Reasoning at Runtime using time-distorted Contexts: A Models@run.time based Approach

Thomas Hartmann*, Francois Fouquet*, Gregory Nain*, Brice Morin‡, Jacques Klein* and Yves Le Traon*
*Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg, first.last@uni.lu
‡SINTEF ICT Norway, Norway, first.last@sintef.no

***Abstract*—Intelligent systems continuously analyze their context to autonomously take corrective actions. Building a proper knowledge representation of the context is the key to take adequate actions. This requires numerous and complex data models, for example formalized as ontologies or meta-models. As these systems evolve in a dynamic context, reasoning processes typically need to analyze and compare the current context with its history. A common approach consists in a temporal discretization, which regularly samples the context (snapshots) at specific timestamps to keep track of the history. Reasoning processes would then need to mine a huge amount of data, extract a relevant view, and finally analyze it. This would require lots of computational power and be time-consuming, conflicting with the near real-time response time requirements of intelligent systems. This paper introduces a novel temporal modeling approach together with a time-relative navigation between context concepts to overcome this limitation. Similarly to time distortion theory, our approach enables building time-distorted views of a context, composed by elements coming from different times, which speeds up the reasoning. We demonstrate the efficiency of our approach with a smart grid load prediction reasoning engine.**

***Keywords*—Temporal data, Time-aware context modeling, Knowledge representation, Reactive systems, Intelligent systems**

I. Introduction

An intelligent system needs to analyze both its surrounding environment and its internal state, which together we refer to as the *context* of a system, in order to continuously adapt itself to varying conditions. Therefore, building an appropriate context representation, which reflects the current context of a system is of key importance. This task is not trivial [1] and different approaches and languages are currently used to build such context representations, *e.g.* ontologies [2] or DSLs [3]. Most approaches describe a context using a set of concepts (also called classes or elements), attributes (or properties), and the relations between them. Recently, in the domain of model-driven engineering, the paradigm of models@run.time [4], [5] has rapidly proved its suitability to safely represent, reason about and dynamically adapt a running software system. Nevertheless, context representations (or models), as abstractions of real system states and environments, are only able to reflect a snapshot of a real system at a specific timestamp. However, context data of intelligent systems rapidly change and evolve over time (at different paces for each element) and reasoning processes not only need to analyze the current snapshot of their contexts but also historical data.

The research leading to this publication is supported by the National Research Fund Luxembourg (grant 6816126) and Creos Luxembourg S.A. under the SnT-Creos partnership program.

Let us take a smart grid as an example. Due to changes in the production/consumption chain over time, or to the sporadic availability of natural resources (heavy rain or wind), the properties of the smart grid must be continuously monitored and adapted to regulate the electric load in order to positively impact costs and/or eco-friendliness. For instance, predicting the electric load for a particular region requires a good understanding of the past electricity production and consumption in this region, as well as other data coming from the current context (such as current and forecast weather). Since the electrical grid cannot maintain an overload for more than a few seconds or minutes [6], it is important that protection mechanisms work in this time range. This is what we call *near real-time*.

It is a common approach for such systems to regularly sample and store the context of a system at a very high rate in order to provide reasoning algorithms with historical data. Fig. 1 shows a context —represented as a graph (inspired by object graphs)— sampled at three different timestamps, t_i, t_{i+1}, and t_{i+2}. Each graph in the figure represents the context at a given point in time, where all context variables, independently from their actual values, belong to the same time. Therefore, each graph lies in a horizontal plane (in time).

This systematic, regular context sampling, however, yields to a vast amount of data and redundancy, which is very difficult to analyze and process efficiently. Moreover, it is usually not sufficient to consider and reason just with data from one timestamp, *e.g.* t_i or t_{i+1}. Instead, for many reasoning processes, *e.g.* to investigate a potential causality between two phenomena, it is necessary to simultaneously consider and correlate data from different timestamps (*i.e.* t_i and t_{i+1}). Reasoning processes therefore need to mine a huge amount of data, extract a relevant view (containing context elements from different snapshots), and analyze this view. This overall process would require some heavy resources and/or be time-consuming, conflicting with the near real-time response time requirements such systems usually need to meet.

Going back to the smart grid reasoning engine example: In order to predict the electric load for a region, a linear regression of the average electric load values of the meters in this region, over a certain period of time, has to be computed. Therefore, reasoning processes would need to mine all context snapshots in this time period, extract the relevant meters and electric load values, and then compute a value.

To address these issues, we propose to make context models aware of time *i.e.* to allow context elements (data) from different timestamps in the same model. We refer to such contexts as *time-distorted* contexts. Fig. 2 shows such a context

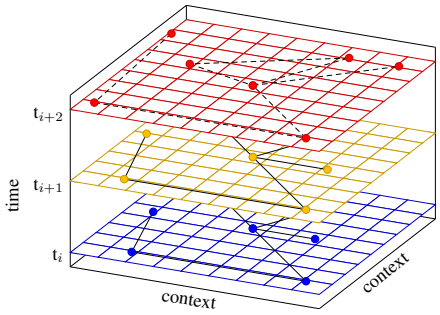

Fig. 1. Linear sampled context

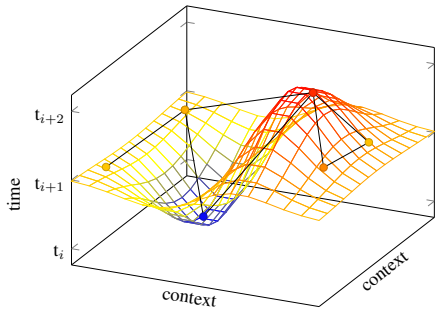

Fig. 2. Time-distorted context

representation, again represented as a graph. Here, the context variables —again independently from their actual values— belong to different timestamps. Such a context can no longer be represented as a graph lying entirely in one horizontal plane (in time). Instead, graphs representing time-distorted contexts lie in a curved plane. They can be considered as specialized *views*, dedicated for a specific reasoning task, composing navigable contexts to reach elements from different times. In contrast to the usage of the term view in database communities we do not aggregate data but offer a way to efficiently traverse specific time windows of a context.
Physics, and especially the study of laser [7], relies on a time distortion [8] property, specifying that the current time is different depending on the point of observation. Applied to our case, this means that context elements can have different values depending on the origin of the navigation context, *i.e.* depending on the timestamp of the inquiring actor. **We claim that time-distorted context representations can efficiently empower continuous reasoning processes and can outperform traditional full sampling approaches by far. The contribution of this paper is to consider temporal information as a first-class property crosscutting any context element, allowing to organize context representations as time-distorted views dedicated for reasoning processes, rather than a mere stack of snapshots. We argue that this approach enables many reasoning processes to react in near real-time (the range of milliseconds to seconds).**

The remainder of this paper is as follows. Section II introduces the background of this work. Section III describes the concepts of our approach and section IV the details on how we implement and integrate these into the open source modeling framework KMF. The provided API is presented in section V. We evaluate our general approach in section VI on a concrete smart grid reasoning engine for electric load prediction. After a discussion about the approach and related work in section VII the conclusion of the paper is presented in section VIII.

II. Background

Over time different languages, formalisms, and concepts to build and represent the context of intelligent systems have been developed and used [1], [9], [10] for different purposes. Entity-relationship models [11], as a general modeling concept for describing entities and the relationships between them, are widely used for building context representations. Ontologies, RDF [12], and OWL [13] are particularly used in the domain of the Semantic Web. These allow to describe facts in a *subject-predicate-object* manner and provide means to reason about these facts. Over the past few years, an emerging paradigm called *models@run.time* [4], [5] proposes to use models both at design and runtime in order to support intelligent systems. At design time, following the model-driven engineering (MDE) paradigm [14], models support the design and implementation of the system. The same (or similar) models are then embedded at runtime in order to support the reasoning processes of intelligent systems, as models offer a *simpler, safer and cheaper* [15] means to reason. Most of these approaches (RDF, OWL, models) have in common that they describe a context using a set of concepts (also called: classes, types, elements), attributes (or properties), and the relations between them. We refer to the representation of a context (set of described elements) as a *context model* or simply as *model* and to a single element (concept) as *model element* or simply as *element*. The concepts of our approach are, in principle, independent of a concrete context representation strategy. However, the implementation of our approach and the provided API are build on a models@run.time based context representation approach and are integrated into an open source modeling framework, called Kevoree Modeling Framework [16] (KMF [1]). KMF is the modeling pillar supporting the Kevoree models@run.time platform [17]. We decided to leverage a models@run.time based approach for several reasons: First of all, models provide a semantically rich way to define a context. Second, models can be used to formally define reasoning activities. Last but not least, the models@run.time paradigm has been proved to be suitable to represent this context during runtime [4], [5].

KMF is an alternative to EMF [18] and specifically designed to support the models@run.time paradigm in terms of memory usage and runtime performance. Two properties of KMF are particularly important for the implementation of our approach: First, in KMF, each model element can be accessed within the model by a path (from the root element of a model to a specific element following containment [19] references), which defines the semantic to efficiently navigate in the model. Our contribution extends the path of model elements with temporal data in order to provide seamless navigation, not only in the model but as well in time. Second, in KMF each model element can be serialized independently, using paths to represent elements of its relationships. We use this property in our implementation to incrementally store model element modifications.

[1] http://kevoree.org/kmf

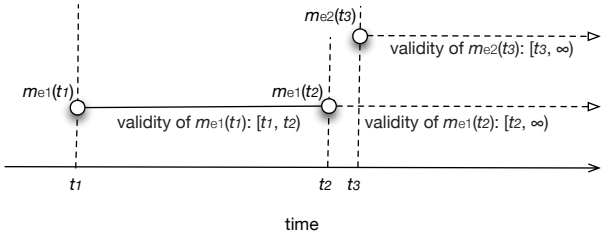

Fig. 3. Continous validity of model elements

III. Enabling time-distorted Contexts

Our hypothesis is that temporal knowledge is part of a domain itself (*e.g.* electric load or wave propagation prediction, medical recommender systems, financial applications) and that defining and navigating temporal data directly within domain contexts is far more efficient and convenient than regularly sampling a context and independently querying each model element with the appropriate time. Therefore, we provide a concept to define and navigate into the time dimension of contexts. Most importantly, we enable a context representation to seamlessly combine elements from different points in time, forming a *time-distorted* context, which is especially useful for time related reasoning. We claim that our approach, which weaves time directly into the context model, as opposed to a full sampling and classic data mining approach, is compatible with near real-time requirements. In the following we present the concepts to enable time-distorted context models.

A. Temporal Validity for Model Elements

Instead of relying on context snapshots, we define a context as a continuous structure. Nevertheless, each context (model) element of this structure can evolve independently. We first define an implicit *validity* for model elements. Therefore, we associate a timestamp to each model element. They reflect a domain-dependent notion of the time at which a model element was captured and can be accessed on the element itself. In other words, a timestamp defines a *version* $v_{m_e}(t)$ of a model element m_e at a time t. If a model element now evolves, an additional version of the same element is created and associated to a new timestamp (the domain time at which the new version is captured). Timestamps can be compared and thus form a chronological sequence. Therefore, although timestamps are discrete values, they logically define intervals in which a model element can be considered as *valid* with regards to a timestamp. A model element is valid from the moment it is captured until a new version is captured. New versions are only created if necessary, *i.e.* if the model element changed. Fig. 3 shows two model elements, m_{e1} with two versions and m_{e2} with one version, and their corresponding validity periods. As represented in the figure, version $m_{e1}(t_1)$ is valid in interval $[t_1, t_2[$. Since there is no version of model element m_{e1}, which is captured later than t_2, the validity of version $m_{e1}(t_2)$ is the open interval $[t_2, +\infty[$. Accordingly, version $m_{e2}(t_3)$ is valid in $[t_3, +\infty[$. Since model elements now have a temporal validity, a relationship r from a model element m_{e1} to m_{e2} is no longer uniquely defined. Instead, the timestamps of model elements have to be taken into account for the resolution of relationships (described in III-C). The association of each model element with a timestamp and thus a validity, provides the foundation for a continuous context representation. Although model elements are still sampled at discrete timestamps, the definition of a continuous validity for each model element allows to represent a context as a continuous structure. This structure provides the foundation for our time-distorted context models.

B. Navigating through Time

Based on the idea that it is necessary for intelligent systems to consider not only the current context but also historical data to correlate or investigate potential causalities between two phenomena, we provide means to enable an efficient navigation into time. Therefore, we define three basic operations for model elements. These can be called on each model element:
The *shift* operation is the main possibility to navigate a model element through time. It takes a timestamp as parameter, looks for the model version of itself, which is valid at the required timestamp, loads the corresponding element from storage (can be RAM) and returns the loaded version.
The *previous* operation is a shortcut to retrieve the direct predecessor (in terms of time) of the current model element.
The *next* method is similar to the *previous* operation but retrieves the direct successor of the current model element.
These operations allow us to shift model elements independently from each other through time. This makes it possible to create context models, combining model elements from different points in time. Now only the last step is missing to have time-distorted context models for efficient runtime reasoning processes: A concept to take the time of model elements into account when navigating between model elements.

C. Time-relative Navigation

Navigating temporal data is a complex task, since a relationship r from an element m_{e1} to an element m_{e2} is no longer uniquely defined. Instead —depending on the timestamps t_1 and t_2 of m_{e1} and m_{e2}, and depending on the set of versions of m_{e1} and m_{e2}— a relationship r from m_{e1} to m_{e2} can link different versions of m_{e2}. This means which version of m_{e2} is related to m_{e1} by r depends on the timestamp t of m_{e1}. Processing this time-relative navigation manually, *e.g.* to correlate or investigate potential causalities between two phenomena, is complicated and error-prone. We therefore provide a concept to automatically resolve relationships, taking the time aspect into account, while navigating the context model. This time related resolution is completely transparent and hidden behind methods to navigate in the context model. Hereby, a context time can be defined (the curve in fig. 2) and each model element is then resolved accordingly to this definition while traversing the model. For example, the context time can be defined as the current time of a model element minus one day. When navigating from model element m_{e1} at timestamp t_i to element m_{e2}, the version of m_{e2}, which is valid at timestamp $t_i - 1$ *day* is resolved. In case that at timestamp t_i, object m_{e2} does not exist, the *prior existing version of* m_{e2} is returned. Considering model elements in the context of a specific time interval creates a navigable time dimension for model elements. This time relative data resolution is one of the novel concepts of this contribution. Unlike in previous approaches (*e.g.* relationships in MOF [19] or predicates in RDF [12]), the navigation function is not constant but yields

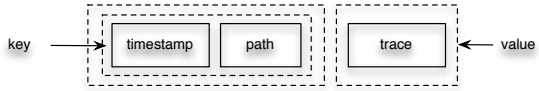

Fig. 4. Key/value structure for time-relative storage

different results depending on the navigation context (*i.e.* the current observation date). This distortion in terms of navigable relations finally enables what we call a time-distorted context.

IV. Integration into KMF

In this section we describe how we integrate our time-distorted modeling approach into the open source modeling framework KMF. We rely on two properties to integrate the time dimension as a crosscutting concern into model elements: *i*) each model element must be uniquely identifiable, and *ii*) it must be possible to get a serialized representation of all attributes and relationships of each model element, with no relativity to a time. To ensure the first property, KMF defines a *path* for each model element, starting from the root element of a model to the element following the containment relationships, as a unique identifier. Since the containment graph is actually a tree (each element, except the root, has to be contained exactly once), the path is a unique full qualified identifier. For the second property, KMF serializes models and model elements into *traces*. A trace defines a sequence of atomic actions to construct a model element, using the path concept (including relationship information). Each model element can be transformed into a trace and vice versa [20], [21]. As stated in III-A we inject a timestamp into all model elements. We do that by extending the KMF generator to automatically generate a timestamp attribute for all model elements. As a consequence, a model element, or more precisely a version of a model element, is now no longer simply defined by its path alone, but by a combination of its path and timestamp. Using the path together with a timestamp as key and the trace as value allows us to store and retrieve model elements within their time dimension in a simple *key/value* manner. The resolution (and storage) of a model element is therefore always relative to the context time. This is shown in fig. 4.

This context data organization allows us to use technologies eligible for big data to efficiently store and load context data. The data can be stored using different back ends, *e.g.* key/value stores, relational databases, or simply in memory (as a cache). In our implementation we use Google LevelDB[2] since it has proven to be suitable for handling big context data and, most importantly, it is very fast for our purpose (see VI). The storage implementation itself, however, is not part of our contribution. We intend to provide an efficient layer for runtime reasoning processes, on top of already existing storage technologies.

Since data captured in context models usually evolve at a very high pace (*e.g.* milliseconds or seconds), and our approach foresees to not only store the current version of model elements but also historical ones, context models can quickly become very large. In such cases, context models may no longer fit completely into memory, or at least it is no longer practical to do so. Therefore, based on our storage concept and the uniqueness of KMF paths in a model, we implement a *lazy loading*[3] mechanism to enable efficient loading of big context models. We use *proxies*[3], containing only path and timestamp, to reduce the overall memory usage. Attributes and referenced elements are only loaded when they are read or written. To enable this we extend KMF so that, while the context model is traversed, relationships are dynamically resolved. First, it must be determined which version of a related model element must be retrieved. This depends on the timestamp of the model element version (and the context time) from which the navigation starts (discussed in III-C). Second, the actual model element version must be loaded from storage. Our implementation allows to manage context models of arbitrary sizes efficiently and it hides the complexity of resolving —and navigating— temporal data behind an API.

V. API

Modeling approaches use meta-model definitions (*i.e.* concept descriptions) to generate domain specific APIs. The following section illustrates our API on a simplified smart grid meta-model definition. The model consists of a smart meter with an attribute for electric load and a relationship to reachable surrounding meters. In addition to a classical modeling API, our time-distorted context extension provides functions for creating, deleting, storing, loading, and shifting versions of model elements. Applications can use this API to create new model elements, specify their timestamps, store them, change their attributes and relationships, and store new versions (with different timestamps). In addition, the API can be used to specify the *context time* on which elements should be resolved while traversing the model. One can imagine the definition of the context time as the curve shown in fig. 2. Listing 1 shows Java code that uses a *Context ctx* (abstraction to manipulate model elements) to perform these actions.

```
// creating and manipulating model elements
m1 = ctx.createSmartMeter("m1","2014/3/1");
m1.setElectricLoad(125.6);
m1.addReachables(ctx.load("m2"));
m1_2 = m1.shift("2014/3/2");
m1_2.setElectricLoad(193.7);
// definition of the context time
ctx.timeDef("m1","2014/3/1");
ctx.timeDef("m2","2014/3/2");
r_m1 = ctx.load("m1");
assert(r_m1.getElectricLoad()==125.6);
r_m2 = r_m1.getReachables().get(0);
assert(r_m2.getTime()=="2014/3/2")
```

Listing 1. Usage of the time-distorted modeling API

The API provides a seamless way to create, manipulate, and navigate in time-distorted context representations.

VI. Evaluation

To quantify the advantages of our approach, we evaluate it on a smart grid reasoning engine to predict the electric load in a region based on current load and historical data. This problem is taken from our cooperation with Creos Luxembourg S.A. and led to the research behind this approach.

[2]https://code.google.com/p/leveldb/

[3]http://wiki.eclipse.org/CDO

TABLE I. BENCHMARK USING FULL SAMPLING

Scenario	Reasoning	Insert
Small Deep Prediction (SDP)	1075.6 ms	267 s
Small Wide Prediction (SWP)	1088.4 ms	
Large Deep Prediction (LDP)	180109.0 ms	
Large Wide Prediction (LWP)	181596.1 ms	

TABLE II. BENCHMARK USING TIME-DISTORTED CONTEXTS

Scenario	Reasoning	Insert
Small Deep Prediction (SDP)	1.8 ms	16 s
Small Wide Prediction (SWP)	0.8 ms	
Large Deep Prediction (LDP)	187.0 ms	
Large Wide Prediction (LWP)	157.6 ms	

Smart grids are infrastructures characterized by the introduction of reactive entities modernizing the electricity distribution grid. Smart meters, entities installed at customer sites to continuously measure consumption data and propagate it through network communication links, are one of the main building blocks of smart grids. Based on the electrical consumption smart meters can determine the electrical load. The load is regularly sampled, which leads to big context models. The idea for this reasoning engine is to predict if the load in a certain region will likely exceed or surpass a critical value. Our experimental validation focuses on two key indicators: **(1)** performance of the reasoning process and **(2)** insertion time.

Experimental results: We implemented the reasoning engine case study leveraging our approach on top of the KMF framework, presented in IV. It has been implemented twice, once with a classical systematic sampling strategy, and once using a time-distorted context model. The full sampling approach is implemented using the same data storage (Google LevelDB) as our time-distorted context representation approach. Our context model definition consists of one concept, a *smart meter*, one attribute for the *electrical load*, and a *reachable* relationship, which connects smart meters. The context model under study contains 100 smart meters with 10000 values history each, resulting in one million elements to store and analyze. This amount of data corresponds to roughly 140 days history. The experimental validation consists of an electrical load prediction for a specific point of the grid. We define two kinds of predictions both based on a linear regression: 1) *deep* uses a deep history (in time) of the meter, 2) *wide* uses history and in addition the electric load of surrounding meters. These two strategies are each executed with two different ranges: 1) *small* using ten hours of history (30 time units), 2) *large* uses a history of two months (4800 time units). Using the two strategies and two ranges for each, we create four different test series (SDP, SWP, LDP, LWP) that represent typical use cases for our case study. The experiments were carried out on a MacBook Pro i5 2.4 Ghz, 16 GB RAM. The results using full sampling are presented in table I, the results leveraging time-distorted contexts in table II. As shown in the tables, our time-distorted context strategy leads to a reduction of the reasoning time by factors of: **598** for *SDP*, **1361** for *SWP*, **963** for *LDP*, and **1152** for *LWP*, compared to the classic full sampling strategy. The insert time (for storing the context values) has also been significantly improved by a factor of **17**. The two reasoning strategies (full sampling and time-distorted models) predict the same electric load values for all tests. Our evaluation shows a reasoning time in the order of magnitude of minutes for regular sampling and milliseconds for time-distorted contexts, to analyze a particular section of the grid. Moreover, according to our smart grid case study, this electric load prediction has to be continuously performed on several hundred grid points. Thus, our solution reduces the computation time from hours to a few seconds, compatible with the near real-time requirements of smart grid overload capacity. The huge gains compared to full sampling can be explained by the fact that time-distorted contexts allow to directly navigate across the time dimension without costly mining the necessary data from different context models.

VII. DISCUSSION AND RELATED WORK

The lack of a temporal dimension in data modeling has been discussed in detail, especially in the area of databases. In an early work Clifford *et al.* [22] provide a formal semantic for historical databases. Rose and Segev [23] suggest to extend the entity-relationship data model into a temporal, object-oriented one, incorporating temporal structures and constraints in the data model itself rather than at the application level. They also propose a temporal query language for the model. Ariav [24] suggests a temporally-oriented data model as a restricted superset of the relational model. He adds a temporal aspect to the tabular notion of data and provides a framework and a SQL-like query language for storing and retrieving data, taking their temporal context into account. Works of Mahmood *et al.* [25] and Segev and Shoshani [26] take a similar direction and seek to extend the relational model with temporal aspects. In an earlier work Segev and Shoshani [27] examine semantics of temporal data and corresponding operators independently from a data model. In a more recent work, Shih *et al.* [28] describe how including time in a context model helps triggering and handling exceptions in system processes. Selig *et al.* [29] describe an interesting mathematical approach to examine time-dependent association between variables. In the Bigtable [30] implementation, Google incorporates time at its core by allowing each cell in a Bigtable to contain multiple versions of the same data, associated to different timestamps. All this work addresses mainly efficient storage and querying of time related data but largely ignores handling of time in the application domain itself. However, considering time in reasoning processes, like correlating causalities between phenomena, is complex and time-consuming, conflicting with the strict response time requirements intelligent systems usually face. The need to represent and reason about temporal knowledge has also been discussed in RDF [12], OWL [13] and the Semantic Web. Motik [31] suggests a logic-based approach for representing validity time in RDF and OWL. He also proposes to extend SPARQL to temporal RDF graphs and presents a query evaluation algorithm. We suggest to add a time dimension directly into the knowledge representation of a domain itself, *i.e.* into the core of context models. Therefore, we not only efficiently store and query historical data (what is done in other works before), but we propose a way to use time-distorted data sets specifically for intelligent reasoning. Also, we do not extend a specific data model (*e.g.* the relational data model) with temporal structures but use model-driven engineering techniques to integrate a time dimension as crosscutting property of any model element. We do not rely on a complex query language for retrieving temporal data. Instead, our approach aims at providing a natural, query-less

and seamless navigation into the time dimension of model elements, allowing a composition of different time-related values to build a dedicated context model for reasoning purposes (inspired by temporal logic [32]). Like version control systems, *e.g.* Git[4], our approach stores incremental changes (over time) rather than snapshots of a complete system.
Our approach is especially useful if model elements evolve at different paces. If all elements of a context evolve at the same pace the main advantage is the navigation concept as well as the lazy loading mechanism. In future work we want to improve and optimize our implementation. Especially the insertion of new model element versions between two existing versions has to be improved. In addition, we want to investigate how to distribute storage across multiple machines and how this affects the performance of our approach.

VIII. Conclusion

Considering time as a crosscutting concern of data modeling has been discussed since more than two decades. However, recent data modeling approaches mostly still rely on a discrete time representation, which can hardly consider model elements (*e.g.* context variables) coming from different points in time. In this paper, we presented a novel approach which considers time as a first-class property crosscutting any model element, allowing to organize context representations as time-distorted views dedicated for reasoning processes, rather than a mere stack of snapshots. By introducing a temporal validity independently for each model element we allowed model elements to evolve independently and at different paces, making the full sampling of a context model unnecessary. Instead of introducing a dedicated querying language we provided operations to move model elements independently through time, enabling the creation of context models, which combine model elements from different timestamps. Finally, we added a time-relative navigation, which makes an efficient navigation between model elements, coming from different timestamps, possible. This allows us to assemble a time-distorted context model for a specific reasoning purpose and seamlessly and efficiently navigate along this time distortion without manual and costly mining of the necessary data from different context models. Our approach has been implemented and integrated into the open source modeling framework KMF and evaluated on a smart grid reasoning engine for electric load prediction. We showed that our approach supports reasoning processes, outperforms a full context sampling by far, and is compatible with most of near real-time requirements.

References

[1] M. Perttunen, J. Riekki, and O. Lassila, "Context representation and reasoning in pervasive computing: a review," *Int. Journal of Multimedia and Ubiquitous Engineering*, pp. 1–28, 2009.

[2] C.-H. Liu, K.-L. Chang, J.-Y. Chen, and S.-C. Hung, "Ontology-based context representation and reasoning using owl and swrl," in *Communication Networks and Services Research Conf. (CNSR), 2010 8th Annu.*, 2010, pp. 215–220.

[3] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling context information in pervasive computing systems," in *Proc. 1st Int. Conf. Pervasive Computing*, ser. Pervasive '02, 2002, pp. 167–180.

[4] G. Blair, N. Bencomo, and R. France, "Models@ run.time," *Computer*, vol. 42, no. 10, pp. 22–27, 2009.

[5] B. Morin, O. Barais, J. Jezequel, F. Fleurey, and A. Solberg, "Models@ run.time to support dynamic adaptation," *Computer*, vol. 42, 2009.

[4] http://git-scm.com/

[6] J. C. Cepeda, D. Ramirez, and D. Colome, "Probabilistic-based overload estimation for real-time smart grid vulnerability assessment," in *Transmission and Distribution: Latin America Conf. and Expo. (T D-LA), 2012 6th IEEE/PES*, 2012, pp. 1–8.

[7] H. W. Tom, G. Aumiller, and C. Brito-Cruz, "Time-resolved study of laser-induced disorder of si surfaces," *Physical review letters*, vol. 60, no. 14, p. 1438, 1988.

[8] P. Hubral, "Time migration-some ray theoretical aspects*," *Geophysical Prospecting*, 1977.

[9] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, no. 4, 2007.

[10] T. Strang and C. L. Popien, "A context modeling survey," in *UbiComp 1st Int. Workshop on Advanced Context Modelling, Reasoning and Management*, 2004, pp. 31–41.

[11] P. P. shan Chen, "The entity-relationship model: Toward a unified view of data," *ACM Trans. Database Syst.*, vol. 1, pp. 9–36, 1976.

[12] O. Lassila and R. R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification," W3C, W3C Recommendation, 1999.

[13] W. W. W. C. W3C, *OWL 2 Web Ontology Language. Structural Specification and Functional-Style Syntax*, Std., 2009.

[14] S. Kent, "Model driven engineering," in *IFM*, 2002.

[15] J. Rothenberg, L. E. Widman, K. A. Loparo, and N. R. Nielsen, "The nature of modeling," in *Artificial Intelligence, Simulation and Modeling*, 1989, pp. 75–92.

[16] F. Fouquet, G. Nain, B. Morin, E. Daubert, O. Barais, N. Plouzeau, and J. Jézéquel, "An eclipse modelling framework alternative to meet the models@runtime requirements." in *MoDELS*, 2012.

[17] F. Fouquet, E. Daubert, N. Plouzeau, O. Barais, J. Bourcier, and J.-M. Jézéquel, "Dissemination of reconfiguration policies on mesh networks, DAIS 2012."

[18] F. Budinsky, D. Steinberg, and R. Ellersick, *Eclipse Modeling Framework : A Developer's Guide*, 2003.

[19] OMG, *OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1*, Object Management Group Std., Rev. 2.4.1, 2011.

[20] X. Blanc, I. Mounier, A. Mougenot, and T. Mens, "Detecting model inconsistency through operation-based model construction," in *Proc. 30th Int. Conf. Software Engineering*, 2008, pp. 511–520.

[21] J. Klein, J. Kienzle, B. Morin, and J.-M. Jézéquel, "Aspect model unweaving," in *In 12th International Conference on Model Driven Engineering Languages and Systems (MODELS 2009)*, L. 5795, Ed., Denver, Colorado, USA, 2009, pp. p 514–530.

[22] J. Clifford and D. S. Warren, "Formal semantics for time in databases." in *XP2 Workshop*, 1981.

[23] E. Rose and A. Segev, "Toodm - a temporal object-oriented data model with temporal constraints." in *ER*, T. J. Teorey, Ed., 1991.

[24] G. Ariav, "A temporally oriented data model," *ACM Trans. Database Syst.*, vol. 11, no. 4, pp. 499–527, 1986.

[25] N. Mahmood, A. Burney, and K. Ahsan, "A logical temporal relational data model," *CoRR*, 2010.

[26] A. Segev and A. Shoshani, "The representation of a temporal data model in the relational environment." in *SSDBM*, 1988.

[27] ——, "Logical modeling of temporal data," in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, ser. SIGMOD '87, 1987.

[28] C.-H. Shih, N. Wakabayashi, S. Yamamura, and C.-Y. Chen, "A context model with a time-dependent multi-layer exception handling policy," *IJICIC*, vol. 7, no. 5A, pp. 2225–2234, 2011.

[29] J. P. Selig, K. J. Preacher, and T. D. Little, "Modeling time-dependent association in longitudinal data: A lag as moderator approach," *Multivariate Behavioral Research*, vol. 47, no. 5, pp. 697–716, 2012.

[30] F. Chang, J. Dean, S. Ghemawat, W.-C. Hsieh, D.-A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.-E. Gruber, "Bigtable: A distributed storage system for structured data," in *Proc. 7th USENIX Symp. OSDI - Volume 7*, ser. OSDI '06, 2006, pp. 15–15.

[31] B. Motik, "Representing and querying validity time in rdf and owl: A logic-based approach," in *Proc. 9th Int. Semantic Web Conf. The Semantic Web - Volume Part I*, ser. ISWC'10, 2010, pp. 550–565.

[32] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annu. Symp.*, 1977, pp. 46–57.