and runtime information (i.e., time series data) should be retrieved and should be accessible through models. For this purpose, it must be decided how objects, slots, operation calls, and links from the models, i.e., object graphs (conform to classes, attributes, operations and references, respectively) are mapped to the TSDB elements such as measurements, tags, and fields.

In this paper, we consider two conceptual strategies for the M2TS mapper: (i) a strategy where it is possible to store each temporal property individually (cf. Section 3.3.1), and (ii), a strategy by which the whole object with all its associated information is stored (cf. Section 3.3.2). Of course, there exist various other combinations of strategies additionally to these two discussed ones. However, selecting a suitable strategy depends on performance, memory size, and general feasibility. In this paper, we focus on the presented ones, since they give already several configuration opportunities for modelers and consider the capabilities of the deployed TSDB functions. The developed TS profile is designed to cover both strategies, but is not limited to them. It can be extended as well as the mappings may be adapted to specific strategy changes.

3.3.1. Single Property Mappings

The first strategy is to map single properties, e.g., the temperature of a room. The goal is to continuously log the progression of such property values in a TSDB and to query these values in terms of the models if necessary. This means that the property becomes a "temporal feature" with its own measurement. Only such time relevant data is stored in the TSDB. The remaining information, such as static metadata, is stored in the model, since such information is constant and does never change over time. However, to ensure that properties of different objects can be distinguished and that the relationship between them is not lost, the information to which object the temporal feature belongs must also be stored in the measurement. For example, if the average temperature of a specific room is required, it should be avoided that the average temperature of all rooms is analyzed. Therefore, the room `id` is important to be related to the measurement.

The table in Appendix A shows the mapping of the object diagram elements to the specific elements of the TSDB based on the TS profile. It has to be mentioned that tags are optional additional information and fields are mandatory for value recording. Similarly, each measurement in the TSDB contains a mandatory timestamp, where precision can be used to determine the accuracy. For `DerivedRTProperties` a query is executed on the TSDB which returns a `series` as a result (cf. Section 3.4).

```
1 ON
2 object.set(feature, value)
3 IF
4 feature.isTemporal
5 THEN
6 time = UnixTimestamp(precision = <<feature.temporal.
      precision>>) //default:ns
7 db.insert(measure=<<feature.name>>, tag1=['object',
      <<object.id>>] field1=['value',<<value>>] time)
```

Listing 2 Template for single property mapping.

A simple pseudo code line protocol template for the mapping of a temporal feature embedded in an ECA rule is shown in Listing 2. If a temporal property is set, then the new value is stored as a field in the TSDB, the object ID as tag, and the timestamp as Unix timestamp with the defined precision.

Additional to this example, annotated tags will be added to temporal features of the same object.

Based on the example of Figure 2(b), the following listings show the entries of the different measurements in the database. Listing 3 shows the stored values of the three different components `c1-c3`, before executing c3.run(). For the two temporal properties `isActive` and `temp` the initial values are stored with their timestamp.

Listing 4 shows the values of the different measurements after starting c3.run(). There is a new measurement for method `run`. Additionally, `isActive` is changed to true for component c3 and also the `temp` value changes, and therefore, in both measurements a new entry for c3 is added.

```
database: s1
measurements: isActive, temp

measurement isActive:
time                 object     value
____                 _______    _____
1589406897116594100  40170008   false
1589406897196737400  1443055846 false
1589406897207745200  502838712  false

measurement temp:
time                 object     value
____                 _______    _____
1589406897184730200  40170008   20
1589406897201751200  1443055846 30
1589406897213821300  502838712  10
```

Listing 3 TSDB entries for the single property strategy before c3.run() is executed.

```
c3.run()
measurements: isActive, temp, run

measurement isActive:
time                 object     value
____                 _______    _____
1589407171078884600  40170008   false
1589407171166080300  1443055846 false
1589407171178340200  502838712  false
1589407171200396800  502838712  true

measurement temp:
time                 object     value
____                 _______    _____
1589407171155071700  40170008   20
1589407171172226700  1443055846 30
1589407171184344500  502838712  10
1589407171206409400  502838712  50

measurement run:
time                 object     value
____                 _______    _____
1589407171190348700  502838712  start
```

Listing 4 TSDB entries for the single property strategy after c3.run() is executed.

3.3.2. Complete Object Mappings The second strategy does not map single properties of objects in isolation but rather the entire object at once. This means that individual properties do not have to be annotated as temporal features, but the containing classes, and thus, the associated objects with their properties are stored in the database as measurements.

The table in Appendix B gives an overview of the complete object mapping strategy, how the individual annotated elements from the model are stored in the TSDB. Based on this, Listing 5 shows a pseudo code line protocol template, again embedded in an ECA rule, for storing complete objects as measurements and their features as fields. In addition, if structural features are annotated as tags, then they would be saved as tags in the object measurement.

```
ON
object.set(feature, value)
IF
object.class.isTemporal
THEN
time = UnixTimestamp(precision = <<object.class.
    temporal.precision>>) //default:ns
db.insert(measure=<<object.id>> FOREACH(f in
    <<object.features>>){fieldN=[<<f.name>>,
    <<f.value>>]} time)
```

Listing 5 Template of complete object mapping.

In comparison to the presented single property mapping (cf. Section 3.3.1), Listing 6 and Listing 7 show the set-up of the database and its entries for the complete object mapping. The structure of the information has changed and therefore also the structure of the TSDB queries (cf. Section 3.4) depends on the corresponding mapping.

```
database: s1
measurements: obi40170008, obj1443055846,
    obj502838712

measurement obj40170008:
time                 isActive     temp
____                 _______      _____
1589406897116594100  false        20

measurement obj1443055846:
time                 isActive     temp
____                 _______      _____
1589406897196737400  false        30

measurement obj502838712:
time                 isActive     temp
____                 _______      _____
1589406897207745200  false        10
```

Listing 6 TSDB entries for the complete object strategy before c3.run() is executed.

```
c3.run()
measurements: obj40170008, obj1443055846,
    obj502838712, run
measurement obj40170008:
time                 isActive     temp
____                 _______      _____
1589406897116594100  false        20

measurement obj1443055846:
time                 isActive     temp
____                 _______      _____
1589406897196737400  false        30

measurement obj502838712:
time                 isActive     temp
____                 _______      _____
1589406897207745200  false        10
1589407171200396800  true         10
```

```
1589407171206409400  true          50

measurement run:
time                 object          value
----                 ------          -----
1589407171190348700  obj502838712    start
```

Listing 7 TSDB entries for the complete object strategy after c3.run() is executed.

3.4. Query Capabilities

On the basis of the TS profile and the applied mapping strategies, we now present the query capabilities of our approach. In a first step, we offer four basic operations for temporal properties, the first two are adapted from previous work (Gómez et al. 2018), and the last two are extensions:

(1) *getValueAt(Instant t)*
Result: DataType value
(2) *getValueBetween(Instant t1, Instant t2)*
Result: Map(Instant time, DataType value)
(3) *getTimePointsforValue(DataType value)*
Result: List(Instant time)
(4) *getTimePointsforValueBetween(DataType value1, DataType value2)*
Result: Map(Instant time, DataType value)

Based on the used mapping strategy, the query implementation in the background, i.e., in the TSDB, differs, since there is a different data structure used in the TSDB. For instance, the following Listing 8 shows the difference for the basic operation (1).

```
Single property mapping:
getValueAt(Instant t) {
    db.exeQuery(SELECT value FROM <<feature.name>>
    WHERE object = <<object.id>> and time = t)
}

Complete object mapping:
getValueAt(Instant t) {
    db.exeQuery(SELECT <<feature.name>> FROM
    <<object.id>> WHERE time = t)
}
```

Listing 8 TSDB query for getValueAt(Instant t) based on single property mapping and complete object mapping.

On the basis of the four defined operations, it is now possible, e.g., to calculate the utilization of a component within a defined period of time directly using Java. Listing 9 shows a pseudo code snippet for such a metric calculation.

```
Map<Instant, Boolean> map = c1.isActiveT.
    getValueBetween(t1, t2);
Duration total = Duration.between(t1, t2);
Duration active = 0;
Instant[] keys = map.keySet().toArray();
for(int i=0; i < keys.length-1; i++){
  if(map.get(keys[i]))
    active+= Duration.between(keys[i], keys[i+1]);
}
float utilization = active / total;
```

Listing 9 Pseudo code for the calculation of the utilization time of a component.

Additionally to these four basic operations, derived properties can be annotated with `DerivedRTProperty(query:TS-OCL)`. As a first realization, the `TS-OCL` query must be expressed in the syntax of the query language of the TSDB (in the InfluxDB case it is Influx QL), or in combination with standard OCL[5] for navigation through the model (i.e., using self, naviation operatores, etc.) The combination of OCL with Influx QL is preformed as a pre-processor approach. OCL is used to query the model elements which are injected into the InfluxQL query. The M2TS mapper is resolving the model elements to database entries and thus completes the InfluxQL query.

As an example, we consider as a derived property the maximum temperature of a component. Listing 10 shows the `TS-OCL` query for this example and the respective conversion to a `TSQuery` based on the two different mapping strategies.

```
DerivedRTProperty: MaxTemperature
    TS-OCL=SELECT max(<<self.temp>>) FROM <<self.
    temp>>

Single property mapping:
    TSQuery=SELECT max(value) FROM temp WHERE object
    =<<object.id>>

Complete object mapping:
    TSQuery=SELECT max(temp) FROM <<object.id>>
```

Listing 10 Example query code of a derived runtime property.

These query capabilities enable the M2TS mapper not only to inject data to the TSDB from model changes, but also to extract data from the TSDB by model-based queries. As the derived runtime properties are in essence standard derived properties, they can be simply reused in standard OCL queries. Finally, the combination of OCL with Influx QL allows to write model-based queries without having to deal with the concrete mapping approach in use.

4. Evaluation

In this section, we present and discuss the performance and scalability of our approach using a case study based on the PS-DSL metamodel (cf. Section 2, Figure 2 (a)). From a methodological view, we follow the guidelines for conducting case studies by Runeson and Höst (Runeson & Höst 2009) for performing the evaluation. The implementation of our approach and evaluation results can be found at our project website[6].

4.1. Research Questions

Our general evaluation interest is the comparison of the two presented mapping strategies for our M2TS mapper on basis of performance and scalability. Therefore, we aim to answer the following research questions (RQs):

RQ1—Scalability of the database size with single property mapping vs. complete object mapping: How does the database size develop regarding different number of model changes and number of entries? Does the database size grow linear to model

[5] https://www.omg.org/spec/OCL
[6] https://cdl-mint.se.jku.at/case-study-artefacts-jot-2020/

changes? Is there a significant difference between the two mapping strategies?

RQ2—Performance of the runtime queries for single property mapping vs. complete object mapping: How long do the queries take for (i) values at a given timestamp, (ii) timestamps for specific values, and (iii) aggregate calculations such as average, maximum, and modal values? Is there a significant difference observable for the two mapping strategies?

4.2. Case Study Design

Requirements: As an appropriate input for our case study, we first require a system based on an Ecore model, which is annotated by our TS profile. The corresponding models must be executable and contribute to time series when executed. In addition, we require InfluxDB as running TSDB to store value records.

Setup: For our evaluation, as already mentioned, we use instances based on the PS-DSL metamodel. Our execution system consists of different numbers of components and the run method of each component is executed for various numbers of time. Table 1 gives an overview of the different evaluation settings regarding number of components, number of runs, and number of entries in the TSDB. For instance, one setting consists of 100 components, 100 runs are executed for each component, and finally 80000 entries are stored in the TSDB. During simulation, the values of the properties `isActive` and `temp` are changing over time and logged in the TSDB based on the respective mapping strategy.

No.	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
Comp.	100	1 000	2 000	3 000	4 000	5 000
Run()	100	1 000	2 000	3 000	4 000	5 000
Entries	80 K	8 M	32 M	72 M	128 M	200 M

Table 1 Number of (i) components in the model, (ii) run() executions for different settings, and (iii) entries in the TSDB.

For these settings, the query performance is evaluated as follows. On the one hand, for the different derived runtime properties, i.e., the maximum, mean, and mode values of the temp attribute for a selected component is calculated, and on the other hand, the general methods provided by our approach *getTimePointsForValue* and *getValueAt* are executed for particular values and time points.

For answering our RQs, we calculate the different durations for storing data, and for each query by System.getNanoTime() in Java based on nanoseconds (ns). The performance is measured on an Acer Aspire VN7-791 with an Intel(R) Core(TM) i7-4720 HQ CPU@2.60 GHz 2.60 GHz, with 16 GB of physical memory, and running Windows 8.1. 64 bits operating system. Please note that we measured the CPU time by executing each mapping five times for all different settings and calculated the arithmetic mean of these runs. We use EMF, JDK 13 (important for precision accuracy of nanoseconds), and InfluxDB 1.8.0 to execute our approach.

Prototype: In a first prototypical implementation, we realized our M2TS mapper for EMF. In particular, we provide annotations for the metamodeling language Ecore with respect to utilizing the TSDB InfluxDB. The different stereotypes of our TS profile are implemented as EAnnotations on the Ecore model. For connecting the database, we make use of the open source Java client for InfluxDB[7] and provide our own InfluxDBConnector which provides the glue between EMF models and InfluxDB. For automation purposes, we adapt the existing Java Emitter Templates (JET) for the EMF code generation. Thus, by the extended code generator we are able to provide an enriched API for EMF models to deal with temporal information, i.e., storage and query capabilities.

4.3. Results

In this subsection, we present the measurements for answering our research questions.

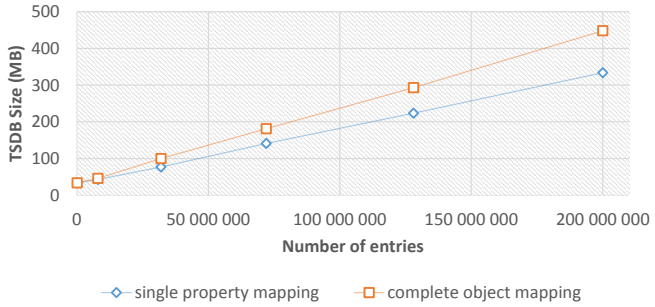

Figure 7 Database size in relation to number of entries for both mapping strategies (in MB).

Answering RQ1 - Scalability of database sizes: Our investigations regarding the TSDB size on the basis of the model changes show that both strategies show a linear increase (cf. Figure 7). It can be recognized that the size of the database for complete object mapping strategy increases slightly faster than for single property mapping strategy. However, this can be explained by the fact that whenever a value of a property is changed, the entire object is stored with a new timestamp.

Answering RQ2 - Performance of runtime queries: Figure 8 shows the measurements of the query duration for both mapping strategies. In general, the queries are fast, as they take only from 1ms to about 7ms, depending on the entries in the TSDB. However, as the size of the database increases, the queries for `MeanMaxMode` and `GetValueAt` for the single property mapping become slightly slower than in the case of the complete object mapping. This can be explained by the fact that starting from a certain number of entries, it plays a role whether the possible results have to be selected first (for single mapping using `object.id`), or are already selected and only need to be screened (for the complete mapping strategy, the object has its own measurement). However, based on a hypothesis testing (i.e., Wilcoxon rank-sum testing (Venables & Ripley 2002)), there is no significance regarding the difference between the

[7] https://github.com/influxdata/influxdb-java

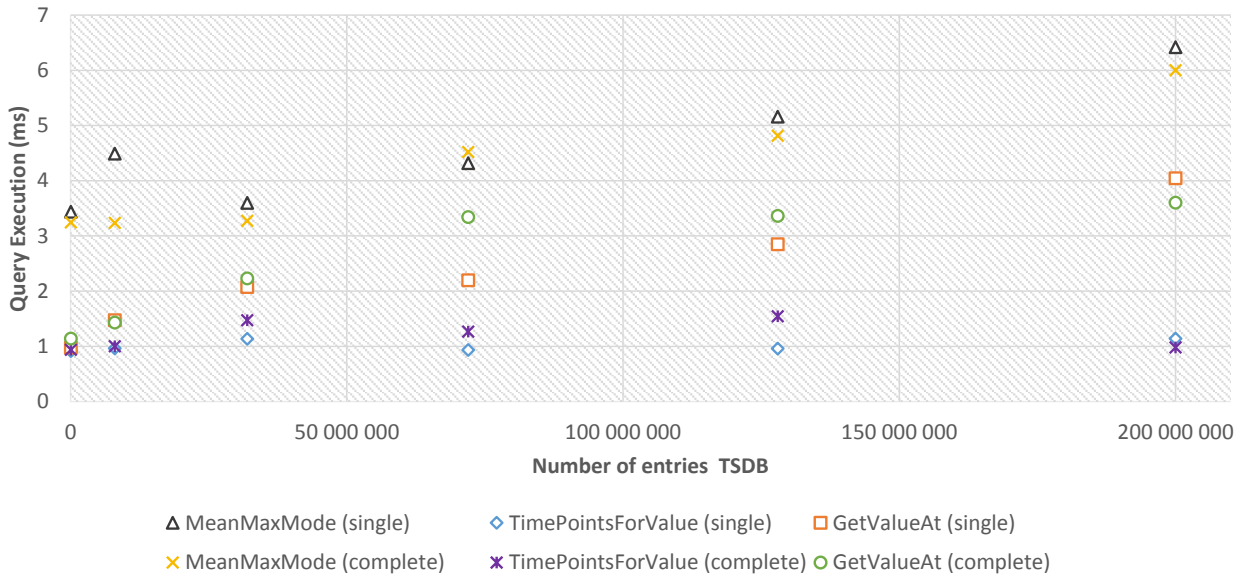

Figure 8 Query execution time in relation to number of entries for both mapping strategies (in ms).

two mapping strategies: p-value =0.4848, H0: single property = complete object; p-value>0.05. Therefore, H0 is not rejected.

4.4. Critical Discussion

In summary, the introduced mapping strategies both have advantages as well as limitations. The right choice of strategy mainly depends on the task to be accomplished, i.e., which queries are subsequently evaluated. Imagine you aim to query all instances for a given type. This would be straightforward for the complete object mapping strategy. Imagine you would like to query the max temperature for all components. This would be faster for the single property mapping.

Overall, the evaluation demonstrated the feasibility of both strategies concerning the data storage and query performance. However, we cannot generalize our results beyond our initial case study. First, we have to mention that there may be other cases where larger objects, i.e., objects having many slot values and links, have to be stored. Consequently, a higher size of the databases may be expected, and this may call for new strategies of mapping objects with only a partial subset of their slots and links. In future studies, we plan to evaluate settings in a larger context such as building a monitoring systems for an IoT network or building a runtime-based verification tool as well. Such studies will allow a more practice-oriented evaluation of the different strategies which may be further collected in a particular benchmark for temporal models.

Another feature to exploit from the TSDB may be the down sampling capability for data for given time frames, i.e., aggregating data from millisec to sec to minute ranges and so on. Again, this feature has to be evaluated in future work. Moreover, the explicit usage of tags instead of fields has to be evaluated in future work as it may have impacts on both: the data storage size and the query performance.

Finally, we would like to mention that our presented case study with all the corresponding artefacts is provided online and may be used by the research community as experimental test bed for future studies concerning finding appropriate mappings from models to time series databases.

5. Related Work

With respect to the contribution of this paper, namely, the mapping and connection of conceptual models to TSDBs, we discuss various threads of related work. First, we discuss temporal modeling approaches for validation and verification of models. Second, we present approaches for linking design and runtime models. Third, we explore approaches for versioning of models in temporal repositories. Finally, we discuss approaches that combine modeling languages with time series analytics.

5.1. Temporal Modeling Languages

There is abundant research on temporal extensions for modeling languages to specify the temporal characteristics of the system data (e.g., consider (Gregersen & Jensen 1999) for a survey), but not regarding the temporal dimensions of models themselves.

Further works advance these first attempts by extending also the query languages with temporal properties, mainly to enable the validation and verification of temporal properties on the data. Temporal OCL (TOCL) (Ziemann & Gogolla 2003) and Temporal UML (Cabot et al. 2003) are two examples of OCL extensions for the evaluation of temporal constraints.

Temporal extensions have also been applied to specific types of systems (e.g., adaptive systems (Mouline et al. 2018)) and DSLs (e.g. timed Petri nets (Bender et al. 2008)). Even TOCL, which can be seen as a generic language, can also be used as an component in other DSLs as described in (Meyers et al. 2014). In this line, (Bousse et al. 2019) discuss and apply a pattern to extend modeling languages with events, traces, and further runtime concepts to represent the state of a model's execution and to use TOCL for defining properties that are verified by mapping the models as well as the properties expressed in TOCL

to formal domains that provide verification support. Efficiency of these types of temporal inspection queries is also the focus of (García-Domínguez et al. 2018) and (García-Domínguez et al. 2019).

Nevertheless, all these approaches (including our own previous TemporalEMF proposal (Gómez et al. 2018)) are mostly oriented towards the retrieval of specific past states of the model/data, elaborating on the concepts of *valid time* and *transaction time* of (bi)temporal models. Instead, in this work we explicitly focus on the support for complete time series storage and analysis, which opens the door to more powerful and rich possibilities, like the computation of different KPIs for models as part of design exploration and simulation scenarios.

5.2. Linking Design-time and Runtime Models

In this subsection, we discuss approaches using traceability between design and runtime models. The evolutionary aspect of engineering artifacts refers to the fact that they change over time. Models in engineering processes, e.g., usually develop from initial ideas to first drafts. They are then continuously revised, often by taking into account feedback from other resources, until they are finally released. However, also the feedback after the release from the operation should be reflected in those models to make traceability between design and operation feasible (Mazak & Wimmer 2016).

For this purpose, the authors of (Wolny et al. 2018) present an architecture to map runtime data back to the model level by using standard metamodeling techniques. Thereby, they do not only develop a unifying architecture for creating model snapshots on-the-fly, but to map the history of operation concerning certain properties. This allows to specify and compute runtime properties based on time series data through design models. This means, design-oriented languages are equipped with extensions for representing runtime states as well as runtime histories, which in turn allow the formulation and computation of runtime properties with OCL. This makes it feasibly to directly interpret measurements within design models without introducing an impedance mismatch. The challenge with using OCL for this purpose is that even simple mathematical calculations (e.g., computing upper bounds or averages) may quickly become complex with respect to their definition and evaluation. For better scalability such calculations should be directly performed in the TSDB as we allow by the presented work of this paper.

If the design model is not yet coupled with its runtime counterpart, i.e., no annotations are made at model level, the authors of (Wolny et al. 2019) present an approach to transform raw sensor log data to UML sequence diagrams for graphical representation. Therefore, they provide a text-to-model transformation to transform text-based traces of a running system to UML sequence diagrams. As a basis for reconstructing such UML sequence diagrams, they develop a metamodel for representing system logs in an object-oriented manner. This makes it feasible to express system logs explicitly as models. However, they only use the time aspect to trace the correct order of the performed operations, but not to store the execution time, e.g., to be able to annotate information about average duration. This could be complemented by the approach presented in this paper.

Another project that also deals with the connection of design and runtime is the project MegaM@Rt2[8]. In this scalable model-based framework for continuous development and runtime validation of complex systems trace links between design models and runtime are established based on bidirectional transformations (Cruz, Sadovykh, Truscan, Bruneliere, et al. 2020). Temporal aspects as we discuss in the context of this paper are not explicitly considered. However, the MegaM@Rt2 approach is applicable for already existing systems which may be combined with our approach to enrich existing systems with TS collection and analysis.

5.3. Temporal Model Repositories

In (Bill et al. 2017), the authors discuss the need for temporal model repositories and the explicit representation of time in models. They discuss the gap of traditional Version Control Systems (VCS) such as SVN and Git, where each version of an evolving model is stored with a timestamp for the whole model (Altmanninger et al. 2009). While versioning the whole model is suitable for many development tasks, it makes it challenging to trace the evolution of specific model elements over time. Furthermore, the authors discuss several challenges when moving towards temporal model repositories such as (i) model storage, (ii) model access, (iii) model consistency, (iv) model manipulation, and (v) model visualization. In this paper, we have mostly focused on the first two points.

In the work presented in (Hartmann et al. 2014), the authors present an approach for versioning on the model element level. They discuss the lack of native mechanisms in MDE as well as Models@run.time to handle the history of data. They state that especially for the Models@run.time paradigm (Blair et al. 2009), which propagates the use of models to support runtime reasoning, an efficient mechanism is needed to store and navigate the history of model element values. Therefore, model elements have to be versioned independently from each other. Furthermore, they simplify and improve the performance of navigating between model elements coming from different versions by defining a navigation context for navigating in two dimensions (space and version). However, the versions have to be explicitly introduced and managed as in the aforementioned versioning systems. In our approach, we store individual model element or even individual properties with their associated timing aspects.

To tackle the discussed challenges in (Bill et al. 2017), in (Gómez et al. 2018) we present a temporal model infrastructure built on top of EMF—*TemporalEMF*. In summary, we showed how TemporalEMF enables to treat conceptual schemas as temporal models. On these models, temporal queries can be performed to retrieve model contents at different time points, e.g., to compare model content and to trace model states in the past. The TemporalEMF approach bases on concepts from temporal languages. The history of a model is transparently stored in a NoSQL database (i.e., HBase[9]). In our newly presented approach no dependence to other DBMS, such as NoSQL ones, is needed, since we use a TSDB to reason about the history of property values accessible in the model.

[8] https://megamart2-ecsel.eu/
[9] http://hbase.apache.org/

In (Haeusler et al. 2019), the authors discuss the need for tool support in the area of IT Landscape documentation. Therefore, they present a solution for storing, versioning, and querying of such IT Landscape models by means of an open source graph-based EMF model repository. In addition, the modular architecture allows to consider those models still as standalone components outside the repository context. A limitation is that *ChronoSphere* operates in local deployments, and therefore, is currently not distributed across several machines for greater scalability. In our approach, models and the TSDB can run separately on different machines. Since, our polyglot approach provides a ModelAPI, it could be considered in the ChronoSphere repository as well, which is generic, and therefore, not limited to the domain of IT Landscape documentation. This allows other applications to use the provided functionalities of our API for various use cases.

5.4. Modeling Languages for Time Series Analytics

In (David et al. 2012), the authors present the OMS3[10] modeling framework which provides an extensible and lightweight layer for simulation description expressed as so-called "Simulation DSL" based on Groovy[11]. The authors propagate DSLs for completing General Purpose Languages (GPLs) for specific simulation purposes. In their work, they present DSL variants in OMS3 such as a DSL for Ensemble Streamflow Prediction (ESP) based on meteorological time series data for predicting future conditions. Instead of creating a DSL for a specific purpose, in our approach, we propose a dedicated profile for extending metamodels with appropriate annotations to extend existing metamodels (e.g., of GPLs) by time series aspects.

Gekko[12] is an open source modeling approach for time series data management and for solving as well as analyzing large-scale time series models. It could be considered as a kind of DSL with a strong time series domain focus. It provides interfaces to statistical computing and graphics packages such as R[13]. In our approach, we use InfluxDB which offers besides high-availability storage and monitoring of time series data, application metrics as well as real-time analytics.

In this context, we also mention TimescaleDB[14] which is an extension of PostgreSQL[15]. TimescaleDB is specially optimized for time series data in order to automatically partition data by time. Like PostgreSQL, TimescaleDB stores the data in a RDBMS and supports SQL as query language. Furthermore, it provides additional features for analyzing and manipulating time series data. Similar to the InfluxDB, TimescaleDB offers the possibility of a continuous calculation of functions. In particular such functions are queries that are executed continuously and in real time on the incoming data. The results of these regular queries are also stored in the TSDB as specified metrics (e.g., average room temperature every half hour with the applied metric). External tools such as Grafana[16] or Tableau[17] may also be used to visualize and analyze time series data. In addition to Grafana, the open source statistics software R for analyzing time series data should also be mentioned. However, the probably most extensive functionalities for querying data, setting warnings, and visualizing time series data is offered by InfluxDB, respectively by the InfluxData platform. Moreover, long-term storage of data is only provided by InfluxDB, and only to a limited extent by TimescaleDB. Additionally, the data scripting and query language Flux[18] can be used in combination with InfluxDB. This standalone tool is optimized, e.g., for monitoring and provides built-in functions as well as importable packages to retrieve, transform, process, and output time series data. In contrast, our approach looks at TSDBs from a model-driven perspective and how conceptual modeling and TSDBs can benefit from each other.

6. Conclusion and Future Work

In this paper, we have presented a novel set of partial mappings from models to TSDB. In particular, we presented a profile to annotate metamodels in order to automatically generate wrappers to time series databases that enable storing model updates as well as querying historical model information. Two different mapping strategies are proposed and evaluated in terms of their feasibility and scalability. While the current work presents interesting insights how modeling technologies may be combined with TSDB, we foresee several additional lines of research worth to investigate in addition to the ones mentioned in the evaluation section.

On the modeling side, we need to deal with co-evolution issues given that the TSDB is schema-less. For usability reasons, we would also like to be able to express complex time-related queries in OCL (e.g., by pre-defining a set of time-series operators, similar to what we did in (Cabot et al. 2010) for multidimensional models).

On the mapping side, we will investigate how to run approximate queries to deal with a variety of uncertainty scenarios (Burgueño et al. 2019) and study the potential of combining both temporal and time-series information. This would enable even more complex analysis where we could, for instance, evaluate whether a new design model behaves better than one we used in the past by comparing their respective associated time-series data. It even allows to forecast the expected behavior of future designs. Finally, we are interested in mapping and storing not only the models themselves but also all modeling operations on them (e.g., by storing the trace information automatically created by some transformation engines such as ATL).

Acknowledgments

The work has been supported by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development (CDG) and by the Austrian Federal Ministry for Education, Science and Research

[10] https://alm.engr.colostate.edu/cb/wiki/16961
[11] https://groovy-lang.org
[12] http://t-t.dk/gekko
[13] https://www.r-project.org
[14] https://www.timescale.com
[15] https://www.postgresql.org
[16] https://grafana.com
[17] https://www.tableau.com
[18] https://www.influxdata.com/products/flux

in the TransIT Project (BMBWF-11.102/0033-IV/8/2019) as well as by the FWF under the Grant Numbers P28519-N31 and P30525-N31, and from the Spanish government under project *Open Data for All* (RETOS TIN2016-75944-R).

References

Altmanninger, K., Seidl, M., & Wimmer, M. (2009). A survey on model versioning approaches. *IJWIS*, *5*(3), 271–304. Retrieved from https://doi.org/10.1108/17440080910983556 doi: 10.1108/17440080910983556

Bader, A., Kopp, O., & Falkenthal, M. (2017). Survey and Comparison of Open Source Time Series Databases. In *Workshopband der 17. fachtagung des gi-fachbereichs datenbanken und informationssysteme (dbis) datenbanksysteme für business, technologie und web (BTW 2017)* (Vol. P-266, pp. 249–268). GI. Retrieved from https://dl.gi.de/20.500.12116/922

Bencomo, N., Götz, S., & Song, H. (2019). Models@run.time: a guided tour of the state of the art and research challenges. *Software and Systems Modeling*, *18*(5), 3049–3082. Retrieved from https://doi.org/10.1007/s10270-018-00712-x doi: 10.1007/s10270-018-00712-x

Bender, D. F., Combemale, B., Crégut, X., Farines, J., Berthomieu, B., & Vernadat, F. (2008). Ladder Metamodeling and PLC Program Validation through Time Petri Nets. In *Proc. of the 4th european conference on model driven architecture - foundations and applications, ECMDA-FA 2008* (Vol. 5095, pp. 121–136). Springer. Retrieved from https://doi.org/10.1007/978-3-540-69100-6_9 doi: 10.1007/978-3-540-69100-6_9

Benelallam, A., Hartmann, T., Mouline, L., Fouquet, F., Bourcier, J., Barais, O., & Traon, Y. L. (2017). Raising Time Awareness in Model-Driven Engineering: Vision Paper. In *Proc. of the 20th ACM/IEEE international conference on model driven engineering languages and systems, MODELS 2017* (pp. 181–188). IEEE Computer Society. Retrieved from https://doi.org/10.1109/MODELS.2017.11 doi: 10.1109/MODELS.2017.11

Bézivin, J., Paige, R. F., Aßmann, U., Rumpe, B., & Schmidt, D. C. (2014). Manifesto - Model Engineering for Complex Systems. *CoRR*, *abs/1409.6591*. Retrieved from http://arxiv.org/abs/1409.6591

Bill, R., Mazak, A., Wimmer, M., & Vogel-Heuser, B. (2017). On the Need for Temporal Model Repositories. In *Software technologies: Applications and foundations - STAF 2017 collocated workshops, revised selected papers* (Vol. 10748, pp. 136–145). Springer. Retrieved from https://doi.org/10.1007/978-3-319-74730-9_11 doi: 10.1007/978-3-319-74730-9_11

Blair, G. S., Bencomo, N., & France, R. B. (2009). Models@run.time. *IEEE Computer*, *42*(10), 22–27. Retrieved from https://doi.org/10.1109/MC.2009.326 doi: 10.1109/MC.2009.326

Böhlen, M. H., Dignös, A., Gamper, J., & Jensen, C. S. (2018). Database Technology for Processing Temporal Data. In *Proc. of the 25th international symposium on temporal representation and reasoning, TIME 2018* (Vol. 120, pp. 2:1–2:7). Schloss Dagstuhl - Leibniz-Zentrum für Informatik. Retrieved from https://doi.org/10.4230/LIPIcs.TIME.2018.2 doi: 10.4230/LIPIcs.TIME.2018.2

Bousse, E., Mayerhofer, T., Combemale, B., & Baudry, B. (2019). Advanced and efficient execution trace management for executable domain-specific modeling languages. *Software and Systems Modeling*, *18*(1), 385–421. Retrieved from https://doi.org/10.1007/s10270-017-0598-5 doi: 10.1007/s10270-017-0598-5

Brambilla, M., Cabot, J., & Wimmer, M. (2017). *Model-Driven Software Engineering in Practice, Second Edition*. Morgan & Claypool Publishers. Retrieved from https://doi.org/10.2200/S00751ED2V01Y201701SWE004 doi: 10.2200/S00751ED2V01Y201701SWE004

Burgueño, L., Mayerhofer, T., Wimmer, M., & Vallecillo, A. (2019). Specifying quantities in software models. *Inf. Softw. Technol.*, *113*, 82–97. Retrieved from https://doi.org/10.1016/j.infsof.2019.05.006 doi: 10.1016/j.infsof.2019.05.006

Cabot, J., & Gogolla, M. (2012). Object Constraint Language (OCL): A Definitive Guide. In *Formal methods for model-driven engineering - 12th international school on formal methods for the design of computer, communication, and software systems, SFM 2012* (Vol. 7320, pp. 58–90). Springer. Retrieved from https://doi.org/10.1007/978-3-642-30982-3_3 doi: 10.1007/978-3-642-30982-3_3

Cabot, J., Mazón, J., Pardillo, J., & Trujillo, J. (2010). Specifying Aggregation Functions in Multidimensional Models with OCL. In *Proc. of the 29th international conference on conceptual modeling - ER 2010* (Vol. 6412, pp. 419–432). Springer. Retrieved from https://doi.org/10.1007/978-3-642-16373-9_30 doi: 10.1007/978-3-642-16373-9_30

Cabot, J., Olivé, A., & Teniente, E. (2003). Representing Temporal Information in UML. In *Proc. of the 6th international conference on modeling languages and applications, «uml» 2003 - the unified modeling language* (Vol. 2863, pp. 44–59). Springer. Retrieved from https://doi.org/10.1007/978-3-540-45221-8_5 doi: 10.1007/978-3-540-45221-8_5

Cruz, J. G., Sadovykh, A., Truscan, D., Brunelière, H., Pierini, P., & Muniz, L. L. (2020). MegaM@Rt2 EU Project: Open Source Tools for Mega-Modelling at Runtime of CPSs. In *Proc. of the 16th IFIP WG 2.13 international conference on open source systems, OSS 2020* (Vol. 582, pp. 183–189). Springer. Retrieved from https://doi.org/10.1007/978-3-030-47240-5_18 doi: 10.1007/978-3-030-47240-5_18

Cruz, J. G., Sadovykh, A., Truscan, D., Bruneliere, H., Pierini, P., & Muñiz, L. L. (2020). MegaM@Rt2 EU Project: Open Source Tools for Mega-Modelling at Runtime of CPSs. In *Open source systems* (pp. 183–189). Springer.

David, O., Lloyd, W., II, J. C. A., Green, T. R., Olson, K., Leavesley, G. H., & Carlson, J. (2012). Domain Specific Languages for Modeling and Simulation: Use Case OMS3. In *Proc. of the international congress on environmental modelling and software managing resources of a limited planet* (p. 1201-1207).

García-Domínguez, A., Bencomo, N., & Paucar, L. H. G. (2018). Reflecting on the past and the present with temporal graph-based models. In *Proc. of MODELS 2018 workshops:*

Modcomp, mrt, ocl, flexmde, exe, commitmde, mdetools, gemoc, morse, mde4iot, mdebug, modevva, me, multi, hufamo, ammore, PAINS co-located with ACM/IEEE 21st international conference on model driven engineering languages and systems (MODELS 2018) (Vol. 2245, pp. 46–55). CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2245/mrt_paper_1.pdf

García-Domínguez, A., Bencomo, N., Ullauri, J. M. P., & Paucar, L. H. G. (2019). Querying and Annotating Model Histories with Time-Aware Patterns. In *Proc. of the 22nd ACM/IEEE international conference on model driven engineering languages and systems, MODELS 2019* (pp. 194–204). IEEE. Retrieved from https://doi.org/10.1109/MODELS.2019.000-2 doi: 10.1109/MODELS.2019.000-2

Gogolla, M. (2005). Tales of ER and RE Syntax and Semantics. In *Transformation techniques in software engineering* (Vol. 05161). Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. Retrieved from http://drops.dagstuhl.de/opus/volltexte/2006/425

Gogolla, M., Desai, N., & Doan, K. (2019). Developing User and Recording Interfaces for Design Time and Runtime Models. In *STAF 2019 co-located events joint proceedings: 1st junior researcher community event, 2nd international workshop on model-driven engineering for design-runtime interaction in complex systems, and 1st research project showcase workshop co-located with software technologies: Applications and foundations (STAF 2019)* (Vol. 2405, pp. 39–48). CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2405/08_paper.pdf

Gogolla, M., Hamann, L., Hilken, F., Kuhlmann, M., & France, R. B. (2014). From application models to filmstrip models: An approach to automatic validation of model dynamics. In *Tagungsband der modellierung 2014* (Vol. P-225, pp. 273–288). GI. Retrieved from https://dl.gi.de/20.500.12116/17056

Gómez, A., Cabot, J., & Wimmer, M. (2018). TemporalEMF: A Temporal Metamodeling Framework. In *Proc. of the 37th international conference on conceptual modeling, ER 2018* (Vol. 11157, pp. 365–381). Springer. Retrieved from https://doi.org/10.1007/978-3-030-00847-5_26 doi: 10.1007/978-3-030-00847-5_26

Gregersen, H., & Jensen, C. S. (1999). Temporal Entity-Relationship Models - A Survey. *IEEE Trans. Knowl. Data Eng.*, *11*(3), 464–497. Retrieved from https://doi.org/10.1109/69.774104 doi: 10.1109/69.774104

Haeusler, M., Trojer, T., Kessler, J., Farwick, M., Nowakowski, E., & Breu, R. (2019). Chronosphere: a graph-based EMF model repository for IT landscape models. *Software and Systems Modeling*, *18*(6), 3487–3526. Retrieved from https://doi.org/10.1007/s10270-019-00725-0 doi: 10.1007/s10270-019-00725-0

Hartmann, T., Fouquet, F., Nain, G., Morin, B., Klein, J., Barais, O., & Traon, Y. L. (2014). A Native Versioning Concept to Support Historized Models at Runtime. In *Proc. of the 17th international conference on model-driven engineering languages and systems, MODELS 2014* (Vol. 8767, pp. 252–268). Springer. Retrieved from https://doi.org/10.1007/978-3-319-11653-2_16 doi: 10.1007/978-3-319-11653-2_16

Jensen, S. K., Pedersen, T. B., & Thomsen, C. (2017). Time series management systems: A survey. *IEEE Trans. Knowl. Data Eng.*, *29*(11), 2581–2600. Retrieved from https://doi.org/10.1109/TKDE.2017.2740932 doi: 10.1109/TKDE.2017.2740932

Jensen, S. K., Pedersen, T. B., & Thomsen, C. (2019). Scalable Model-Based Management of Correlated Dimensional Time Series in ModelarDB. *CoRR*, *abs/1903.10269*. Retrieved from http://arxiv.org/abs/1903.10269

Kästner, A., Gogolla, M., Doan, K., & Desai, N. (2018). Sketching a Model-Based Technique for Integrated Design and Run Time Description - Short Paper - Tool Demonstration. In *Software technologies: Applications and foundations - STAF 2018 collocated workshops, revised selected papers* (Vol. 11176, pp. 529–535). Springer. Retrieved from https://doi.org/10.1007/978-3-030-04771-9_39 doi: 10.1007/978-3-030-04771-9_39

Kholod, I., Efimova, M., Rukavitsyn, A., & Shorov, A. (2017). Time Series Distributed Analysis in IoT with ETL and Data Mining Technologies. In *Proc. of the 17th international conference on internet of things, smart spaces, and next generation networks and systems* (Vol. 10531, pp. 97–108). Springer. Retrieved from https://doi.org/10.1007/978-3-319-67380-6_9 doi: 10.1007/978-3-319-67380-6_9

Mazak, A., Lüder, A., Wolny, S., Wimmer, M., Winkler, D., Kirchheim, K., ... Biffl, S. (2018). Model-based generation of run-time data collection systems exploiting AutomationML. *Automatisierungstechnik*, *66*(10), 819–833. Retrieved from https://doi.org/10.1515/auto-2018-0022 doi: 10.1515/auto-2018-0022

Mazak, A., & Wimmer, M. (2016). Towards Liquid Models: An Evolutionary Modeling Approach. In *Proc. of the 18th IEEE conference on business informatics, CBI 2016* (pp. 104–112). IEEE. Retrieved from https://doi.org/10.1109/CBI.2016.20 doi: 10.1109/CBI.2016.20

Meyers, B., Deshayes, R., Lucio, L., Syriani, E., Vangheluwe, H., & Wimmer, M. (2014). ProMoBox: A Framework for Generating Domain-Specific Property Languages. In *Proc. of the 7th international conference on software language engineering, SLE 2014* (Vol. 8706, pp. 1–20). Springer. Retrieved from https://doi.org/10.1007/978-3-319-11245-9_1 doi: 10.1007/978-3-319-11245-9_1

Mouline, L., Benelallam, A., Fouquet, F., Bourcier, J., & Barais, O. (2018). A Temporal Model for Interactive Diagnosis of Adaptive Systems. In *Proc. of the IEEE international conference on autonomic computing, ICAC 2018* (pp. 175–180). IEEE Computer Society. Retrieved from https://doi.org/10.1109/ICAC.2018.00029 doi: 10.1109/ICAC.2018.00029

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. Retrieved from https://doi.org/10.1007/s10664-008-9102-8 doi: 10.1007/s10664-008-9102-8

Schmidt, D., Dittrich, A. K., Dreyer, W., & Marti, R. W. (1995). Time Series, A Neglected Issue in Temporal Database Research? In *Proc. of the international workshop on temporal*

databases, recent advances in temporal databases (pp. 214–232). Springer. Retrieved from https://doi.org/10.1007/978-1-4471-3033-8_12 doi: 10.1007/978-1-4471-3033-8_12

Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with s, 4th edition*. Springer. Retrieved from http://www.worldcat.org/oclc/49312402

Wolny, S., Mazak, A., Carpella, C., Geist, V., & Wimmer, M. (2020). Thirteen years of SysML: a systematic mapping study. *Software and Systems Modeling*, *19*(1), 111–169. Retrieved from https://doi.org/10.1007/s10270-019-00735-y doi: 10.1007/s10270-019-00735-y

Wolny, S., Mazak, A., & Wimmer, M. (2019). Automatic Reverse Engineering of Interaction Models from System Logs. In *Proc. of the 24th IEEE international conference on emerging technologies and factory automation, ETFA 2019* (pp. 57–64). IEEE. Retrieved from https://doi.org/10.1109/ETFA.2019.8869502 doi: 10.1109/ETFA.2019.8869502

Wolny, S., Mazak, A., Wimmer, M., Konlechner, R., & Kappel, G. (2018). Model-Driven Time-Series Analytics. *Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model.*, *13*(Special), 252–261. Retrieved from https://doi.org/10.18417/emisa.si.hcm.19 doi: 10.18417/emisa.si.hcm.19

Ziemann, P., & Gogolla, M. (2003). OCL Extended with Temporal Logic. In *Proc. of the 5th international andrei ershov memorial conference on perspectives of systems informatics, PSI 2003, revised papers* (Vol. 2890, pp. 351–357). Springer. Retrieved from https://doi.org/10.1007/978-3-540-39866-0_35 doi: 10.1007/978-3-540-39866-0_35

A. Single Property to TSDB Mapping Table

Model	TS Profile	T			
		measurement	tag key	tag value	fi
Slot	Temporal (precsion: TimeUnit)	attribute.name	“object”	object.id	“
	Tag	-	attribute. name	slot.value	-
Derived Slot	DerivedRTProperty (query: TS-OCL)	-	-	-	-
Link	Temporal (precsion: TimeUnit)	reference.name	“object”	object.id	re op na
	Tag	-	reference. name	target.id	-
Derived Link	DerivedRTProperty (query: TS-OCL)	-	-	-	-
Method	Log	method.name	“object”	object.id	“

B. Complete Object to TSDB Mapping Table

Model	TS Profile				
		measurement	tag key	tag value	fie
Object	Temporal (precsion: TimeUnit)	object.id	-	-	for att for ref
Slot	Tag	-	attribute. name	slot.value	-
Derived Slot	DerivedRTProperty (query: TS-OCL)	-	-	-	-
Link	Tag	-	reference. name	target.id	-
Derived Link	DerivedRTProperty (query: TS-OCL)	-	-	-	-
Method	Log	method.name	“object”	object.id	“v

About the authors

Alexandra Mazak is a Professor at Montanuniversität Leoben in the research unit Subsurface Engineering. Before she headed Module 3 in the Christian Doppler Laboratory for Model-Integrated Smart Production (CDL-MINT) at JKU Linz. Her research field focuses on digital transformation in tunnel information modeling as well as in modeling of production systems. Her research interests are on information integration, model-driven engineering and model-based data analytics. She headed numerous interdisciplinary national funded projects in the field of digital transformation. You can contact the author at alexandra.mazak-huemer@unileoben.ac.at or visit https://www.tunnellinghub.at/.

Sabine Wolny is currently working as PhD student at the JKU Linz in the module Reactive Model Repositories of the CDL-MINT and in the Research Unit of Building Physics at TU Wien with focus on project management and software development. Her research interests include SysML-based modeling, execution/simulation of production systems, reverse engineering, data integration and back-propagation of runtime information. You can contact the author at sabine.wolny@jku.at or visit https://www.se.jku.at/sabine-wolny/.

Abel Gómez is a senior researcher of the Internet Interdisciplinary Institute, and an Assistant Professor at the Faculty of Computer Science, Multimedia and Telecommunications, both belonging to the Universitat Oberta de Catalunya in Spain. His research interests fall in the broad field of model-driven engineering (MDE), and his research lines have evolved in two complementary directions: the development of core technologies to support MDE activities, and the application of MDE techniques to solve Software Engineering problems. You can contact the author at agomezlla@uoc.edu or visit https://abel.gomez.llana.me.

Jordi Cabot is currently an ICREA Research Professor with the Internet Interdisciplinary Institute. His research interests include software and systems modeling, formal verification, and the role of AI can play in software development (and vice versa). He has published over 200 peer-reviewed conference and journal articles on these topics. Apart from his scientific publications, he writes and blogs about all these topics in several sites. You can contact the author at jordi.cabot@icrea.cat or visit https://jordicabot.com/.

Manuel Wimmer is Full Professor leading the Institute of Business Informatics - Software Engineering at the Johannes Kepler University Linz and he is the head of the Christian Doppler Laboratory CDL-MINT. His research interests comprise foundations of model engineering techniques as well as their application in domains such as tool interoperability, legacy modeling tool modernization, model versioning and evolution, and industrial engineering. You can contact the author at manuel.wimmer@jku.at or visit https://www.se.jku.at/manuel-wimmer/.

Gerti Kappel is Full Professor at the Institute of Information Systems Engineering at TU Wien, chairing the Business Informatics Group. Since the beginning of 2020 she acts as the dean of the Faculty of Informatics at TU Wien. Her current research interests include Model Engineering, Web Engineering, and Process Engineering, with a special emphasis on cyber-physical production systems. You can contact the author at kappel@big.tuwien.ac.at or visit https://www.big.tuwien.ac.at/people/gkappel/.