Consistent Persistence of Context-Dependent Runtime Models

Thomas Kühn and Christopher Werner
Software Technology Group
Technische Universität Dresden
Dresden, Germany
Email: {thomas.kuehn3, christopher.werner}@tu-dresden.de

Tobias Jäkel
Database Systems Group
Technische Universität Dresden
Dresden, Germany
Email: tobias.jaekel@tu-dresden.de

***Abstract*—Today's complex software systems act in various situations and contexts and thus, have to adapt themselves correspondingly during runtime. To model and represent the underlying context-dependent domain knowledge, contextual modeling languages, such as the Compartment Role Object Model (CROM), can be employed. However, these models and their instances become unwieldy rather quickly and are subject to many adaptations. Especially, when persisting a runtime model of a self-adaptive system this becomes a huge performance bottleneck. Notably, though not all elements of a context-dependent domain model have to be persisted to save the overall state of the application. Yet, simply removing information can easily lead to inconsistent models and instances in the database. To remedy too much or too less data saving and maintain adaptation processes, the persistent elements of a context-dependent domain model have to be annotated, such that the persisted domain model and instance is consistent with the runtime domain model and instance. For our solution, we introduce a formal approach to derive a persistent CROM from an arbitrary CROM model with persistence annotations, such that the persistent CROM is well-formed and consistent to the domain model and instance at runtime. In conclusion, this will allow context-aware systems to persist partial runtime model instances of context-dependent domain models while guaranteeing their consistency and the automatic adaptation of the persistent model after adapting the domain model.**

Keywords–CROM; RSQL; context-dependent domain model; persistency; transformation function.

I. INTRODUCTION

Self-adaptive Systems (SaS) have been conquering a lot of areas in automation, like robotics, control systems, or even home automation [1]. In recent years, several definitions of SaS arose, as shown in [1][2]. Notably, all definitions share the characteristic that SaS monitor themselves, their environment, and/or related components to adjust their behavior accordingly, e.g., dynamic production systems, autonomous mobile robots, and smart home solutions. To represent SaS' knowledge bases, current SaS employ context-dependent domain models [3][4], like CROM [5]. These domain models capture the system's situational state by means of contexts [2], as well as its context-dependent relations and constraints. Moreover, they are adapted and extended over time. For instance, a self-adaptive smart home, which monitors itself and features two basic contexts: a regular context and an emergency context. In the emergency context, for example, the front door is unlocked, such that the rescue team may enter faster, or the connected smart devices emit alarm sounds to alert the residents. However, such context-dependent domain models become unwieldy rather quickly, because all environmental information, as well as the system's context-dependent structure is modeled. This, especially, holds true for the resulting instances of such models at runtime. Considering persistence, this will result in a huge database containing potentially unnecessarily stored information, such as auxiliary sensor data. Yet, simply removing this unnecessary information can easily lead to inconsistent, invalid models and instances in the database [5]. Moreover, in case of system failure and recovery, this leads to invalid model instances at runtime. Especially, in case of context-dependent domain models, such as CROM [6], invalid instances can ultimately lead to unanticipated system behavior [7, p. 58]. For instance, persisting empty contexts or contextually unassigned behavior violates CROM's validity, as shown in [8]. To remedy this, the persistent elements of a context-dependent domain model must be annotated, such that the persisted domain model and instance is consistent with the runtime domain model and instance. The adaptation, like modification of running contexts and integration of new contexts, of the context-dependent domain models leads to an adaptation of the underlying database schema and the created persistency annotations. To manage this adaptation scenario, like adding a new context-aware application to a smart home, an automated transformation algorithm is needed that maps the update of the domain model to the evolution of the persisted domain model and underlying database schema. We employ CROM [5] as modeling language for context-dependent domain models and provide a formal approach for deriving both a well-formed persistence model, as well as a valid, reduced instance from an arbitrarily annotated CROM model and corresponding instances. Moreover, we utilize the role-based contextual database system (RSQL) [9] as the corresponding target database system for the resulting persistent models and instances. Finally, we demonstrate our algorithm utilizing a small example scenario within a smart home setting, and prove that our method guarantees the consistency of the resulting persistence models and reduced instances [10]. In sum, this will allow SaS to persist partial runtime model instances of context-dependent domains models before and after adaptation steps while guaranteeing their consistency.

The paper is organized as follows: Before delving into the formal definitions, Section II introduces a simple smart home scenario as our running example. Afterwards, Section III presents a brief introduction to both CROM and RSQL. In Section IV, our formal transformation algorithm is defined. Moreover, it is proven that this algorithm ensures well-formedness and validity persisted context-dependent domain models and instances. To illustrate the application of our approach, Section V applies the transformation algorithm to the running example. Related work is discussed in Section VI and the paper is concluded in Section VII.

 ISBN: 978-1-61208-706-1

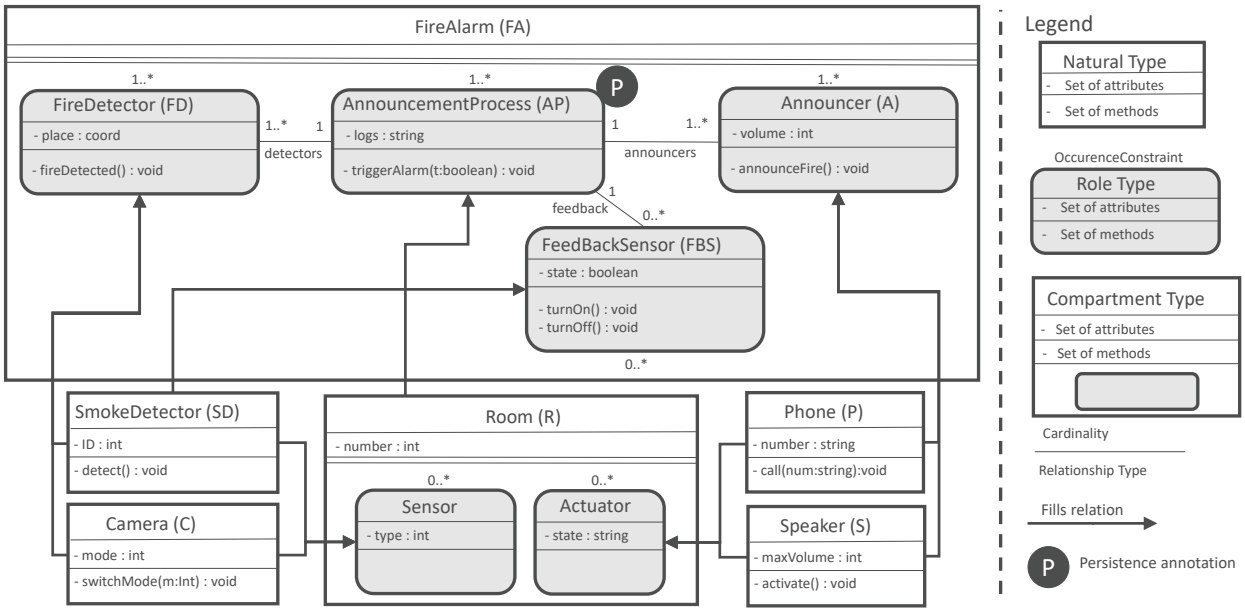

Figure 1. Annotated CROM Model of a Fire Alarm Scenario.

II. Running Example

Emergencies usually require SaS behavior and consequently utilize context-dependent domain models, because typical behavior is substituted with an adequate emergency response. Henceforth, we focus on fire as an emergency within a smart home, illustrated in Figure 1. Like any regular house, the smart home setting features *Rooms (R)*. Additionally, we assume that each *R* may contain several *Sensors* and *Actuators*. A *Sensor* can be either a *SmokeDetector (SD)* or a *Camera (C)*. As player type for the *Actuator* role type, we assume Phones (P), as well as *Speakers (S)*. While speakers are stationary, phones have the tendency to move around with their owner. In each room, with at least one sensor and actuator, the *FireAlarm (FA)* compartment is created. The available sensors and actuators of the room will start playing the *FireDetector (FD)* and *Announcer (A)* role type, respectively. In case a fire is detected by a fire detector, the *AnnouncementProcess (AP)* is triggered, which announces the fire alarm via all actuators playing the announcer role. For example, a smart speaker could announce the fire by activating noisy sounds or notifying the fire fighter department via Internet. All the detection and announcement procedures are coordinated by the *AP* role type, which also holds the log information. Additionally, our scenario requires the system to store the log information of each fire alarm persistently in a database system. Thus, the most basic annotation is the *AP* role type, which is in fact an invalid model with respect to the CROM metamodel. In case of restoring the system after a breakdown, an *AP* role could not be situated in any compartment, since this information will not be persistently stored. However, applying the *Persistency Transformation* algorithm φ will ensure a consistent database model by adding additional types to the database schema.

III. Preliminaries

Before describing our method to restrict a context-dependent domain model to a consistent partial persistence model, we first introduce CROM and RSQL to model respectively persistent context-dependent domain models.

A. Compartment Role Object Model

CROM [5] permits modeling dynamic, context-dependent domains by introducing *compartment types* to represent an objectified context, i.e., containing *role types* and *relationship types*. *Natural types*, in turn, fulfill role types in multiple compartment types. The following definitions are retrieved from [5], where a more detailed discussion can be found.

Definition 1 (Compartment Role Object Model)**.** *Let* NT*,* RT*,* CT*, and* RST *be mutual disjoint sets of Natural Types, Role Types, Compartment Types, and Relationship Types. Then,* $\mathcal{M} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{rel})$ *is a* CROM*, where* $\mathsf{fills} \subseteq \mathsf{T} \times \mathsf{CT} \times \mathsf{RT}$ *is a relation and* $\mathsf{rel} : \mathsf{RST} \times \mathsf{CT} \to (\mathsf{RT} \times \mathsf{RT})$ *is a partial function. Here,* $\mathsf{T} := \mathsf{NT} \cup \mathsf{CT}$ *denotes the set of all rigid types. A CROM is* well-formed *if it holds that:*

$$\forall rt \in \mathsf{RT}\ \exists\ t \in \mathsf{T}\ \exists! ct \in \mathsf{CT} : (t, ct, rt) \in \mathsf{fills} \tag{1}$$

$$\forall ct \in \mathsf{CT} : (t, ct, rt) \in \mathsf{fills} \tag{2}$$

$$\forall rst \in \mathsf{RST}\ \exists ct \in \mathsf{CT} : (rst, ct) \in \boldsymbol{domain}(\mathsf{rel}) \tag{3}$$

$$\forall (rt_1, rt_1)\ \in \boldsymbol{codomain}(\mathsf{rel}) : rt_1 \neq rt_2 \tag{4}$$

$$\forall (rst, ct)\ \in \boldsymbol{domain}(\mathsf{rel}) : \mathsf{rel}(rst, ct) = (rt_1, rt_2) \wedge (\_, rt_1, ct), (\_, rt_2, ct) \in \mathsf{fills} \tag{5}$$

In detail, fills denotes that rigid types can play roles of a certain role type in which compartment type and rel capture the two role types at the respective ends of each relationship type. The well-formedness rules ensure that the fills relation is surjective (1); each compartment type has a nonempty, disjoint set of role types as its parts (2, 3); and rel maps each relationship type to exactly two distinct role types of the same compartment type (4, 5). For a given function $f : A \to B$, $\mathbf{domain}(f) = A$ returns the domain and $\mathbf{codomain}(f) = B$ the range of f. For comprehensibility, we use subscripts to indicate the model a set, relation or function belongs to, e.g., $\mathsf{RT}_{\mathcal{M}}$ denotes the set of role types of the CROM $\mathcal{M}$. Accordingly, a CROM can be constructed for the fire alarm scenario, depicted in Figure 1.

 ISBN: 978-1-61208-706-1

Example 1 (Fire Alarm Model)**.** *Let* $\mathcal{F} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{parts}, \mathsf{rel})$ *be the model of the fire alarm scenario, where the components are defined as:*

$$\begin{aligned}
\mathsf{NT} &:= \{\mathsf{SD}, \mathsf{C}, \mathsf{P}, \mathsf{S}\} & \mathsf{CT} &:= \{\mathsf{FA}, \mathsf{R}\}\\
\mathsf{RT} &:= \{\mathsf{FD}, \mathsf{AP}, \mathsf{A}, \mathsf{FBS}, \mathsf{Sensor}, \mathsf{Aktuator}\}\\
\mathsf{RST} &:= \{\mathsf{detectors}, \mathsf{announcers}, \mathsf{feedback}\}\\
\mathsf{fills} &:= \{(\mathsf{SD}, \mathsf{FA}, \mathsf{FD}), (\mathsf{C}, \mathsf{FA}, \mathsf{FD}), (\mathsf{A}, \mathsf{FA}, \mathsf{AP}),\\
&\qquad (\mathsf{S}, \mathsf{FA}, \mathsf{A}), \dots\}\\
\mathsf{rel} &:= \{(\mathsf{detectors}, \mathsf{FA}) \to (\mathsf{FD}, \mathsf{AP}),\\
&\qquad (\mathsf{feedback}, \mathsf{FA}) \to (\mathsf{AP}, \mathsf{FBS}),\\
&\qquad (\mathsf{announcers}, \mathsf{FA}) \to (\mathsf{AP}, \mathsf{A}), \dots\}
\end{aligned}$$

Unsurprisingly, a well-formed CROM can directly encode concepts of context-dependent domains. A CROM instance features naturals, roles, compartments and relationships.

Definition 2 (Compartment Role Object Instance)**.** *Let* $\mathcal{M} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{rel})$ *be a well-formed CROM and* N*,* R*, and* C *be mutual disjoint sets of Naturals, Roles and Compartments, respectively. Then, a* Compartment Role Object Instance (CROI) *of* $\mathcal{M}$ *is a tuple* $\mathfrak{i} = (N, R, C, \mathsf{type}, \mathsf{plays}, \mathsf{links})$*, where* $\mathsf{type} : (N \to \mathsf{NT}) \cup (R \to \mathsf{RT}) \cup (C \to \mathsf{CT})$ *is a labeling function,* $\mathsf{plays} \subseteq (N \cup C) \times C \times R$ *a relation, and* $\mathsf{links} : \mathsf{RST} \times C \to \mathcal{P}(R \times R)$ *is a total function. Moreover,* $O := N \cup C$ *denotes the set of all objects in* $\mathfrak{i}$*. To be* compliant *to the model* $\mathcal{M}$ *the instance* $\mathfrak{i}$ *must satisfy the following conditions:*

$$\forall (o, c, r) \in \mathsf{plays} : (\mathsf{type}(o), \mathsf{type}(c), \mathsf{type}(r)) \in \mathsf{fills} \tag{6}$$

$$\forall (o, c, r),\ (o, c, r') \in \mathsf{plays} : r \neq r' \Rightarrow \mathsf{type}(r) \neq \mathsf{type}(r') \tag{7}$$

$$\forall r \in R\ \exists! o \in O\ \exists! c \in C : (o, c, r) \in \mathsf{plays} \tag{8}$$

$$\begin{aligned}
\forall rst \in \mathsf{RST} \forall c \in C\ \forall (r_1, r_2) \in \mathsf{links}(rst, c) :\\
(rst, \mathsf{type}(c)) \in \textbf{\textit{domain}}(\mathsf{rel}) \wedge\\
\mathsf{rel}(rst, \mathsf{type}(c)) = (\mathsf{type}(r_1), \mathsf{type}(r_2)) \wedge\\
(_, c, r_1), (_, c, r_2) \in \mathsf{plays}
\end{aligned} \tag{9}$$

The type function assigns a distinct type to each instance, plays identifies the objects (either natural or compartment) playing a certain role in a specific compartment, and links captures the roles currently linked by a relationship type in a certain compartment. A compliant CROI guarantees the consistency of both the plays relation and the links function with the model $\mathcal{M}$. Axioms (6), (7) and (8) restrict the plays relation, such that it is consistent to the types defined in the fills relation and the parts function, an object is prohibited to play instances of the same role type multiple times in the same compartment, and each role has one distinct player in one distinct compartment, respectively. In contrast, Axiom (9) ensures that the links function only contains those roles, which participate in the same compartment c as the relationship and whose types are consistent to the relationship's definition in the rel function.

Admittedly, neither Definition 1 nor 2 captures constraints of context-dependent domains. Hence, two context-dependent constraints, i.e., *occurrence constraints* and *relationship cardinalities* are introduced. Henceforth, cardinalities are given as $\mathsf{Card} \subset \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ with $i \leq j$, whereas elements of Card are written as $i..j$.

Next, the *Constraint Model* is defined to collect all constraints imposed on a particular CROM $\mathcal{M}$.

Definition 3 (Constraint Model)**.** *Let* $\mathcal{M} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{rel})$ *be a well-formed CROM. Then* $\mathcal{C} = (\mathsf{occur}, \mathsf{card})$ *is a* Constraint Model over $\mathcal{M}$*, where* $\mathsf{occur} : \mathsf{CT} \to \mathcal{P}(\mathsf{Card} \times \mathsf{RT})$ *and* $\mathsf{card} : \mathsf{RST} \times \mathsf{CT} \to (\mathsf{Card} \times \mathsf{Card})$ *are partial functions. A Constraint Model is* compliant *to* $\mathcal{M}$*, iff:*

$$\begin{aligned}
\forall ct \in \textbf{\textit{domain}}(\mathsf{occur})\ \forall (_, rt) \in \mathsf{occur}(ct) :\\
(_, ct, rt) \in \mathsf{fills}
\end{aligned} \tag{10}$$

$$\textbf{\textit{domain}}(\mathsf{card}) \subseteq \textbf{\textit{domain}}(\mathsf{rel}) \tag{11}$$

In detail, occur collects a cardinality limiting the occurrence of the given role type in each compartment. Moreover, card assigns a cardinality to each relationship type. Notably, all these constraints are defined context-dependent, i.e., no constraint crosses the boundary of a compartment type. In contrast to [5], our definition excludes empty counter roles ε and *Role Groups*. Similar to the CROM $\mathcal{F}$, the corresponding compliant constraint model is easily derived, from Figure 1:

Example 2 (Fire Alarm Constraints)**.** *Let* $\mathcal{F}$ *be the fire alarm from Example 1. Then* $\mathcal{C}_{\mathcal{F}} = (\mathsf{occur}, \mathsf{card})$ *is the compliant constraint model with the following components:*

$$\begin{aligned}
\mathsf{occur} &:= \{\mathsf{FA} \to \{(1..\infty, \mathsf{FD}), (1..\infty, \mathsf{AP}), (1..\infty, \mathsf{A})\}\}\\
\mathsf{card} &:= \{(\mathsf{detectors}, \mathsf{FA}) \to (1..\infty, 1..1),\\
&\qquad (\mathsf{announcers}, \mathsf{FA}) \to (1..1, 1..\infty),\\
&\qquad (\mathsf{feedback}, \mathsf{FA}) \to (1..1, 0..\infty)\}
\end{aligned}$$

Finally, the validity of a given CROI is defined with respect to a CROM and corresponding constraint model.

Definition 4 (Validity)**.** *Let* $\mathcal{M} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{rel})$ *be a well-formed CROM,* $\mathcal{C} = (\mathsf{occur}, \mathsf{card})$ *a constraint model on* $\mathcal{M}$*, and* $\mathfrak{i} = (N, R, C, \mathsf{type}, \mathsf{plays}, \mathsf{links})$ *a CROI compliant to* $\mathcal{M}$*. Then* $\mathfrak{i}$ is valid with respect to $\mathcal{C}$ *iff the following conditions hold:*

$$\forall c \in C\ \forall (i..j, rt) \in \mathsf{occur}(\mathsf{type}(c)) : i \leq \left|R^c_{rt}\right| \leq j \tag{12}$$

$$\begin{aligned}
\forall c \in C \forall rst \in \mathsf{RST} :\ & \mathsf{rel}(rst, \mathsf{type}(c)) = (rt_1, rt_2) \wedge\\
& \mathsf{card}(rst, \mathsf{type}(c)) = (i..j, k..l) \wedge\\
& \left(\forall r_2 \in R^c_{rt_2} : i \leq \left|\mathsf{pred}(rst, c, r_2)\right| \leq j\right) \wedge\\
& \left(\forall r_1 \in R^c_{rt_1} : k \leq \left|\mathsf{succ}(rst, c, r_1)\right| \leq l\right)
\end{aligned} \tag{13}$$

Here, $R^c_{rt} := \{r \in R \mid (o, c, r) \in \mathsf{plays} \wedge \mathsf{type}(r) = rt\}$ *denotes the set of roles of type* rt *played in a compartment* c*;* $\mathsf{pred}(rst, c, r) := \{r' \mid (r', r) \in \mathsf{links}(rst, c)\}$ *and* $\mathsf{succ}(rst, c, r) := \{r' \mid (r, r') \in \mathsf{links}(rst, c)\}$ *collects all predecessors respectively successors of a given role* r *with respect to an* rst*.*

Each axiom verifies a particular set of constraints. Axiom (12) validates the occurrence of role types. In essence, it checks the number of roles of the given type played in a constrained compartment. In contrast, (13) checks whether relationships respect the imposed cardinality constraints. In conclusion, the formal model easily captures the context-dependent concepts and constraints. Moreover, it allows for checking the well-formedness of CROMs and validity of CROIs. Although a database schema can be generated for a CROM (including the Constraint Model), due to the lack of persistence annotations, the full model must be stored for compliance and validity.

 ISBN: 978-1-61208-706-1

B. Role-Based Contextual Database System (RSQL)

RSQL directly addresses the persistence of context-dependent information in a database system [9]. In particular, RSQL combines a metatype distinction in the database model, an adapted query language on the database model's basis, and finally a proper result representation [11]. The database model, including the metatype distinction, consists of two levels, (i) the schema level and (ii) the instance level. On the schema level, RSQL introduces Dynamic Data Types (DDT) that combine the notion of an entity type with the notion of roles while fully implementing the metatype distinction on the basis of CROM [11, p. 72]. On the instance level, the database model introduces *Dynamic Tuples* (DT) [11, p. 78], that are defined to allow for dynamic structure adaptations during runtime without changing an entity's overall type [9]. Hence, DDTs define the space in which DTs might expand or shrink their structure, depending on the context they are acting in. Also, RSQL features a set of formal operators to process context-dependent information on the basis of DTs [11, p. 84].

The query language, as external database system interface, features an individual data definition (DDL), data manipulation (DML), and data query language (DQL) [9]. As the database model, the query language implements a metatype distinction on the basis of CROM as first class citizen. The list of DDL statements and grammar, DML statements, and DQL statement grammar are shown in [11, p. 116, p. 122, p. 127].

Finally, to complete the database integration of context-dependent information, RSQL returns *RSQL Result Nets*. These represent a novel data structure for results, which feature various functionalities to navigate through players, their roles, compartments, and relationships [11, p. 147].

IV. Persistence Algorithm

Henceforth, we present an algorithm that transforms an annotated CROM model into a dedicated CROM model for persistence. In detail, we first introduce persistence annotations to annotate CROM model elements. Afterwards, the transformation algorithm is presented. Finally, we prove that given a well-formed CROM model with persistence annotations and a valid CROM instance, the resulting limited CROM model is well-formed and the restricted CROI is valid.

A. Persistence Annotation

Accordingly, the following definition extends CROM by introducing annotations for modeling elements.

Definition 5 (Persistence Annotation)**.** *Let* $\mathcal{M} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{rel})$ *be an arbitrary CROM, then* $P = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{rel})$ *denotes a persistence annotation of* $\mathcal{M}$*, whereas* $\mathsf{NT}_P \subseteq \mathsf{NT}_\mathcal{M}$*,* $\mathsf{RT}_P \subseteq \mathsf{RT}_\mathcal{M}$*,* $\mathsf{CT}_P \subseteq \mathsf{CT}_\mathcal{M}$*, and* $\mathsf{rel}_P \subseteq \boldsymbol{domain}(\mathsf{rel}_\mathcal{M})$

In general, this definition permits to select a subset of natural types, role types, compartment types, and context-dependent relationship types. For the sake of simplicity, we excluded attributes, because persisting attributes does not add any complexity to the proposed method. In case of the fire alarm scenario (Figure 1), the persistence annotation for the CROM $\mathcal{F}$ could be defined as $P = (\emptyset, \{AP\}, \emptyset, \emptyset)$. As the transformation must be aware of compartment types, which are existentially dependent on at least one of its containing role types, we define the following auxiliary definitions.

Definition 6 (Existential Parts)**.** *Let* $\mathcal{M} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{parts}, \mathsf{rel})$ *be an arbitrary CROM,* $\mathcal{C} = (\mathsf{occur}, \mathsf{card})$ *a constraint model on* $\mathcal{M}$*, and* $ct \in \mathsf{CT}$ *an arbitrary compartment type in* $\mathcal{M}$*. Then* $\boldsymbol{ext}(ct) \coloneqq \{rt \mid (t, ct, rt) \in \mathsf{fills} \wedge (i..j, rt) \in \mathsf{occur}(ct) \wedge i \geq 1\}$ *collects the set of* existential parts *of the compartment type* ct*.*

In general, this definition introduces the **ext** function to determine the existential parts of the given compartment type, i.e., contain a role type with an occurrence constraint $(i..j)$ with $i \geq 1$. For instance, while the room R has no existential parts, the fire alarm compartment type FA has three, i.e., $\mathbf{ext}(\mathsf{FA}) = \{\mathsf{FD}, \mathsf{AP}, \mathsf{A}\}$.

B. Transformation Algorithm

After defining persistence annotations over CROM models, it is possible to define the *Persistency Transformation* for arbitrary CROM models and corresponding constraint models, as follows:

Definition 7 (Persistency Transformation)**.** *Let* $\mathcal{M}$ *be a well-formed CROM,* $\mathcal{C}$ *a constraint model compliant to* $\mathcal{M}$*, and* $P = (\mathsf{NT}_P, \mathsf{RT}_P, \mathsf{CT}_P, \mathsf{rel}_P)$ *a persistence annotation over* $\mathcal{M}$*. Then* $(\mathcal{N}, \mathcal{D}) = \varphi(\mathcal{M}, \mathcal{C}, P)$ *constructs the persistence model* $\mathcal{N} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{rel})$ *and respective constraint model* $\mathcal{D} = (\mathsf{occur}, \mathsf{card})$ *from the given CROM, constraint model, and persistence annotation.* φ *first computes the* $\mathsf{fills}_\mathcal{N}$ *relation and* $\mathsf{rel}_\mathcal{N}$ *function by applying the inference rules depicted in Figure 2. From them, the sets* $\mathsf{NT}_\mathcal{N}$*,* $\mathsf{RT}_\mathcal{N}$*,* $\mathsf{CT}_\mathcal{N}$*,* $\mathsf{RST}_\mathcal{N}$ *can be determined as:*

$$\mathsf{T}_\mathcal{N} \coloneqq \{t \mid (t, ct, rt) \in \mathsf{fills}_\mathcal{N}\} \tag{14}$$

$$\mathsf{CT}_\mathcal{N} \coloneqq \{ct \mid (t, ct, rt) \in \mathsf{fills}_\mathcal{N}\} \tag{15}$$

$$\mathsf{RT}_\mathcal{N} \coloneqq \{rt \mid (t, ct, rt) \in \mathsf{fills}_\mathcal{N}\} \tag{16}$$

$$\mathsf{NT}_\mathcal{N} \coloneqq \mathsf{NT}_P \cup (\mathsf{T}_\mathcal{N} \setminus \mathsf{CT}_\mathcal{N}) \tag{17}$$

$$\mathsf{RST}_\mathcal{N} \coloneqq \{rst \mid (rst, ct) \in \boldsymbol{domain}(\mathsf{rel}_\mathcal{N})\} \tag{18}$$

Finally, the partial function occur *and* card *can be restricted to the model* $\mathcal{N}$*, as showcased in Figure 2.*

To put it succinctly, the *Persistency Transformation* first constructs the CROM $\mathcal{N}$ for persistence by inductively creating the $\mathsf{fills}_\mathcal{N}$ relation and the $\mathsf{rel}_\mathcal{N}$ partial function before creating the carrier sets $\mathsf{NT}_\mathcal{N}$, $\mathsf{RT}_\mathcal{N}$, $\mathsf{CT}_\mathcal{N}$, $\mathsf{RST}_\mathcal{N}$. Lastly, the constraint model $\mathcal{D}$ is constructed by restricting the $\mathsf{occur}_\mathcal{D}$ and $\mathsf{card}_\mathcal{D}$ functions accordingly. In case of fills, the axioms (19), (20), (25), and (26) initialize $\mathsf{fills}_\mathcal{N}$ with respect to the annotated compartment types, role types, and relationships types, respectively. Afterwards, $\mathsf{fills}_\mathcal{N}$ is inductively extended in three ways. First, (23) checks all compartment types that fulfill a previously selected role type and adds all of its existential parts (role types), if any exist. Likewise, (24) checks all previously selected compartment types and includes all of its existential parts (role types), if any exist. Third, (25) and (26) check if a previously selected role type has a relationship to another role type with a cardinality $(i..j)$ with $i \geq 1$. By contrast, rel is initialized only with relationships either directly annotated (27) or contained in an annotated compartment type (28). Afterwards, this partial function is inductively expanded, if a previously selected role type has a relationship to another role type with a cardinality greater than zero. Consequently, the components of $\mathcal{N}$ are derived from the contents of both $\mathsf{fills}_\mathcal{N}$ and $\mathsf{rel}_\mathcal{N}$.

 ISBN: 978-1-61208-706-1

Inductive definition of $\mathsf{fills}_{\mathcal{N}}$:

$$\text{CTSel}\frac{ct \in \mathsf{CT}_P \qquad (t, ct, rt) \in \mathsf{fills}_{\mathcal{M}}}{(t, ct, rt) \in \mathsf{fills}_{\mathcal{N}}} \tag{19}$$

$$\text{RTSel}\frac{rt \in \mathsf{RT}_P \qquad (t, ct, rt) \in \mathsf{fills}_{\mathcal{M}}}{(t, ct, rt) \in \mathsf{fills}_{\mathcal{N}}} \tag{20}$$

$$\text{RelSelLeft}\frac{(rst, ct) \in \mathsf{rel}_P \qquad \mathsf{rel}_{\mathcal{M}}(rst, ct) = (rt_1, rt_2) \qquad (t, ct, rt_1) \in \mathsf{fills}_{\mathcal{M}}}{(t, ct, rt_1) \in \mathsf{fills}_{\mathcal{N}}} \tag{21}$$

$$\text{RelSelRight}\frac{(rst, ct) \in \mathsf{rel}_P \qquad \mathsf{rel}_{\mathcal{M}}(rst, ct) = (rt_1, rt_2) \qquad (t, ct, rt_2) \in \mathsf{fills}_{\mathcal{M}}}{(t, ct, rt_2) \in \mathsf{fills}_{\mathcal{N}}} \tag{22}$$

Induction cases for $\mathsf{fills}_{\mathcal{N}}$:

$$\text{CTExt}\frac{(ct_1, ct_2, \_) \in \mathsf{fills}_{\mathcal{N}} \qquad (t, ct_1, rt) \in \mathsf{fills}_{\mathcal{M}} \qquad ct_1 \in \mathsf{CT}_{\mathcal{M}} \qquad rt \in \mathbf{ext}_{\mathcal{M}}(ct_1)}{(t, ct_1, rt) \in \mathsf{fills}_{\mathcal{N}}} \tag{23}$$

$$\text{OccurExt}\frac{(s, ct, rt') \in \mathsf{fills}_{\mathcal{N}} \qquad rt \in \mathbf{ext}_{\mathcal{M}}(ct) \qquad (t, ct, rt) \in \mathsf{fills}_{\mathcal{M}}}{(t, ct, rt) \in \mathsf{fills}_{\mathcal{N}}} \tag{24}$$

$$\text{RelExtRight}\frac{\begin{array}{c}(s, ct, rt_1) \in \mathsf{fills}_{\mathcal{N}} \qquad (t, ct, rt_2) \in \mathsf{fills}_{\mathcal{M}} \qquad (rst, ct) \in \mathbf{domain}(\mathsf{card}_{\mathcal{C}}) \\ \mathsf{rel}_{\mathcal{M}}(rst, ct) = (rt_1, rt_2) \qquad \mathsf{card}_{\mathcal{C}}(rst, ct) = (i..j, k..l) \qquad k \geq 1\end{array}}{(t, ct, rt_2) \in \mathsf{fills}_{\mathcal{N}}} \tag{25}$$

$$\text{RelExtLeft}\frac{\begin{array}{c}(t, ct, rt_2) \in \mathsf{fills}_{\mathcal{N}} \qquad (s, ct, rt_1) \in \mathsf{fills}_{\mathcal{M}} \qquad (rst, ct) \in \mathbf{domain}(\mathsf{card}_{\mathcal{C}}) \\ \mathsf{rel}_{\mathcal{M}}(rst, ct) = (rt_1, rt_2) \qquad \mathsf{card}_{\mathcal{C}}(rst, ct) = (i..j, k..l) \qquad i \geq 1\end{array}}{(s, ct, rt_1) \in \mathsf{fills}_{\mathcal{N}}} \tag{26}$$

Inductive definition of $\mathsf{rel}_{\mathcal{N}}$:

$$\text{RelSel}\frac{(rst, ct) \in \mathsf{rel}_P}{\mathsf{rel}_{\mathcal{N}}(rst, ct) \coloneqq \mathsf{rel}_{\mathcal{M}}(rst, ct)} \tag{27}$$

$$\text{RelInCT}\frac{ct \in \mathsf{CT}_P \qquad (rst, ct) \in \mathbf{domain}(\mathsf{rel}_{\mathcal{M}})}{\mathsf{rel}_{\mathcal{N}}(rst, ct) \coloneqq \mathsf{rel}_{\mathcal{M}}(rst, ct)} \tag{28}$$

Induction case for $\mathsf{rel}_{\mathcal{N}}$:

$$\text{RelExt}\frac{\begin{array}{c}(s, ct, rt_1), (t, ct, rt_2) \in \mathsf{fills}_{\mathcal{N}} \qquad (rst, ct) \in \mathbf{domain}(\mathsf{card}_{\mathcal{C}}) \qquad \mathsf{rel}_{\mathcal{M}}(rst, ct) = (rt_1, rt_2) \\ \mathsf{card}_{\mathcal{C}}(rst, ct) = (i..j, k..l) \qquad (i \geq 1 \vee k \geq 1)\end{array}}{\mathsf{rel}_{\mathcal{N}}(rst, ct) \coloneqq \mathsf{rel}_{\mathcal{M}}(rst, ct)} \tag{29}$$

Definition of $\mathsf{occur}_{\mathcal{D}}$ and $\mathsf{card}_{\mathcal{D}}$:

$$\text{OccurIn}\frac{(t, ct, rt) \in \mathsf{fills}_{\mathcal{N}} \qquad (i..j, rt) \in \mathsf{occur}_{\mathcal{C}}(ct)}{(i..j, rt) \in \mathsf{occur}_{\mathcal{D}}(ct)} \tag{30}$$

$$\text{CardIn}\frac{(rst, ct) \in \mathbf{domain}(\mathsf{rel}_{\mathcal{N}}) \qquad (rst, ct) \in \mathbf{domain}(\mathsf{card}_{\mathcal{C}})}{\mathsf{card}_{\mathcal{D}}(rst, ct) \coloneqq \mathsf{card}_{\mathcal{C}}(rst, ct)} \tag{31}$$

Figure 2. Inductive definition of $\mathsf{fills}_{\mathcal{N}}$, $\mathsf{rel}_{\mathcal{N}}$, $\mathsf{occur}_{\mathcal{D}}$, and $\mathsf{card}_{\mathcal{D}}$.

Besides the CROM $\mathcal{N}$, the components of the constraint model $\mathcal{D}$ are defined inductively, as well. In detail, (30) and (31) restrict the occurrence respectively cardinality constraints of $\mathcal{C}$ to those compliant to the target model $\mathcal{N}$, i.e., include all rules from $\mathsf{occur}_{\mathcal{C}}$ for all compartment types and role types in $\mathcal{N}$ and the cardinality from $\mathsf{card}_{\mathcal{C}}$ for all relationship types defined in $\mathsf{rel}_{\mathcal{N}}$. In conclusion, φ generates both a CROM $\mathcal{N}$ and a corresponding $\mathcal{D}$ from the given CROM $\mathcal{M}$, the constraint model $\mathcal{C}$, and persistence annotation P.

Up to this point, this definition is only applicable to the type level, and has no direct effect on the instance level. However, to store an instance of an annotated CROM model, it must be reduced, as well.

Definition 8 (Instance Restriction)**.** *Let* $\mathcal{M} = (\mathsf{NT}, \mathsf{RT}, \mathsf{CT}, \mathsf{RST}, \mathsf{fills}, \mathsf{rel})$ *be a well-formed CROM and* $\mathfrak{i}$ *an arbitrary CROI. Then* $\mathfrak{p} \coloneqq \Psi(\mathcal{M}, \mathfrak{i})$ is a restriction of $\mathfrak{i}$ with respect to $\mathcal{M}$. *In detail,* $\mathfrak{p} = (N, R, C, \mathsf{type}, \mathsf{plays}, \mathsf{links})$ *is determined by first applying the induction rules, shown in Figure 3, to determine* $\mathsf{plays}_{\mathfrak{p}}$ *and* $\mathsf{links}_{\mathfrak{p}}$. *Afterwards, the other components are defined as follows:*

$$N_{\mathfrak{p}} \coloneqq \{o \mid (o, c, r) \in \mathsf{plays}_{\mathfrak{p}} \wedge \mathsf{type}_{\mathfrak{i}}(o) \in \mathsf{NT}_{\mathcal{M}}\} \tag{32}$$

$$R_{\mathfrak{p}} \coloneqq \{r \mid (o, c, r) \in \mathsf{plays}_{\mathfrak{p}}\} \tag{33}$$

$$C_{\mathfrak{p}} \coloneqq \{o \mid (o, c, r) \in \mathsf{plays}_{\mathfrak{p}} \wedge \mathsf{type}_{\mathfrak{i}}(o) \in \mathsf{CT}_{\mathcal{M}}\} \cup \{c \mid (o, c, r) \in \mathsf{plays}_{\mathfrak{p}}\} \tag{34}$$

$$\mathsf{type}_{\mathfrak{p}} \coloneqq \mathsf{type}_{\mathfrak{i}} \tag{35}$$

Inductive definition of $\mathsf{plays}_\mathfrak{p}$:

$$\frac{(o,c,r)\in\mathsf{plays}_\mathfrak{i} \qquad (\mathsf{type}_\mathfrak{i}(o),\mathsf{type}_\mathfrak{i}(c),\mathsf{type}_\mathfrak{i}(r))\in\mathsf{fills}_\mathcal{M}}{(o,c,r)\in\mathsf{plays}_\mathfrak{p}} \tag{36}$$

Inductive definition of $\mathsf{links}_\mathfrak{p}$:

$$\frac{\begin{array}{c}(rst,c)\in\mathbf{domain}(\mathsf{links}_\mathfrak{i}) \qquad (r_1,r_2)\in\mathsf{links}_\mathfrak{i}(rst,c)\\ (rst,\mathsf{type}_\mathfrak{i}(c))\in\mathbf{domain}(\mathsf{rel}_\mathcal{M})\\ \mathsf{rel}_\mathcal{M}(rst,\mathsf{type}_\mathfrak{i}(c))=(\mathsf{type}_\mathfrak{i}(r_1),\mathsf{type}_\mathfrak{i}(r_2))\end{array}}{(r_1,r_2)\in\mathsf{links}_\mathfrak{p}(rst,c)} \tag{37}$$

Figure 3. Definitions of the restriction of $\mathsf{plays}_\mathfrak{p}$ and $\mathsf{links}_\mathfrak{p}$.

In fact, the restriction of a CROI $\mathfrak{i}$ with respect to a CROM $\mathcal{M}$ is derived by only including elements from $\mathsf{plays}_\mathfrak{i}$ whose types correspond to $\mathsf{fills}_\mathcal{M}$ (36) and by only including relationships from $\mathsf{links}_\mathfrak{i}$ that have been defined in $\mathcal{M}$ including the correct types of the left and right side (37). Afterwards, the carrier sets N, R, C are computed simply based on $\mathsf{plays}_\mathfrak{p}$. Notably, $\mathsf{type}_\mathfrak{i}$ is passed as is, because the typing of entities must be retained after restricting the CROI. In conclusion, both the *Persistency Transformation* φ and the *instance restriction* ψ work in concert. While φ generates the persistence CROM model $\mathcal{N}$ and constraints $\mathcal{D}$ from a given CROM model $\mathcal{M}$ with constraints $\mathcal{C}$, ψ can be employed to restrict a CROI instance $\mathfrak{i}$ of $\mathcal{M}$ to the persistence CROM model $\mathcal{N}$, ultimately, constructing the persistence CROI instance.

C. Well-formedness, Compliance, and Validity

Although, both transformations were designed thoroughly, their suitability for persisting context-dependent domain models is determined by their ability to retain the well-formedness, compliance, and validity of the created models and instances. In detail, this entails that given a well-formed CROM $\mathcal{M}$ and compliant constraint model $\mathcal{C}$, φ will always generate a well-formed CROM $\mathcal{N}$ and compliant constraint model $\mathcal{D}$ for persistence. Moreover, given an arbitrary CROI $\mathfrak{i}$ compliant to $\mathcal{M}$ and valid with respect to $\mathcal{C}$, then ψ will always create a restricted CROI $\mathfrak{p}$ that is compliant to $\mathcal{N}$ and valid with respect to $\mathcal{D}$. Thus, the resulting partial domain model can be safely stored in and loaded from a database system.

Conversely, we henceforth discuss three main theorems. First, we show that the *Persistency Transformation* ensures well-formedness of the model and compliance of the constraints. Second, we extend this result to the compliance and validity of instances of such CROM models. Finally, we show that each restricted instance is also a compliant and valid instance of the original model. Consequently, this guarantees the consistency of stored partial runtime model instances of context-dependent domains models. For brevity, the full proofs are omitted, but will be presented in a separate technical report.

Theorem 1 (Well-formedness and Compliance)**.** *Let $\mathcal{M}$ be a well-formed CROM, $\mathcal{C}$ a constraint model compliant to $\mathcal{M}$, and P a persistence annotation of $\mathcal{M}$. Then $(\mathcal{N},\mathcal{D}) \coloneqq \varphi(\mathcal{M},\mathcal{C},P)$ constructs a well-formed persistence model $\mathcal{N}$ and corresponding constraint model $\mathcal{D}$ compliant to $\mathcal{N}$.*

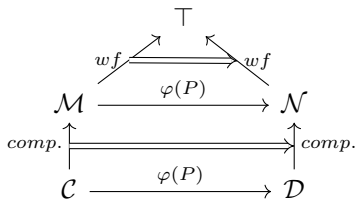

Figure 4. Commutative diagram for well-formedness and compliance.

Proof: Before proving this theorem, we can make the following observations when investigating the inference rules (cf. Figure 2). Specifically, from the structure of (19–29) we can deduce the following relations between a given CROM $\mathcal{M}$ and the resulting $\mathcal{N}$:

$$\mathsf{fills}_\mathcal{N}\subseteq\mathsf{fills}_\mathcal{M}$$
$$\mathbf{domain}(\mathsf{rel}_\mathcal{N})\subseteq\mathbf{domain}(\mathsf{rel}_\mathcal{M})$$
$$\forall(rst,ct)\in\mathbf{domain}(\mathsf{rel}_\mathcal{N}):\ \mathsf{rel}_\mathcal{N}(rst,ct)=\mathsf{rel}_\mathcal{M}(rst,ct)$$

Similarly, the structure of (30–31) entails the following relations between the constraint model $\mathcal{C}$ and $\mathcal{D}$:

$$\mathbf{domain}(\mathsf{occur}_\mathcal{D})\subseteq\mathbf{domain}(\mathsf{occur}_\mathcal{C})$$
$$\forall ct\in\mathbf{domain}(\mathsf{occur}_\mathcal{C}):\mathsf{occur}_\mathcal{D}(ct)\subseteq\mathsf{occur}_\mathcal{C}(ct)$$
$$\mathbf{domain}(\mathsf{card}_\mathcal{D})\subseteq\mathbf{domain}(\mathsf{card}_\mathcal{C})$$
$$\forall(rst,ct)\in\mathbf{domain}(\mathsf{card}_\mathcal{D}):\mathsf{card}_\mathcal{D}(rst,ct)=\mathsf{card}_\mathcal{C}(rst,ct)$$

Finally, Theorem 1 can be shown to hold for well-formed CROMs $\mathcal{M}$ and corresponding compliant constraint models $\mathcal{C}$ following the commutative diagram in Figure 4. To show that the well-formedness of $\mathcal{N}$ is implied by $\mathcal{M}$, we successively apply the relations between $\mathcal{N}$ and $\mathcal{M}$ to the axioms (1–5). Likewise, the compliance of $\mathcal{D}$ to $\mathcal{N}$ can be entailed from the compliance of $\mathcal{C}$ to $\mathcal{M}$ by applying the relations between $\mathcal{D}$ and $\mathcal{C}$ to axioms (10) and (11). ■

After proving that φ preserves the well-formedness and compliance of the generated persistence CROM and constraint model, the final step is to prove that ψ restricts a CROI $\mathfrak{i}$ compliant to $\mathcal{M}$ and valid with respect to $\mathcal{C}$ to a persistence CROI $\mathfrak{p}$, which is itself compliant and valid to the persistence CROM of $\mathcal{M}$ and $\mathcal{C}$, respectively.

Theorem 2 (Validity)**.** *Let $\mathcal{M}$ be a well-formed CROM, $\mathcal{C}$ a constraint model compliant to $\mathcal{M}$, P a persistence annotation of $\mathcal{M}$, as well as $\mathfrak{i}$ a CROI compliant to $\mathcal{M}$ and valid with respect to $\mathcal{C}$. Then the construction $(\mathcal{N},\mathcal{D}) \coloneqq \varphi(\mathcal{M},\mathcal{C},P)$ and the restriction $\mathfrak{p} \coloneqq \psi(\mathcal{N},\mathfrak{i})$ entails that $\mathfrak{p}$ is compliant to $\mathcal{N}$ and valid with respect to $\mathcal{D}$.*

Proof: Again, to prove Theorem 2 major conclusions can already be drawn from Definition 8 (cf. Figure 3) and the fact that $\mathfrak{i}$ is compliant to the well-formed CROM $\mathcal{M}$. Thus, the following relations between $\mathfrak{p}$ and $\mathfrak{i}$ can be deduced:

$$\mathsf{plays}_\mathfrak{p}\subseteq\mathsf{plays}_\mathfrak{i}$$
$$\mathbf{domain}(\mathsf{links}_\mathfrak{p})\subseteq\mathbf{domain}(\mathsf{links}_\mathfrak{i})$$
$$\forall(rst,c)\in\mathbf{domain}(\mathsf{links}_\mathfrak{i}):\mathsf{links}_\mathfrak{p}(rst,c)=\mathsf{links}_\mathfrak{i}(rst,c)$$

To show that Theorem 2 holds for all $\mathfrak{i}$ compliant to $\mathcal{M}$ and valid wrt. $\mathcal{C}$, we need to show the compliance of $\mathfrak{p}$ to $\mathcal{N}$ and the validity to $\mathcal{D}$ (cf. Figure 5).

 ISBN: 978-1-61208-706-1

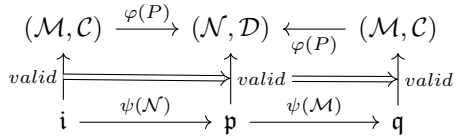

Figure 5. Commutative diagram for validity and lifted validity.

The compliance of $\mathfrak{p}$ to $\mathcal{N}$ can be shown by applying the relations between $\mathfrak{p}$ and $\mathfrak{i}$ to the axioms (6–9) assuming that $\mathfrak{i}$ is compliant to $\mathcal{M}$. Conversely, the validity of $\mathfrak{p}$ with respect to $\mathcal{D}$ is entailed from the validity of $\mathfrak{i}$ with respect to $\mathcal{C}$, as well as from Definition 8. Especially, it holds that if a role type is persisted $rt \in \mathsf{RT}_{\mathcal{N}}$ then all corresponding role (instances) are retained from $\mathfrak{i}$ in $\mathfrak{p}$, such that $R^c_{rt,\mathfrak{p}} = R^c_{rt,\mathfrak{i}}$ holds for all $c \in C_{\mathfrak{i}}$. Applying these entailments to (12) and (13), we can show that $\mathfrak{p}$ is valid with respect to $\mathcal{D}$. ■

While this proves the consistency of stored partial context-dependent domain models and instances, the question arises whether the persistent partial runtime model can be safely used as starting point after a system breakdown. To show this, we extend the notion of compliance and validity of a persisted CROI $\mathfrak{p}$ to the original CROM $\mathcal{M}$ and corresponding $\mathcal{C}$. Yet, the persisted CROI $\mathfrak{p}$ must first be lifted to the original CROM $\mathcal{M}$, i.e., $\psi(\mathcal{M},\mathfrak{p})$. This leads to the following theorem:

Theorem 3 (Lifted Validity)**.** *Let $\mathcal{M}$ be a well-formed CROM, $\mathcal{C}$ a constraint model compliant to $\mathcal{M}$, P a persistence annotation of $\mathcal{M}$, as well as $\mathfrak{i}$ a CROI compliant to $\mathcal{M}$ and valid with respect to $\mathcal{C}$. Then the construction $(\mathcal{N},\mathcal{D}) \coloneqq \varphi(\mathcal{M},\mathcal{C},P)$ and the restriction $\mathfrak{q} \coloneqq \psi(\mathcal{M},\psi(\mathcal{N},\mathfrak{i}))$ entails that $\mathfrak{q}$ is compliant to $\mathcal{M}$ and valid with respect to $\mathcal{C}$.*

Proof: As a consequence of Theorem 2, it also holds that $\mathfrak{p} \coloneqq \psi(\mathcal{N},\mathfrak{i})$ is compliant to $\mathcal{N}$ and valid with respect to $\mathcal{D}$. Thus, to prove Theorem 3, we need to show that $\mathfrak{q} \coloneqq \psi(\mathcal{M},\mathfrak{p})$ is compliant to $\mathcal{M}$ and valid with respect to $\mathcal{C}$, as the right side of Figure 5 indicates. Moreover, though from Definition 7 it follows, that $\mathcal{N}$ might contain natural types, which where defined as compartment types in $\mathcal{M}$. Accordingly, $\mathfrak{q} \coloneqq \psi(\mathcal{M},\mathfrak{p})$ leaves the CROI as is and only moves affected instances form $N_{\mathfrak{p}}$ to $C_{\mathfrak{q}}$, in short, transforming natural instances back to compartment instances. However, as both rules (23) and (24) ensure that only compartment types without existential parts (cf. Definition 6) will be persisted as natural type, consequently, for each compartment instance $c \in C_{\mathfrak{q}}$ with $\mathsf{type}_{\mathfrak{q}}(c) \in (\mathsf{CT}_{\mathcal{M}} \cap \mathsf{NT}_{\mathcal{N}})$, its type has no existential parts $\mathbf{ext}(\mathsf{type}_{\mathfrak{q}}(c)) = \emptyset$ and $R^c_{rt,\mathfrak{q}} = \emptyset$ for all $rt \in \mathsf{RT}_{\mathcal{M}}$. As a result, the compliance of $\mathfrak{q}$ to $\mathcal{M}$ immediately follows from the compliance of $\mathfrak{p}$ to $\mathcal{N}$. In contrast, the validity of $\mathfrak{q}$ with respect to $\mathcal{C}$ can be easily entailed from the emptiness of $R^c_{rt,\mathfrak{q}}$ (see above) applied to axioms (12) and (13). ■

In conclusion, these theorems prove that the transformation algorithm guarantees well-formedness, compliance and validity of the generated persistent context-dependent domain model and their instance. This does not only entail correct storage, but also retrieval after a system breakdown. Simply put, the transformation ensures that the persisted instance model p can also be loaded back as a valid instance model q of the runtime domain model $(\mathcal{M},\mathcal{C})$. To put it succinctly, the transformation guarantees that any persisted instance model is also valid wrt. the complete domain model and constraints.

V. Illustrative Case Study

A. Model Transformation

The *Persistency Transformation* will produce the persistence model depicted in Figure 6. As it can be seen, the ***FBS*** role type is gone, as well as ***R*** has been transformed into a natural type, because it has no existential parts that will not be persisted in the database schema. All other role types and natural types are necessary for compliance and consistency to the domain model. However, in the following we will demonstrate the algorithm step by step and explain how the algorithm extends the model and why.

We start with the source model $\mathcal{F}$ and the constraint model $\mathcal{C}_{\mathcal{F}}$ as defined in Example 1 and 2, respectively. Additionally, we assume the persistence annotation $P = (\emptyset, \{AP\}, \emptyset, \emptyset)$. Henceforth, we define the persisted CROM model $\mathcal{P} = (\mathsf{NT}_{\mathcal{P}}, \mathsf{RT}_{\mathcal{P}}, \mathsf{CT}_{\mathcal{P}}, \mathsf{RST}_{\mathcal{P}}, \mathsf{fills}_{\mathcal{P}}, \mathsf{rel}_{\mathcal{P}})$. At first, the $\mathsf{fills}_{\mathcal{P}}$ and $\mathsf{rel}_{\mathcal{P}}$ relations are constructed by the rules given in Figure 2. By applying Rule (20), $\mathsf{fills}_{\mathcal{P}}$ is extended by $(\mathrm{R}, \mathrm{FA}, \mathrm{AP})$ with all their respective player types. Next, Rule (24) must be applied, resulting in four new entries in $\mathsf{fills}_{\mathcal{P}}$. Specifically, all role types with an occurrence constraint greater than 0 are added, which applies to *FD* and *A*. Consequently, the resulting $\mathsf{fills}_{\mathcal{P}}$ is populated with the following triples:

$$\mathsf{fills}_{\mathcal{P}} = \{(R, FA, AP), (SD, FA, FD), (C, FA, FD), (P, FA, A), (S, FA, A)\}$$

Additionally, the $\mathsf{rel}_{\mathcal{P}}$ is populated by the rules (27–29), whereas rules (27) and (28) do not apply here, since neither a relationship type is annotated nor a compartment type. Hence, the relationship types *detectors* and *announcers* are collected by Rule (29), because they link role types that are already in $\mathsf{fills}_{\mathcal{P}}$ and have a cardinality constraint of at least 1. As a result, the $\mathsf{rel}_{\mathcal{P}}$ is populated as following:

$$\mathsf{rel}_{\mathcal{P}}(detectors, FA) = (FD, AP)$$
$$\mathsf{rel}_{\mathcal{P}}(announcer, FA) = (AP, A)$$

Out of these two relations, the corresponding type sets are created by employing rules (15–18) and are the following:

$$\mathsf{CT}_{\mathcal{P}} = \{FA\} \qquad \mathsf{RST}_{\mathcal{P}} = \{detectors, announcers\}$$
$$\mathsf{RT}_{\mathcal{P}} = \{AP, FD, A\} \qquad \mathsf{NT}_{\mathcal{P}} = \{SD, C, P, S, R\}$$

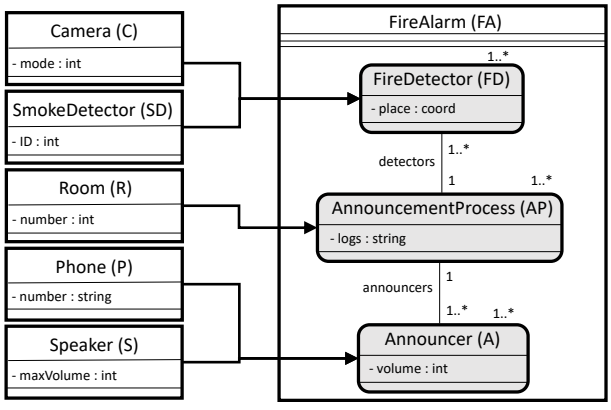

Figure 6. The resulting persisted CROM model.

```
1 CREATE CT FA (ID Int PRIMARY KEY);
2 CREATE RT FD (place Coord,ID Int PRIMARY KEY)
    ↪ PLAYED BY (SD,C) PART OF FA WITH OCC (1,*);
3 CREATE RT A (loud Int,ID Int PRIMARY KEY)
    ↪ PLAYED BY (P,S) PART OF FA WITH OCC (1,*);
4 CREATE RT AP (logs Text,ID Int PRIMARY KEY)
    ↪ PLAYED BY (R) PART OF FA WITH OCC (1,*);
5 CREATE RST detectors CONSISTING OF
    ↪ FD BEING (1..*) AND AP BEING (1..1);
6 CREATE RST announcers CONSISTING OF
    ↪ AP BEING (1..1) AND A BEING (1..*);
```

Listing 1. RSQL Data Definition Language Statements to Create $\mathcal{N}$.

After inductively defining the persisted CROM $\mathcal{P}$, the constraint model $\mathcal{D}_\mathcal{P} := \{\text{occur}, \text{card}\}$ is computed from $\mathcal{C}_\mathcal{F}$ by applying (30) and (31). This results in the following partial function definitions, where the feedback cardinality constraint is removed.

$$\text{occur}_{\mathcal{D}_\mathcal{P}} := \{\text{FA} \rightarrow \{(1..\infty, \text{FD}), (1..\infty, \text{AP}), (1..\infty, \text{A})\}\}$$
$$\text{card}_{\mathcal{D}_\mathcal{P}} := \{(\text{detectors}, \text{FA}) \rightarrow (1..\infty, 1..1), (\text{announcers}, \text{FA}) \rightarrow (1..1, 1..\infty)\}$$

Finally, the CROM $\mathcal{P}$ and constraint model $\mathcal{D}_\mathcal{P}$ is used to create a database schema and persist valid instances of the fire alarm model.

B. Database Schema

As aforementioned, the persistence CROM model $\mathcal{P}$ will be stored in an RSQL database that preserves the context-dependent semantics and information [9]. RSQL is able to directly store CROM-based models, thereby avoiding the need to transfer runtime semantics onto traditional database semantics. Listing 1 lists the RSQL data definition language statements to create the schema for $\mathcal{P}$. Please note, for the sake of brevity, we assume that all natural types have already been created. In practice, this schema can be generated from a given CROM and corresponding constraint model [12]. Additionally, queries directly leverage the context-dependent information during query processing. This allows the SaS to consistently persist, i.e., insert, update, delete and query, parts of the context-dependent knowledge base into the RSQL database system.

VI. Related Work

The Unified Modeling Language (UML) lacks expressive power to model context-dependent domains and while some approaches extended UML in this regard [13][14][15][16], their semantics is usually more ambiguous.

The *Metamodel for Roles* [14] tries to be the most general formalization of context-dependent roles. Similar to CROM, it distinguishes between *Players*, *Roles*, and *Context* on the type and the instance level. Yet, the metamodel is too general to be useful, because the sets of entities are not required to be disjoint (on both the type and instance level) [5]. Similarly, the *Information Networking Model* (INM) [17] is a data modeling approach designed to overcome the inability of data models to capture context-dependent information. While this approach allows to model nested *Contexts* with attributes containing *Roles*, the various kinds of relations cannot be constrained [17]. By extension, the few hybrid models are presented in the *HELENA* approach [16]. It features *Ensembles* as compartments to capture a collaborative task by means of roles that are played by *Components*. In particular, *HELENA* provides formal definitions for both type and instance level, as well as an operational semantics based on sets and labeled transition systems [16]. Furthermore, *HELENA* only supports occurrence constraints on roles, and no cardinality constraints on relationships. In contrast to them, CROM has a well-defined, formal semantics [5], has graphical editing support [12][18] and supports reasoning [6]. A more detailed comparison of context-dependent modeling and programming languages can be found in [19][8].

To persist context-dependent domain models, several approaches are proposed. Generally, these can be classified by their implemented technique. In detail, we distinguish the following techniques: (i) mapping engines, (ii) persistent programming languages, and (iii) full DBS implementation.

Firstly, mapping engines, as technique to bridge the transient application and persistent database world, are well-known from the object-relational impedance mismatch [20]. However, mapping engines like DAMPF [21], ObjectTeams JPA [22], or ConQuer [23] store the transient runtime information in traditional database systems, which does not preserve the context-dependent semantics and thus, cannot be leveraged for storing or querying. Additionally, in multi-application scenarios the database system cannot ensure the metamodel constraints, because the mapping engines hide these constraints from the database system. Secondly, persistent programming languages like the Dynamic Object-Oriented Database Programming Language with Roles (DOOR) [24] or Fibonacci [25] unify the transient application world with the persistent database world. Unfortunately, these approaches help in single application scenarios only, because the persistent data storage is part of the individual applications and thus, not shared with other applications. Especially for self-adaptive software systems, several applications need to share their information, not only directly, but also using a persistent database system, which makes persistent programming language inappropriate for such systems. Finally, the last class of approached comprises the integration of contextual semantics with a database system. The conceptual model of INM has been implemented into a database system and features a dedicated query language [17], [26]. However, INM misses the constraints of relations between the classes. However, none of them considers to partially persist a context-dependent domain model while ensuring its well-formedness and validity.

VII. Conclusion and Future Work

SaS rely on context-dependent domain models at runtime to reason about their environmental situations and system state. Considering persistence, not all information captured in the knowledge base needs to be stored persistently. Yet, finding a valid partial model that is consistent with the original domain model is a daunting task, as the model's well-formedness and validity depends on the well-formedness rules, constraints, and instances of the model itself and also changes after an adaptation of the original domain model. To remedy this, we proposed the *Persistency Transformation* φ, an algorithm that computes a minimal valid partial runtime model given a set of annotated model elements. In fact, we employed CROM, a dedicated, formalized modeling language for context-dependent domain models. Based on CROM, we were able to show that our transformation ensures both the

 ISBN: 978-1-61208-706-1

well-formedness of the partial CROM model (including the compliance of the partial constraint model). Moreover, we showed that arbitrary valid instances of a CROM model can be automatically restricted (Instance Restriction ψ) with respect to the partial CROM model, ultimately, yielding valid partial instances of the partial CROM model. Furthermore, we proved that such a valid partial instance can be lifted (Instance Restriction ψ) to the complete CROM model retaining its validity and compliance. Conversely, we can ensure, in case of restoring, such context-dependent information will result in a valid and well-formed runtime model. While the proposed transformation is independent of the underlying database system, our case study employed RSQL, a dedicated database system for storage and retrieval of context-dependent knowledge bases. Regardless of the underlying database system, our approach does not only automate finding a viable partial model, but also guarantees the consistency of this partial model and runtime instances. This reduces the requirements for databases persisting context-dependent knowledge bases of SaS by reducing the memory footprint and avoiding unintended system behavior after a system restore. Notably, the presented algorithm relies on the formal underpinning of CROM and thus might not be applicable to other context-dependent domain models. Moreover, the performance and complexity of the algorithm was not considered in this work. Furthermore, the presented approach explicitly excludes *role groups*, as they can express arbitrary propositional logic formulas [5], thus significantly increasing the expressiveness of CROM.

In the future, we want to fully evaluate the feasibility of our approach by developing a reference implementation and evaluating its performance in more realistic application scenarios. Moreover, we want to introduce role groups by extending the *Persistency Transformation* and investigating the resulting time complexity in the presence of arbitrary role groups. Ultimately, we want to set up a persistence framework for context-dependent domain models. Such an integrated framework would combine modeling the SaS graphically, adding persistence annotations, computing the valid partial persistence model, as well as creating the corresponding RSQL schema statements. Finally, the designed SaS could directly utilize this framework to consistently persist partial context-dependent domain models.

ACKNOWLEDGMENT

This work has been funded by the German Research Foundation within the Research Training Group "Role-based Software Infrastructures for continuous-context-sensitive Systems" (GRK 1907).

REFERENCES

[1] F. D. Macías-Escrivá, R. Haber, R. del Toro, and V. Hernandez, "Self-adaptive Systems: A Survey of current Approaches, Research Challenges and Applications," Expert Systems with Applications, vol. 40, no. 18, 2013, pp. 7267–7279.

[2] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," Pervasive and Mobile Computing, vol. 17, 2015, pp. 184 – 206.

[3] C. Hoareau and I. Satoh, "Modeling and processing information for context-aware computing: A survey," New Generation Computing, vol. 27, no. 3, May 2009, pp. 177–196.

[4] T. Jäkel, M. Weissbach, K. Herrmann, H. Voigt, and M. Leuthäuser, "Position Paper: Runtime Model for Role-Based Software Systems," in International Conference on Autonomic Computing, ICAC. Wuerzburg, Germany: IEEE, Jul. 2016, pp. 380–387.

[5] T. Kühn, S. Böhme, S. Götz, C. Seidl, and U. Aßmann, "A Combined Formal Model for Relational Context-Dependent Roles," in International Conference on Software Language Engineering. ACM, 2015, pp. 113–124.

[6] S. Böhme and T. Kühn, "Reasoning on context-dependent domain models," in 7th Joint International Conference on Semantic Technology. Springer, November 2017, pp. 69–85.

[7] Y. Brun et al., "Engineering self-adaptive systems through feedback loops," in Software engineering for self-adaptive systems. Springer, 2009, pp. 48–70.

[8] T. Kühn, "A family of role-based languages," Ph.D. dissertation, Technische Universität Dresden, 2017.

[9] T. Jäkel, T. Kühn, H. Voigt, and W. Lehner, "Towards a Role-Based Contextual Database," in 20th East European Conference on Advances in Databases and Information Systems. Springer International Publishing, 2016, pp. 89–103.

[10] T. Kühn, "Persistence transformation," 2019. [Online]. Available: https://github.com/Eden-06/formalCROM/tree/master/persistency

[11] T. Jäkel, "Role-based data management," Ph.D. dissertation, Technische Universität Dresden, 2017.

[12] T. Kühn, K. Bierzynski, S. Richly, and U. Aßmann, "Framed: Full-fledge role modeling editor (tool demo)," in International Conference on Software Language Engineering. ACM, 2016, pp. 132–136.

[13] Q. Z. Sheng and B. Benatallah, "ContextUML: a UML-based Modeling Language for Model-driven Development of Context-aware Web Services," in International Conference on Mobile Business. IEEE, 2005, pp. 206–212.

[14] V. Genovese, "A Meta-Model for Roles: Introducing Sessions," in 2nd Workshop on Roles and Relationships in Object Oriented Programming, Multiagent Systems, and Ontologies, 2007, pp. 27–38.

[15] G. Guizzardi and G. Wagner, "Conceptual Simulation Modeling with onto-UML," in Winter Simulation Conference. Winter Simulation Conference, 2012, pp. 5:1–5:15.

[16] R. Hennicker and A. Klarl, "Foundations for Ensemble Modeling–The Helena Approach," in Specification, Algebra, and Software. Springer, 2014, pp. 359–381.

[17] M. Liu and J. Hu, "Information Networking Model," in International Conference on Conceptual Modeling. Springer, 2009, pp. 131–144.

[18] T. Kühn, K. I. Kassin, W. Cazzola, and U. Aßmann, "Modular feature-oriented graphical editor product lines," in 22th International Software Product Line Conference. Gothenburg, Sweden: ACM, 10th-14th of September 2018, pp. 76–86.

[19] T. Kühn, M. Leuthäuser, S. Götz, C. Seidl, and U. Aßmann, "A Metamodel Family for Role-based Modeling and Programming Languages," in 7th International Conference on Software Language Engineering. Springer, 2014, pp. 141–160.

[20] C. Ireland, D. Bowers, M. Newton, and K. Waugh, "A Classification of Object-relational Impedance Mismatch," in Advances in Databases, Knowledge, and Data Applications. IEEE, 2009, pp. 36–43.

[21] S. Götz, "Dampf - Dresden Auto-Managed Persistence Framework," Technische Universität Dresden, Diploma Thesis, 2010.

[22] O. Otto, "Development of a Persistence Solution for Object Teams based on the Java Persistence API (dt: Entwicklung einer Persistenzlösung für Object Teams auf Basis der Java Persistence API)," Technische Universität Berlin, Diploma Thesis, 2009.

[23] A. Bloesch and T. Halpin, "Conceptual Queries using ConQuer-II," in International Conference on Conceptual Modeling. Springer, 1997, pp. 113–126.

[24] R. Wong, L. Chau, and F. Lochovsky, "A Data Model and Semantics of Objects with Dynamic Roles," in International Conference on Data Engineering. IEEE, Apr 1997, pp. 402–411.

[25] A. Albano, G. Ghelli, and R. Orsini, "Fibonacci: A Programming Language for Object Databases," The VLDB Journal, vol. 4, no. 3, 1995, pp. 403–444.

[26] J. Hu, Q. Fu, and M. Liu, "Query Processing in INM Database System," in Web-Age Information Management. Springer, 2010, pp. 525–536.

 ISBN: 978-1-61208-706-1