# A Unified Modeling Framework to Abstract Knowledge of Dynamically Adaptive Systems

Ludovic Mouline

April 5, 2019

# Abstract

**Vision:** As state-of-the-art techniques fail to model efficiently the evolution and the uncertainty existing in dynamically adaptive systems, the adaptation process makes suboptimal decisions. To tackle this challenge, modern modeling frameworks should efficiently encapsulate time and uncertainty as first-class concepts.

*Context* Smart grid approach introduces information and communication technologies into traditional power grid to cope with new challenges of electricity distribution. Among them, one challenge is the resiliency of the grid: how to automatically recover from any incident such as overload? These systems therefore need a deep understanding of the ongoing situation which enables reasoning tasks for healing operations. **Abstraction** is a key technique that provided an illuminating description of systems, their behaviors, and/or their environments alleviating their complexity. **Adaptation** is a cornerstone feature that enables reconfiguration at runtime for optimizing software to the current and/or future situation.

Abstraction technique is pushed to its paramountcy by the model-driven engineering (MDE) methodology. However, information concerning the grid, such as loads, is not always known with absolute confidence. Through the thesis, this lack of confidence about data is referred to as **data uncertainty**. They are approximated from the measured consumption and the grid topology. This topology is inferred from fuse states, which are set by technicians after their services on the grid. As humans are not error-free, the topology is therefore not known with absolute confidence. This data uncertainty is propagated to the load through the computation made. If it is neither present in the model nor not considered by the adaptation process, then the adaptation

process may make suboptimal reconfiguration decision.

The literature refers to systems which provide adaptation capabilities as dynamically adaptive systems (DAS). One challenge in the grid is the phase difference between the monitoring frequency and the time for actions to have measurable effects. Action with no immediate measurable effects are named **delayed action**. On the one hand, an incident should be detected in the next minutes. On the other hand, a reconfiguration action can take up to several hours. For example, when a tree falls on a cable and cuts it during a storm, the grid manager should be noticed in real time. The reconfiguration of the grid, to reconnect as many people as possible before replacing the cable, is done by technicians who need to use their cars to go on the reconfiguration places. In a fully autonomous adaptive system, the reasoning process should be considered the ongoing actions to avoid repeating decisions.

*Problematic* **Data uncertainty and delayed actions are not specific to smart grids.**

First, data are, almost by definition, uncertain and developers always work with estimates. Hardware sensors have by construction a precision that can vary according to the current environment in which they are deployed. A simple example is the temperature sensor that provides a temperature with precision to the nearest degree. Software sensors approximate also values from these physical sensors, which increases the uncertainty. For example, CPU usage is computed counting the cycle used by a program. As stated by Intel, this counter is not error-prone[1].

Second, it always exists a delay between the moment where a suboptimal state is detected by the adaptation process and the moment where the effects of decisions taken are measured. This delayed is due to the time needed by a computer to process data and, eventually, to send orders or data through networks. For example, migrating a virtual machine from a server to another one can take several minutes.

**Through this thesis, I argue that this data uncertainty and this delay cannot be ignored for all dynamic adaptive systems.** To know if the data uncertainty should be considered, stakeholders should wonder **if this data uncertainty**

[1] https://software.intel.com/en-us/itc-user-and-reference-guide-cpu-cycle-counter

**affects the result of their reasoning process, like adaptation**. Regarding delayed action, they should verify **if the frequency of the monitoring stage is lower than the time of action effects to be measurable**. These characteristics are common to smart grids, cloud infrastructure or cyber-physical systems in general.

*Challenge* These problematics come with different challenges concerning the representation of the knowledge for DAS. The global challenge address by this thesis is: **how to represent the uncertain knowledge allowing to efficiently query it and to represent ongoing actions in order to improve adaptation processes?**

*Vision* **This thesis defends the need for a unified modeling framework which includes, despite all traditional elements, temporal and uncertainty as first-class concepts.** Therefore, a developer will be able to abstract information related to the adaptation process, the environment as well as the system itself.

Concerning the adaptation process, the framework should enable abstraction of the actions, their context, their impact, and the specification of this process (requirements and constraints). It should also enable the abstraction of the system environment and its behavior. Finally, the framework should represent the structure, behavior and specification of the system itself as well as the actuators and sensors. All these representations should integrate the data uncertainty existing.

*Contributions* Towards this vision, this document presents two approaches: a temporal context model and a language for uncertain data.

The temporal context model allows abstracting past, ongoing and future actions with their impacts and context. First, a developer can use this model to know what the ongoing actions, with their expect future impacts on the system, are. Second, she/he can navigate through past decisions to understand why they have been made when they have led to a sub-optimal state.

The language, named Ain'tea, integrates data uncertainty as a first-class concept. It allows developers to attach data with a probability distribution which represents their uncertainty. Plus, it mapped all arithmetic and boolean operators to uncertainty propagation operations. And so, developers will automatically propagate the uncertainty

of data without additional effort, compared to an algorithm which manipulates certain data.

*Validation* Each contribution has been evaluated separately. The language has been evaluated through two axes: its ability to detect errors at development time and its expressiveness. Ain'tea can detect errors in the combination of uncertain data earlier than state-of-the-art approaches. The language is also as expressive as current approaches found in the literature. Moreover, we use this language to implement the load approximation of a smart grid furnished by an industrial partner, Creos S.A.[2].

The context model has been evaluated through the performance axis. The dissertation shows that it can be used to represent the Luxembourg smart grid. The model also provides an API which enables the execution of query for diagnosis purpose. In order to show the feasibility of the solution, it has also been applied to the use case provided by the industrial partner.

**Keywords:** dynamically adaptive systems, knowledge representation, model-driven engineering, uncertainty modeling, time modeling

[2]Creos S.A. is the power grid manager of Luxembourg. https://www.creos-net.lu

# Table of Contents

# 1

# Introduction

## Contents



**Abstract:** *Model-driven engineering methodology and dynamically adaptive systems approach are combined to tackle new challenges brought by systems nowadays. After introducing these two software engineering techniques, I give one example of such systems: the Luxembourg smart grid. I will also use this example to highlight two of the problematics: uncertainty of data and delays in actions. Among the different challenges which are implied by them, I present the global one addressed by the vision defended in this thesis: modeling of temporal and uncertain data. This global challenge can be addressed by splitting up in several ones. I present two of them, which are directly tackled by two contributions presented in this thesis.*

## 1.1 Use case: Luxembourg smart grid

Should contain: -

# 2

# TKM: a temporal knowledge model to represent actions, their contexts and their impacts

## Contents




**Abstract:** *The evolving complexity of adaptive systems impairs our ability to deliver anomaly-free solutions. Fixing these systems require a deep understanding on the reasons behind decisions which led to faulty or suboptimal system states. Developers thus need diagnosis support that trace system states to the previous circumstances –targeted requirements, input context– that had resulted in these decisions. However, the lack of efficient temporal representation limits the tracing ability of current approaches. To tackle this problem, we first propose a knowledge formalism to define the concept of a decision. Second, we describe a novel temporal data model to represent, store and query decisions as well as their relationship with the knowledge (context, require- ments, and actions). We validate our approach through a use case based on the smart grid at Luxembourg. We also demonstrate its scalability both in terms of execution time and consumed memory.*

## 2.1 Introduction

should define: decision, action, context, knowledge

## 2.2 Knowledge formalization

### 2.2.1 Formalism

As discussed previously, I consider knowledge to be the association of context information, requirements, and action information, all in one global and unified model. While context information captures the state of the system environment and its surroundings, the system requirements define the constraints that the system should satisfy along the way. The actions, on the other hand, are means to reach the goals of the system.

In this section, I provide a formalization of the knowledge used by adaptation processes based on a temporal graph. Indeed, due to the complexity and interconnectivity of system entities, graph data representation seems to be an appropriate way to represent the knowledge. Augmented with a temporal dimension, temporal graphs are then able to symbolize the evolution of system entities and states over time. We benefit from the well-defined graph manipulation operations, namely temporal graph pattern matching and temporal graph relations to represent the traceability links between the decisions made and their circumstances.

Let $K$ be an adaptive process over a system knowledge. $K$ is defined by the tuple $(C, R, A, D)$, where:

- $C$ is a **directed temporal attributed graph** representing **context information**,
- $R$ is a set of **temporal queries or invariants** over $C$ specifying the **system requirements**,
- $A$ is a set of **temporal queries** determining actions,
- $D$ is a set of **temporal relations** representing decisions, *i.e.,* **tracing the execution of actions and their impact**,

While the source of relations in $D$ represents the state before the execution of an action, the target shows its impact on the context. Its intent is **to trace back impacts of actions execution to the decisions they originated from**.

We consider $T$ an ordered discrete time domain where time points are non-uniformly distributed. This time domain can be divided into 3 different sub-domains $T = T_{past} \cup \{t\} \cup T_{future}$, where:

- $T_{past}$ is the sub-domain $\{t_0;t_1;\ldots;t_{current-1}\}$ representing graph data history starting from $t_0$, the oldest point, until current time, t, excluded.
- {t} is a singleton representing the current time point
- $T_{future}$ is sub-domain $\{t_{current+1};\ldots;t_\infty\}$ representing future time points
- the three domains depend completely on the current time {t} as these domains slide as time passes. At any point in time, these domains never overlap: $T_{past} \cap \{t\} = \emptyset$, $T_{future} \cap \{t\} = \emptyset$, and $T_{past} \cap T_{future} = \emptyset$

We also define right-opened time interval $I \in T \times T$ as $(t_1, t_2)$ where $t_2 - t_1 > 0$. We denote by $start(i)$ the lower bound of a time interval $i$ and $end(i)$ its upper bound.

Context information $\mathcal{C} = (\mathcal{N}, \mathcal{E}, \mathcal{V}^T, \mathcal{Z}^T, \mathcal{P})$ is represented by a temporal nodes set $\mathcal{N}$, and a temporal relationships set $\mathcal{E}$. Nodes have a set of attribute values. An attribute value has a type (numerical, boolean, ...). Every temporal relationship $r \in \mathcal{E}$ can be considered as a couple of nodes $(n_s, n_t) \in \mathcal{N} \times \mathcal{N}$, where $n_s$ is the source node and $n_t$ is the target node. Every node or relationship has a defined validity time interval $\mathcal{V}^T$ in which it exists. In order to track the evolution of the temporal graph elements in time, we refer to the relation $\mathcal{Z}^T$. Finally, $\mathcal{P}$ is a function that returns the values of a node attribute. In the remainder of the paper, relationship and edge are used interchangeably.

The validity relation $\mathcal{V}$ has a very important role, which is augmenting the graph with the temporal dimension. A node is considered invalid either when it is removed or when one of its attributes value changes. In the latter case, a new node with the updated value is created. Whilst, a relationship is considered invalid if either its source node or target node is invalid, or when the relationship itself is removed. The temporal

validity relation is defined as $\mathcal{V}^T : \mathcal{N} \cup \mathcal{E} \rightarrow I$. It takes as a parameter a node or a relationship ($e \in \mathcal{N} \cup \mathcal{E}$) and returns a time interval $i$ during which the graph element is valid.

Furthermore, the temporal graph has the capacity to keep track of data history thanks to the $\mathcal{Z}^T$ relation. It serves to trace the progress[1] of a node $n \in \mathcal{N}$ at any point in time. This relation can also be seen as a temporal identity function which takes as parameters a given node $n$ and a specific time point $t$, and returns the corresponding node at that point. Formally, $\mathcal{Z}^T : \mathcal{N} \times \mathcal{T} \rightarrow \mathcal{N}$.

The set of attributes values of a node $n \in \mathcal{N}$ is denoted as $\mathcal{P}(n) = \{(a_1(n), c_1), \ldots, (a_j(n), c_j)\}$. Every couple represents the value of a given attribute ($(a_1)$ and its corresponding confidence, in percentage. The confidence value corresponds to the accuracy value introduced in the previous section.

Historical data is extracted using a simple projection over the temporal graph $\mathcal{C}(T_{past})$. The current state of the graph is referred to as $\mathcal{C}(t)$. Finally, the future states of the graph are designated by the $\mathcal{C}(T_{future})$. The later set of states includes only graph data that can be predicted or planned, using, for instance, statistical models, machine learning, or computed based on some prior knowledge provided by the users.

We define a temporal graph pattern as a temporal subgraph of $\mathcal{C}$, but with a time limiting constraint coming in the form of a time interval $i$. The time interval can be either fixed (absolute) or sliding (relative). In what follows, we denote by $\mathbb{P}_{[t_j,t_k]}(\mathcal{C})$ a temporal graph pattern, where $t_j$ and $t_k$ are the lower and upper bound of the time interval respectively.

We define the system requirements $\mathcal{R}$ as a set patterns $\mathbb{P}_{[t_j,t_k]}(\mathcal{C})$ and queries over these patterns. Temporal graph queries consist commonly of two parts: (i) path description to traverse the graph nodes, at both structural and temporal dimensions; (ii) arithmetic expressions on nodes, edges, and attribute values. These queries can be expressed, for instance, using existing temporal graph algebra *TODO*: **readd citation**

A decision present in $\mathcal{D}$ is defined as a set of executed actions, *i.e.,* a subset of $\mathcal{A}$. Formally, $\mathcal{D} = \{\ \mathcal{A}_\mathcal{D} | \mathcal{A}_\mathcal{D} \subseteq \mathcal{A}\}$. We assume that each action should result from one decision: $\forall a \in \mathcal{A}, \forall d1, d2 \in \mathcal{D} | a \in d1 \wedge a \in d2 \rightarrow d1 = d2$.

[1]The set of nodes resulting from the update of a given node

Actions $\mathcal{A}$ can be regarded as a set of relations or isomorphisms mapping a source temporal graph pattern $\mathbb{P}^C_{[t_j,t_k]}$ to a target one $\mathbb{P}^C_{[t_l,t_m]}(\mathcal{C})$, $\mathcal{A}_x : (\mathcal{C}) \times I_s \rightarrow (\mathcal{C}) \times I_t$, wherein:

- $\forall (t_j, t_k) \in I_s, \forall (t_l, t_m) \in I_t \rightarrow t_j \leq t_k \leq t_l \leq t_m$
- $\forall e \in \mathbb{P}_{[t_j,t_k[}, \exists (i, c) \subset \mathcal{V}^T(e) | i \cap [t_j, t_k[ \neq \emptyset$
- $\forall e \in \mathbb{P}_{[t_l,t_m[}, \exists (i, c) \subset \mathcal{V}^T(e) | i \cap [t_l, t_m[ \neq \emptyset$

The LHS of the relation depicts the temporal graph elements over which an action is applied. The side effects of these actions are represented by the RHS. Every relation may have a set of application conditions. They describe the circumstances under which an action should take place. These application conditions are either positive, should hold, or negative, should not hold. Application conditions come in the form of temporal graph invariants.

Finally, we associate to $\mathcal{A}$ a temporal function $\mathcal{X}$ to determine the time interval at which an action has been executed. Formally, $\mathcal{X} : (\mathcal{A} \cup \mathcal{D}) \rightarrow I$. As for decisions, we assume that once created they are always valid. Their start time can be derived from the set of actions belonging to them. More precisely, it corresponds to the creation time of the least recently created action. Whilst, their end time is equal to positive infinity. Formally, $\forall d \in \mathcal{D} \rightarrow \mathcal{V}(d) = [l, t_\infty[$, where $l = \min_{a \in \mathcal{A}_d} \{\mathcal{V}(a)[start]\}$.

### 2.2.2 Formalism applied on the use case

Through this section, we will formalize the use case defined in

According to the formalism, $K_{SG}$ represents the adaptive process such as $K_{SG} = (\mathcal{C}_{\mathcal{SG}}, \mathcal{R}_{\mathcal{SG}}, \mathcal{A}_{\mathcal{SG}}, \mathcal{D}_{\mathcal{SG}})$.

# Glossary

**action** "Process that, given the context and requirements as input, adjusts the system behavior", IEEE Standards [**?**]. 4

**circumstance** State of the knowledge when a decision has been taken. 4

**context** In this document, I use the definition provided by Anind K. Dey [**?**]: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and [the system], including the user and [the system] themselves". 4, 5

**decision** A set of actions taken after comparing the state of the knowledge with the requirement. 4

**knowledge** The knowledge of an adaptive system gathers information about the context, actions and requirements. 4

**requirement** "(1) Statement that translates or expresses a need and its associated constraints and conditions, (2) Condition or capability that must be met or possessed by a system [...] to satisfy an agreement, standard, specification, or other formally imposed documents", IEEE Standards [**?**]. 4