A Preliminary Framework for Strategic and Compliance Monitoring

Evellin Cardoso
KRDB Research Centre
Free University of Bozen-Bolzano
Bolzano, Italy
ecardoso@unibz.it

Marco Montali
KRDB Research Centre
Free University of Bozen-Bolzano
Bolzano, Italy
montali@inf.unibz.it

***Abstract*—The complexity of modern organisations drives the usage of enterprise architectures models and execution data as suitable tools to manage such complexity. To promote a successful enterprise management, this paper proposes the 2QUEN framework that uses enterprise architecture models to monitor the achievement of enterprise goals. In order to represent the relevant architectural domains, 2QUEN uses abstractions like *goals*, *business processes* and *normative primitives*. In order to gather data to monitor goal achievement, 2QUEN proposes the first steps of a methodology to access and query data of a legacy relational database with the purpose of querying norms states. The methodology starts from normative primitives (commitments, authorisations, prohibitions and powers) and uses the architecture provided by the 2-level version of OBDA [2] (2OBDA), a framework from the Semantic Web community, to semantically represent normative primitives and their states, and subsequently map those norms to the underlying legacy data.**

I. Introduction

The management of organisations involves a high level of complexity since it aggregates several knowledge domains (motivational, behavioural, normative, etc.). Each of these domains may be influenced by potentially conflicting quality factors which affect organisation's overall performance. In order to support enterprise governance, organisations typically rely on enterprise architectures as a valuable tool to represent, balance and prioritise such factors. In addition to architectural models, the use of data extracted from information systems (e.g. Enterprise Resource Planning (ERP), Customer Relations Management (CRM), Supply Chain Management (SCM), Executive Information Systems (EIS), Business Intelligence (BI), Data Warehouse and OLAP tools, etc.) is also adopted as a promising tool for extracting insights and trends from running data during the decision-making process of enterprise management.

Achieving a coherent enterprise management by means of architectural models aligned with execution data poses a twofold challenge. The first challenge is about finding the right set of constructs for the *representation* of each architectural perspective to enable a meaningful enterprise management. In particular, the representation challenge amounts to find the suitable architectural perspectives, their right set of constructs together with suitable interrelations. The second challenge is about the *actual data gathering and reporting* during enterprise runtime execution, i.e., how to meaningfully connect execution data with enterprise architecture models.

In order to solve both challenges, enterprise architecture approaches usually focus on one enterprise perspective (e.g. motivational, behavioural) and propose methods/techniques to infer the runtime properties of the elements in such perspective. For example, Maté et. al [1] infer goal achievement by linking goal models with Key Performance Indicators (KPIs) calculated with data stemmed from Data Warehouses. A similar approach is adopted by the BIM framework [2], whereas Zaki et. al [3] extract KPI values from execution logs on the basis of process models. Alberti et. al. [4] link normative primitives and business process models with execution data, Hernández-Julio et. al. [5] propose a framework that links software architecture models with computational intelligence algorithms, whereas Veneberg et. al. [6] propose a method for linking architectural models with data. From the information systems side, the main issue is that the data stored during the operation of the enterprise system are typically at a low level of abstraction, and do not directly refer to the business concepts represented by enterprise architecture models, such as goals, business processes, norms, etc [6], [7]. This poses the question on how to *access and query* such low-level data so as to obtain relevant information about the evolution of enterprise primitives (e.g., in which state a given business process is, whether a business goal has been achieved or not, whether a normative primitive has been violated or not, etc). Such a question is not peculiar to architectural models, but is in fact pervasive and relates, in general, to the problem of accessing and querying legacy data using conceptual, high-level abstractions that are in line with the knowledge and vocabulary of human experts [7].

In order to overcome both challenges, this paper proposes the 2QUEN framework that uses holistic enterprise architecture models to monitor the achievement of enterprise goals. In order to tackle the *representation* challenge, 2QUEN adopts a variety of abstractions like *goals*, *business processes* and *normative primitives* inherited from the SIENA framework [8], a framework for the representation of strategic enterprise architectures. Other types of normative primitives (commitments, authorisations, prohibitions, and powers) are also extracted from Custard [9], a framework for the specification and

2325-6605/19/$31.00 ©2019 IEEE
DOI 10.1109/EDOCW.2019.00024

monitoring of the evolution of normative primitives along their lifecycles. 2QUEN is structured in terms of a *goal view* that captures motivational elements, whereas the *behavioural view* captures the business process control-flow expressed in terms of normative primitives.

In order to overcome the *data gathering* challenge, our approach proposes the first steps of a methodology to access and query data of a legacy relational database with the purpose of querying normative primitives states. Our methodology has two starting points, the Custard framework and the *Ontology-Based Data Access* (OBDA) paradigm [10], [11] (also known as Virtual Knowledge Graphs). Custard captures norms equipped with a lifecycle that captures the states in which these norms may be, and the transitions between states, triggered by punctual events. Norm instances states are maintained in a relational information store, and they can be manipulated through expressions that, on the one hand provide a declarative and very compact specification, but on the other hand, presents a number of shortcomings. First, Custard adopts a relational representation of norms, a long advocated feature [12], but still supports only limited forms of joins based on equality. This represents a potentially serious drawback especially when accessing legacy databases, where the structure of the data is not tailored towards the manipulation of norms. Furthermore, the Custard event algebra does not support explicit expirations or violations of norms, determined by the semantics of the domain of interest.

Since Custard representational form does not adequately reflect the semantics of the domain of interest and presents shortcomings with respect to data access, we take inspiration on the well-established paradigm of *Ontology-Based Data Access* (OBDA) [10], [11] to overcome such limitations. OBDA is based on the idea that end-users are exposed to the data through a high-level, conceptual representation of the domain of interest, given in the form of an *ontology*, that abstracts away low-level details about the organisation and storage of the data itself. The ontology is expressed in terms of a domain vocabulary (of concepts and relations) that is familiar to the users, and moreover it is able to capture complex interrelationships between the conceptual elements of the domain by means of logical axioms. The ontology is linked to the underlying legacy data through a declarative *mapping* [10], which intuitively describes, by means of correspondences between queries, how the conceptual elements of the ontology are populated from the data. The mapping is exploited by an OBDA-system to automatically translate user queries formulated over the ontology into SQL queries over the relational data source(s), taking also into account the domain semantics encoded by the ontology axioms so as to enrich the set of provided answers [11]. Different variants of OBDA systems exist and in this paper, we use a 2-level version in which the conceptual representation of the domain has two layers of ontologies. OBDA has been applied successfully within the Semantic Web community in a number of scenarios where the access to data poses a major challenge [7].

In our methodology, starting with the normative primitives from the 2QUEN behavioural view and using the architecture provided by the 2-level version of OBDA [13] (2OBDA), we semantically represent normative primitives and their states, and map them to the underlying legacy data. The distinctive feature of 2QUEN is that mappings are not arbitrarily expressed by the modeller, but are instead automatically synthesised starting from a specification that explicitly indicates how the constitutive components of a normative primitive, as well as its lifecycle, can be reconstructed from the legacy data.

In particular, our long-term research goal is to monitor the achievement of normative primitives along their lifecycles in the presence of complex real-world data with the purpose of monitoring the achievement of enterprise's goals. As such, our work gives a first step towards a framework for normative and performance monitoring. More specifically, we provide the following contributions:

- A novel framework for the representation and monitoring of enterprise goals, business processes and normative primitives that relies on 2OBDA;
- A methodology for lifting data stored in a relational system to a fully semantic framework for the representation and querying of norms. By querying the norm states that govern the interactions among multiple actors in the execution of business process, the framework has the potential to verify norms and business process states and indirectly infer the achievement of enterprise goals.

The rest of the paper is structured as follows: Section II provides the research baseline of our work that includes the Custard framework, SIENA framework and the (2)OBDA paradigm. Section III presents the *Goal* and *Behavioural Views* of the 2QUEN framework, while Section IV presents a sequence of methodological steps that start from the normative primitives of the 2QUEN behavioural view and derive the fully semantic representation of norms based on 2OBDA architecture. As a final outcome, this methodology semantically links 2QUEN normative primitives with the relational schema of the legacy database. Finally, in Section V we provide conclusions and outline future work.

II. Baseline

This section introduces the three main preliminary frameworks on which we rely, namely the SIENA framework, the 2QUEN normative primitives from the Custard language and the Ontology-Based Data Access (OBDA) paradigm.

A. Custard Framework

Custard [9] is a framework for the specification of information-based normative primitives (called simply *norms* hereafter), and the retrieval of the states of such primitives from a relational information store.

A *norm schema* η contains normative (norm) types that socially relate interacting parties and implicitly regulate their interactions: commitments, authorisations, prohibitions, and powers. Each norm instantiates its schema creating a contractual relationship between two agents, an *expector* and an *expectee*, where the expectee is accountable to the expector for

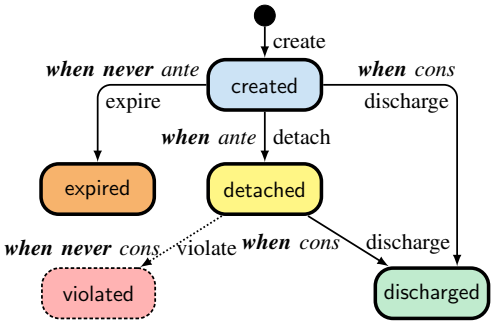

Figure 1: Lifecycle of norm types (from [9]). The violated state exists only for prohibition and commitment.

the satisfaction of the expectation implied by the contractual relationship.

More technically, each norm type comes with a canonical *lifecycle* that captures the different states in which a norm of that type can be, as well as the possible transitions from one state to the other, and the conditions that induce those transitions. The lifecycles of the different Custard norm types are depicted in Fig. 1. The idea behind such lifecycles is borrowed from the well-established notion of commitment machine [14]. Specifically, once a norm is created, it corresponds to a conditional expectation that comes into force (i.e., is detached to an actual expectation) when the expectee brings about the *antecedent* condition associated to the norm. If the course of execution is so that such a condition cannot be realised, the norm expires. A created or detached norm becomes discharged when its associated *consequent* condition is made true by the expectee agent. The achievement of the consequent indeed witnesses that the expectation associated to the norm has been fulfilled. In the case of prohibitions and commitments, a detached norm may become violated if the course of execution is so that the consequent can never be achieved.

In Custard, conditions are specified with an algebra that employs punctual events as basic building blocks. An *information schema I* records event instances that may happen in the system (e..g, register, sign up, allow to disclose information, etc.). This information schema is specified as a relational database in which each event corresponds to a database relation containing a key and a timestamped column that store the occurrence time of event instances.

Example 1 (Inspired from [9]). We use a real-world scenario in the scope of Health Level Seven (HL7) that consists of a set of standards for prescribing protocols of communication for exchanging clinical and administrative data among software applications used by several healthcare providers. The following information schema captures four event types related to the request and access to patient data by a sanitary organisation, where the primary key of each relation schema is underlined:

- $Allowed(pid, hid, \underline{discid}, tpid, t)$ – at time t, patient pid accepts to disclose her data to third party $tpid$ through health vault provider hid. The id value used for $discid$ hence constitutes a "disclosure token" for the patient data.
- $SentCred(hid, tpid, \underline{discid}, t)$ – at time t, health vault provider hid sends access credentials to third party $tpid$ using the disclosure token $discid$. The field $discid$ is a foreign key referencing the $Allowed$ relation. Since it is also a primary key for $SentCred$, credentials can be sent at most once for a given entry in $Allowed$.
- $ReqData(tpid, hid, \underline{reqid}, discid, t)$ – at time t the third party entity $tpid$ requests patient data to the health vault provider hid, referring to the credentials obtained via $discid$ (which is in fact a foreign key referencing the $Allowed$ relation).
- $Accessed(tpid, hid, \underline{reqid}, discid, t)$ – at time t the third party entity $tpid$ accesses, via health vault provider hid, the patient data related to the disclosure token $discid$ (which is a foreign key referencing the $Allowed$ relation). Similarly to the relation schema used for $SentCred$, here the field $reqid$ simultaneously acts as primary key for the $Accessed$ relation, and as a foreign key referencing the $ReqData$ relation. This guarantees that at most one access per request is actually recorded.

The different event tables have to be understood as log tables recording events triggered within the enterprise system. The *effect* induced by each of such events depends on the actual data. For example, credentials are effectively delivered only if the ids of the health vault provider and of the third party actually coincide with those obtained by inspecting the entry of the $Allowed$ relation with matching $discid$. Similarly, data are actually accessed only if the access request is done consistently with the parties involved in the corresponding request, and with the right disclosure token. As we will see, these implicit semantic constraints can be explicitly captured in 2QUEN when relating the information system to the ontology. ◁

In order to compute the progression of norm instances across the several stages of its lifecycle, each stage of a norm instance in the norm schema η points to the relations of the information schema I. Furthermore, the norm schema also supports the specification of complex events expressions over database relations involving logical operators (AND, OR, EXCEPT), aggregation operators (SUM), set operators (COUNT, MIN, MAX, AVG), relative time intervals within which events should occur ([startEvent, endEvent]), and nested norms. In addition, an ad-hoc form of negation is used to determine when an event expression never holds (e.g., the negation of an event that should occur within a certain deadline corresponds to checking that the event did not occur before the deadline expiration time).

Specifically, the progression of a specific norm is defined in Custard by qualifying the *create*, *detach*, and *discharge* transitions with corresponding event expressions that instantiate the expecter and expectee agents, and the corresponding transition timestamps. The *expire* and *violate* transitions are then implicitly qualified by negating the antecedent and consequent event expressions respectively associated to the detach and discharge transitions.

Example 2 (Inspired from [9]). Consider the information schema of Example 1. The following Custard specification defines an authorization norm regulating the possibility of accessing sensible patient data:

authorization	$DisclosureAuth\ tpid$ **by** hid
create	$SentCred$
detach	$ReqData$
discharge	$Accessed[ReqData + 1, ReqData + 10]$

In particular, the specification indicates that a disclosure authorization is created by hid for $tpid$ at time t whenever there exists an entry in the $SentCred$ relation relating hid, $tipd$, and t. A consequent detach of the authorization occurs at a consequent time t_2 whenever an entry in $ReqData$ exists with time t_2 and matching the common fields of $ReqData$ and $SentCred$. Finally, the authorization is discharged if a consequent matching entry in $Accessed$ exists at a time that lies between 1 and 10 time units after t_2.

As dictated by the notion of negation used in Custard event patterns, the authorization is considered to be violated if, upon detachment, no access entry exists between t_2+1 and t_2+10.◁

The example shows that Custard is extremely compact and declarative when specifying the lifecycle of norms, but also highlights its main two shortcomings:

- *Lack of flexibility in querying relations*. Custard assumes that when relating relations to each other, joins are implicitly done based on equality of field names. A more general approach relying on the full SQL query language is needed to handle the situation where the underlying information schema is a legacy one, and consequently requires the power of complex queries so as to extract the required information. Already in Example 1, one would need more powerful queries than those supported by Custard to retrieve the disclosure tokens that have been potentially misused. For example, the SQL query

  SELECT $r.discid$ **FROM** $ReqData\ r, SentCred\ c$
  WHERE $r.discid = c.discid$ **AND** $r.tpid \neq c.tpid$

  can be used to retrieve potentially misused disclosure tokens, i.e., tokens used to request data by a third party that does not match the one to which the credentials for that token have been sent.
- *Lack of explicit expirations and violations*. While the Custard event algebra has the nice property of being able to express implicit expiration/violation of a norm, the specification of norms does not allow to complement such implicit transitions with explicit expirations/violations determined by the semantics of the domain under study, with conditions that cannot be simply ascribed to the "absence" of the norm antecedent or consequent. For example, one could use a variant of the SQL query shown in the previous bullet so as to induce a violation of an authorization for which the disclosure token has been potentially misused.

Given our final goal to create a framework for normative and performance querying on top of legacy information systems, Custard has been chosen due to its ability to specify different types of normative primitives (and check their status using event data), but we lift it to a full, semantical setting where the aforementioned key limitations are resolved. More specifically, we would be interested in querying the framework to monitor performance question such as: 1) How many patients had access to the system in 2018? 2) How many disclosure authorisations have been granted by health vault provider hID1? 3) What is the average time for the overall disclosure process? and some compliance questions such as: 4) How many commitments have been discharged last month? 5) Which third parties have violated a disclosure prohibition?

Since our framework formalizes compliance and performance questions in terms of some conceptual elements of the SIENA framework, the next section introduces such conceptual elements.

B. SIENA Framework

SIENA [8] is a framework for the representation of strategic enterprise architectures and automated reasoning with such models.

The SIENA language [15] provides abstractions for capturing enterprise's *motivational elements* (i.e. goals of different shades like mission, vision, strategic, tactical and operational goals) and their connections with *behavioural elements* (i.e., operations, business processes and activities) through which they are operationalised. The language is also accompanied by modelling guidelines and a formal refinement approach for the construction of its models.

Besides the SIENA language, the SIENA framework also contains a business process language called Azzurra [16] which is founded on the primitives of *commitments* and *protocols* for the specification of the social perspective of business processes. Methodological guidelines also allow the transition from motivational to behavioural perspectives, by supporting the design of business processes' control-flow from operational goals. This business process' control-flow is then specified in terms of Azzurra's modelling primitives.

SIENA contributes to the enterprise modelling field by proposing a framework founded on different notions of motivational and behavioural concepts and a top-down strategic planning approach for the generation of best set of business processes to achieve enterprise strategic goals. In this paper, SIENA provides the conceptual foundation for the development of the 2QUEN framework that aims to monitor compliance to regulations and performance of business processes.

C. Ontology-Based Data Access (OBDA) Paradigm

The Ontology-Based Data Access (OBDA) paradigm consists of a novel philosophy of conceiving information systems in which conceptual schemas (ontologies) are used as an intermediate layer for accessing and querying data stored in legacy information systems [13], [11]. The ultimate goal of OBDA is to provide a vocabulary for end-users based on domain notions, thus allowing a transparent access to legacy

information systems, as it abstracts away from implementation details on how data is concretely stored.

Formally, an OBDA specification consists of a $\mathcal{S} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, where $\mathcal{R}$ is a *relational database schema*, $\mathcal{T}$ is a conceptual schema (*domain schema*) and $\mathcal{M}$ consists of a set of *mapping assertions*. The success achieved with the application of OBDA in a plethora of application domains [13] led researches to develop the 2OBDA approach that extends the OBDA paradigm by accommodating a higher-level schema (*upper-schema* $\mathcal{T}'$) that further abstracts the domain knowledge. This *upper schema* $\mathcal{T}'$ is then declaratively mapped to the *domain schema* $\mathcal{T}$ by means of a *schema transformation* that consists of a set $\mathcal{N}$ of *transformation rules*. Fig. 2 depicts a schematic representation of the 2OBDA architecture.

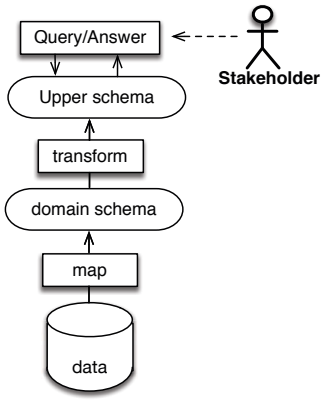

Figure 2: Schematic representation of 2OBDA architecture (adapted from [13])

In 2OBDA, both domain and upper schemas are represented as UML class diagrams which is the concrete language for conceptual data modelling with its logic-based encoding in terms of OWL2QL [13]. The 2OBDA approach is supported by a set of tools that allow one to model the domain schema, the upper-schema, and the transformation rules between the two schemas. The latter are represented as annotations of the domain schema with the elements of the upper-schema.

Similarly to the OBDA approach that provides domain-oriented vocabulary for querying data sources, 2OBDA has the same advantages, as data sources can be queried in terms of upper schema abstractions (Fig. 2). One example of the application of 2OBDA [13] is depicted in the context of service management, with the upper schema capturing business relationships between an organization and its external stakeholders and allowing one to inspect the states of contractual relationships during the enactment of business exchanges.

III. 2QUEN Framework

This section introduces the elements of the 2QUEN framework that are basically organised into two views. The Goal View expresses motivational elements, while the Behavioural View expresses the required behaviour to achieve such motivational elements. As our ultimate goal is to answer the compliance and performance questions settled in Sec. II-A, the 2QUEN framework uses suitable elements from the SIENA framework (Sec. II-B) to answer such questions, adding more elements when necessary.

A. Goal View

This section presents the Goal View of the 2QUEN framework, by depicting the different shades of motivational elements required to answer our compliance and performance questions.

The core idea of the SIENA framework is to capture enterprise's motivational elements, ranging from high abstraction elements, such as mission (why does organisation exists?), strategic goals (which strategies does the organisation adopt to tackle competitors?), until reaching low abstraction elements, such as tactical goals (which resources do we use to attain our strategy?) and operational goals (how enterprise's strategy is realised by behavioural elements?). The methodological guidelines and the formal refinement approach for strategic goals both embody a top-down managerial process of deciding the best set of business processes (Business Process Architecture (BPA)) that achieve enterprise's strategic goals.

Since Operational goals are directly connected to business processes in SIENA, we introduce such concept here as follows:

Operational Goals [8]. Operational goals correspond to the results that must be achieved in the course of performing enterprise's operations, representing the milestones to be achieved by each operation. They are refined into Role Goals and Business Process Goals, depending on who is responsible for their achievement. Role Goals specify the results to be achieved by *roles and individuals* while performing their daily work, whereas Business Process Goals represent the final state to be achieved by a *business process*. The concept of Business Process is explained in Sec. III-B.

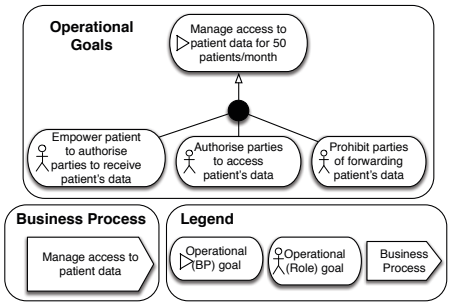

Figure 3: Operational Goals from HL7 Privacy Consent Scenario

Fig. 3 depicts the operational goals relative to the HL7 privacy consent scenario introduced in Sec. II-A using the SIENA operational goal concept. The model has been elaborated using the natural language description of the domain. In this figure, the root goal of the process is to "Manage access to patient's personal data for 50 patients/month". Since this goal is a business process goal, this goal also reflects the final state to be achieved by the "Manage access to patient data" business process. "Manage access to patient's personal data

for 50 patients/month" is AND-refined into three (operational role) goals, namely "Empower patient to authorise parties to receive patient's data", "Authorise parties to access patient's data" and "Prohibit patients of forwarding patient's data" that are assigned to the roles that perform the business process.

Besides AND-refinements, operational goals may be also OR-refined to depict alternatives and positive and negative (+/-) contributions may be also specified to depict partial influences in the achievement of operational goals (not depicted in Fig. 3). Given that the SIENA methodology focus on the decomposition of the root process goals into the responsibilities that need to be achieved by the roles, the operational goals refinement methodology in fact consists of designing the business process in terms of the intermediate milestones that need to be achieved.

In contrast, rather than using goals for designing business processes, the 2QUEN framework intends to use goals for monitoring the BPA runtime behaviour. For that, the 2QUEN framework introduces the concept of (Process) Performance Indicator that is required for monitoring operational goals:

Key Performance Indicator (KPI) (inspired by [17]). Represents a quantifiable metric that characterises the performance of some entity (e.g., organisation, role, business process, product, etc.) towards the achievement of some goal. Formally, it is defined as an 7-tuple KPI = (identifier, business goal, relatedTo, algorithm, target value, upper threshold, lower threshold, analysis period) 1) The *Identifier* represents an unique identifier for the indicator, 2) The *Business Goal* captures the business goal that the KPI quantifies, 3) The *RelatedTo* refers to the architectural element (e.g. process) that needs to have some attribute measured to quantify the KPI value 4) The *Algorithm/mathematical formula* defines how to measure the KPI in terms of some architectural element (e.g. role, business process), 5) The *Target Value* indicates the expected value for the KPI in a given period of time, 6) The *Upper Threshold* and *Lower Threshold* define a corridor of intended performance, 7) The *Analysis Period* corresponds to the period of time where such target must be achieved.

To exemplify the use of process performance indicators in our approach, performance question 1 (How many patients had access to the system in 2018?) of Sec. II-A is here modelled as indicators.

- *Identifier*: numbPat
- *Business Goal*: "Manage access to patient data for 50 patients/month"
- *RelatedTo*: "Manage access to patient data" business process
- *Mathematical formula*: $Measure_{numbPat} = \sum_{i=1}^{n} i$, where n is the number of process instances
- *Target Value* (TV): 50 patients (the target from the operational goal). Although this target has been acquired from the operational goal target, since the performance question does not specify a target, we might have left as undefined if the manager is solely interested in discovering the current number of patients served by the business process,
- *Upper Threshold*: 40 patients, *Lower Threshold*: 60 patients
- *Analysis Period*: 1 month

B. Behavioural View

This section describes Behavioural View of the 2QUEN framework, by depicting the legislative and behavioural elements required to answer our compliance and performance questions.

Within the SIENA framework, the Azzurra language represents business processes from a social standpoint, abstracting away the representation of operational and technical details. In order to perform this shift, Azzurra adopts the notion of *social commitment* among actors as the fundamental business process abstraction. *Commitment protocols* are then used to represent business processes as a protocol in which commitments explicitly capture the social responsibilities of actors towards each other. Formally, a (social) commitment [18] c(x,y,p,q) is a promise with contractual validity made by an agent x (debtor) (the agent who is committed) to another agent y (creditor) (the agent who receives the commitment) that, if proposition p is brought about (antecedent), then proposition q will be brought about (consequent).

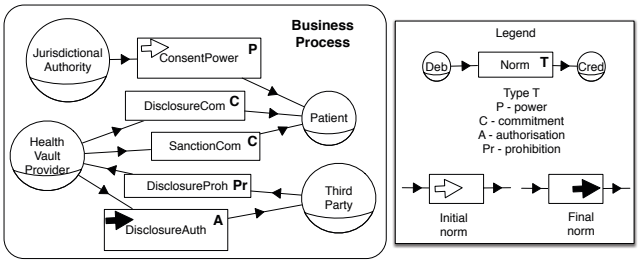

Figure 4: Azzurra Specification (Business Process) from HL7 Privacy Consent Scenario

By following SIENA's top-down methodology, the design of business process's control-flow starts with the Operational Goals from previous section and derives the elements required for the specification of business processes in Azzurra.

Fig. 4 depicts the Azzurra model relative to internal details of the "Manage access to patient data" business process from Fig. 3. This Azzurra model has been developed by taking each operational goal from Fig. 3 and operationalising it by means of (one or more) normative primitives in Azzurra. Since there are no OR decompositions at the operational goals model (Fig. 3), no alternative goals needed to be chosen. More concretely, from "Authorise parties to access patient's data" goal, we derive $DisclosureAuth$ and $DisclosureCom$ primitives, while from "Prohibit parties of forwarding patient's data" goal, $DisclosureProh$ and $SanctionCom$ primitives are derived.

Furthermore, although Azzurra protocols are solely founded on the notion of commitments and the 2QUEN framework intends to support compliance and performance monitoring for the different normative primitives from Custard, Fig. 4

uses the four types of norms (commitments, authorisations, prohibitions and powers). Finally, although SIENA methodology also allows the derivation of a procedural specification from Azzurra models, we skip this step by observing that the Custard methodology allows a direct connection between the normative primitives expressed by the Azzurra model and the database relations.

By following SIENA methodology, we have started from operational goals (defined in previous section) and designed the set of business process the company has to achieve its operational goals, together with the internal structure of such business processes. The second step consists of selecting the operational goals that need to be measured and then attach an indicator (e.g. number of patients) to such operational goals. Attaching such indicator to the operational goal means that we know "what" needs to be measured (the operational goal) and "how" to measure it (by defining the algorithm and the relatedTo KPI attributes). Observe also that, since operational goals may be attached to normative primitives, it is also possible to answer compliance questions with such framework (e.g. if the operational goal is attached to a violated prohibition, does it imply also in the failure of the goal?)

With this framework in hands, we intend to develop a bottom-up methodology for monitoring the state of goals, normative primitives and business processes. For that, we have to measure the execution of business process (and their corresponding normative primitives) with events stemmed from process execution that is stored in a process execution environment. If we consider the legacy Custard database as our process execution environment, the Custard tables may contain the following data:

Table I: *Registered*

resID	*pID*	*jID*	*t*
1	Diego	Italy	11-11-18 11:52:30
2	Maria	Italy	10-11-18 15:25:13
3	Maria	Italy	10-11-18 16:27:13
4	Carlo	Italy	09-11-18 20:16:21
5	Andrea	Italy	12-11-18 21:10:26
6	Maria	Italy	12-11-18 22:55:10

.
.
.

Table II: *Accessed*

reqID	*tpID*	*hID*	*t*
1	Evellin	hID1	16-11-18 12:40:00
3	Evellin	hID1	25-11-18 12:35:00
5	Evellin	hID1	19-11-18 15:11:55

From this legacy database, the idea is to construct a log of events to be automatically analysed to determine the answers of compliance and performance questions (Sec. II-A) for business analysts or auditors. Since the IEEE standard eXtensible Event Stream (XES) [19] format is the current standard for the data representation for process mining algorithms, the final outcome of approach should target XES event logs. In order to tackle this problem of log extraction, next section gives the first step towards the 2QUEN methodology for data extraction from legacy databases.

IV. Lifting Normative Primitives Representation

This section presents the methodology for lifting the representation of normative primitives from the 2QUEN behavioural view (Sec. III-B) to reference complex domain entities represented as first-order data objects and subsequently map such entities to the underlying legacy data. The five-step methodology guides the analyst in the development a target information system that reconstructs the elements of the 2OBDA layered architecture (Sec. II-C), starting from 2QUEN normative primitives.

1. Build Upper Schema. The first step of the methodology consists of understanding which 2QUEN elements belong to which 2OBDA layer. Since in a 2OBDA architecture the *upper schema* $\mathcal{T}'$ contains abstractions that reference domain concepts, we interpret that normative primitives represented by the 2QUEN *behavioural view* belong to the 2OBDA *upper schema*, whereas the domain entities (e.g., patient, third party, etc.) and their social relations (e.g. register, etc.) belong to the *domain schema*.

2QUEN normative primitives extracted from Custard norm schema (Sec. II-A) and their admissible states (Fig. 1) have been modelled as an UML model in Fig. 5. Observe that each norm instance has its own lifecycle since each lifecycle stage will be mapped to concepts of the domain schema.

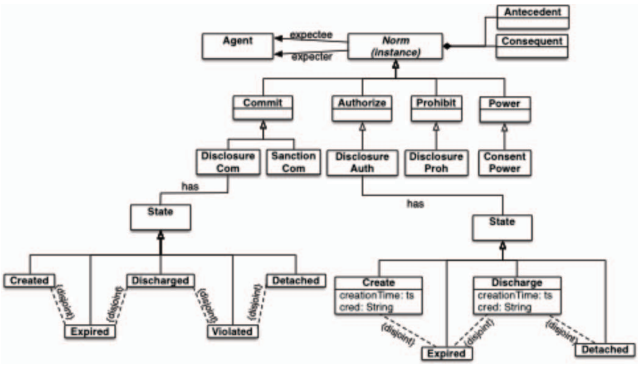

Figure 5: Upper Schema

2. Build Domain Schema. The second element of the 2OBDA architecture consists of developing the *domain schema* $\mathcal{T}$ by capturing the domain ontology as an UML model. Since the Custard *information schema I* latter becomes the Custard database schema, the information schema has been used as starting point to understand the meaning of the data stored by the Custard approach, also enriched by an analysis of the natural language description of the domain. The outcome of this phase is depicted in Fig. 6 with the domain ontology represented as an UML model. Observe that the domain actors (e.g. patient, health vault provider) become concepts in such ontology, related by their interaction types (e.g send credential, require data). Observe also that tables from the *information*

schema represent the events and such events become concepts in the ontology (e.g., request, credential, etc.).

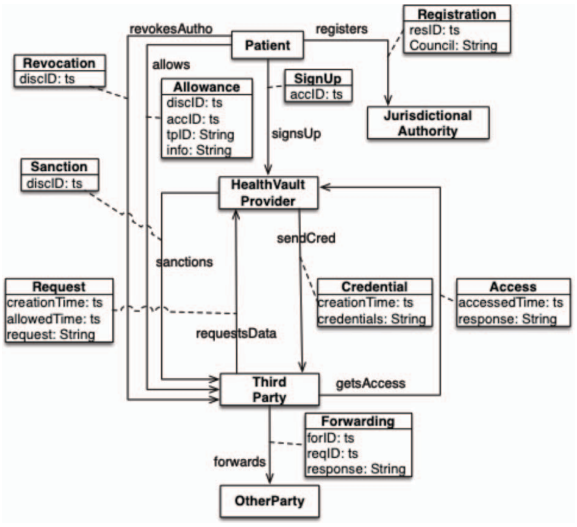

Figure 6: Conceptual (Domain) Schema

3. Build Database Schema. The third layer of 2OBDA architecture requires the *domain schema* $\mathcal{T}$ to be connected with the *relational database schema* $\mathcal{R}$ to allow the domain ontology to access the data stored by the database system. Here, we simple use the Custard *information schema* presented in Example 1 as this schema latter becomes the Custard database schema.

4. Map Domain Schema (Ontology) to Database Schema. With the three schema in hands, the fourth step in our methodology consists of establishing the links between the *domain schema* $\mathcal{T}$ and the underlying *relational schema* $\mathcal{R}$ in terms of *mapping assertions* $\mathcal{M}$.

In OBDA [13], each mapping assertion assumes the form $\phi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$, where $\phi(\vec{x})$ is a SQL query over $\mathcal{R}$, $\psi(\vec{y}, \vec{t})$ is a conjunction of atoms over $\mathcal{T}$ (i.e. atomic concepts and relations of $\mathcal{T}$), $\vec{x}$ and $\vec{y}$ are variables in such way that $\vec{y} \subseteq \vec{x}$ and $\vec{t}$ are variable terms of the form $\mathbf{f}(\vec{z})$, with $\mathbf{f} \in \Lambda$ and $\vec{z} \subseteq \vec{x}$. We give examples of mapping assertions in the context of our work that follow this form:

Mapping Assertions. Below, mapping assertion m_1 has been written to populate the concept of *Credential* from the domain schema. For that, m_1 extracts the *discID* and *credentials* attributes from the *SendCred* table in Custard *information schema* and populate the *Credential* concept and its attribute *creationTime* at the domain schema as follows:

m_1: `SELECT discID, credentials AS cred`
`FROM SendCred`
$\rightsquigarrow$ *Credential* (**credential**(cred)) $\wedge$
creationTime(**credential**(cred)), discID))

Mapping assertion m_2 performs similar operation of populating the concept of *Request* (from domain schema) by extracting the *reqID* and *discID* attributes from the *ReqData* table in Custard *information schema*. Such attributes are then used to create an instance of the *Request* concept. *reqID* is assigned to *creationTime* attribute as it corresponds to the timestamp in which the request has been done, whereas *discID* is assigned to *allowedTime* attribute as it corresponds to the timestamp in which the patient allowed the third part to access her data.

m_2: `SELECT reqID, discID`
`FROM ReqData`
$\rightsquigarrow$ *Request* (**request**(reqID)) $\wedge$
creationTime(**request**(reqID), reqID)) $\wedge$
allowedTime(**request**(reqID), discID))

Similarly to 2OBDA approach [13], we currently do not have native tool support for automatically writing the mappings between the domain ontology and the database schema. Although a manual approach has been adopted here, for cases in which this approach becomes prohibitive due to the complexity and size of the ontologies and database, *bootstrapping techniques* [20] may be used by extracting a preliminary ontology and mappings from a database schemata, if the information system already has a "high-level" structure understandable by domain experts.

5. Map Upper Schema to Domain Ontology. The final step of the methodology consists of relating the *upper-schema* $\mathcal{T}'$ and the *domain schema* $\mathcal{T}$ by means of *transformation rules* to get insights from data using the abstractions from the upper schema.

2OBDA uses transformation rules based on *GAV mappings* that declaratively map the domain schema $\mathcal{T}$ to the upper schema $\mathcal{T}'$. GAV mappings have been widely used in the field of data integration [21] for connecting a global schema with the source schema of multiple data sources in a data integration system. Data from underlying data sources are then accessed by querying the global schema from the data integration system.

Formally, each transformation rule from a TBox $\mathcal{T}$ to a TBox $\mathcal{T}'$ assumes the form of $\phi(\vec{x}) \rightsquigarrow \psi(\vec{y}, \vec{t})$, where $\phi(\vec{x})$ is an union of conjunctive queries (UCQ)[1] over $\mathcal{T}$ with answer variables $\vec{x}$ and $\psi(\vec{y}, \vec{t})$ is a conjunction of atoms over $\mathcal{T}'$ with variables $\vec{y} \subseteq \vec{x}$ and terms $\vec{t}$. Similarly to the OBDA mappings, the terms in $\vec{t}$ have also the form $f(\vec{z})$, with $f \in \Lambda$ and $\vec{z} \subseteq \vec{x}$, and they are used to construct the identifiers of individuals in $\mathcal{T}'$. Essentially, transformation rules populate the upper schema $\mathcal{T}'$ with the information obtained from the answers to queries over the domain schema $\mathcal{T}$.

In order to write transformation rules to populate the upper schema $\mathcal{T}'$, 2OBDA relies on UCQs expressed in SPARQL syntax as a machine-processable language. We adopt the same solution in this work.

Transformation Rules. In order to present a unique example that illustrates the overall approach, we use the norm instance *DisclosureAuth* from Example 2. As can be seen in this example, three stages are specified, i.e., an instance of *DisclosureAuth* is *created* at time t_1 when a credential is sent (*sendCred* event), this instance is *detached* at time t_2 when a

[1]A union of conjunctive queries (UCQ) $q(\vec{x})$ over $\mathcal{T}$ is a First-Order Logic (FOL) formula of the form $\exists \vec{y}_1 . conj_1(\vec{x}, \vec{y}_1) \vee \cdots \vee \exists \vec{y}_n . conj_n(\vec{x}, \vec{y}_n)$, where $\exists \vec{y} . conj(\vec{x}, \vec{y})$ is a FOL formula whose answer variables are $\vec{x}$ [13].

third party requests patient data from a health vault provider (*ReqData* event) and finally, the instance is *discharged* at time t_3 when the third party access the information any time between one and ten days after the request (*Accessed [t_2+1, t_2+10]* event). In this way, each stage of the DisclosureAuth lifecycle in the upper schema (Fig. 5) should be linked to its corresponding concepts and relations at the domain schema (Fig. 6) by means of transformation rules written as SPARQL queries.

In order to populate each stage of the norm lifecycle, we write transformation rules tr_1 and tr_2 as SPARQL queries below. In transformation rule tr_1, the query applied over the domain schema retrieves the concept of *Credential* (since sending a credential is the event that correspond to the *creation* of the *DisclosureAuth*). The result of the query is then mapped to the upper schema concept of *Create* (that corresponds to the creation state of the *DisclosureAuth* norm instance). Transformation rule tr_1 is written as follows:

```
SELECT ?Credential ?creationTime, ?credentials
WHERE {
        ?Credential :creationTime ?creationTime .
        ?Credential :credentials ?credentials .
}
⇝   : discAuth/{Credential} rdf:type :Create  .
  : discAuth/{Credential} :creationTime
{creationTime}^^xsd:dateTime
  : discAuth/{Credential} :cred
{credentials}^^xsd:string
```

Transformation rule tr_2 performs similar operation for the *Discharge* concept in the upper schema (i.e., the discharge state of the *DisclosureAuth* concept). Since the discharge condition states that patient's data has to be accessed between 1 and 10 days after the third party requested the data, this condition is translated to filter condition in the SPARQL query. The other complex event expressions used for referencing Custard information schema (e.g. logic operators, aggregation operators (SUM), set operators (COUNT, MIN, MAX, AVG)) may be also translated to filtering conditions in the SPARQL query.

```
SELECT ?Access, ?accessedTime
WHERE {
        ?Access :accessedTime ?accessedTime
        ?Request :creationTime ?creationTimeReq .
        FILTER

        (?creationTimeReq + 1 <
        ?accessedTime^^xsd:dateTime) &&
        (?creationTimeReq + 10 >
        ?accessedTime^^xsd:dateTime)
}
⇝   : discAuth/{Access} rdf:type :Discharge .
                : discAuth/{Access} :creationTime
{accessedTime}^^xsd:dateTime .
```

Once we have reached this point of the methodology, the next step consists of executing the mappings that will populate the *domain ontology* and *upper schema* with their instances. Subsequently, the XES events log need to extracted and the algorithms to answer our compliance and performance questions from Sec. II-A need to be developed. Both approaches (log extraction and algorithms) are left as future work in this paper.

V. Conclusion

This paper presents a conceptual framework based on the abstractions of *operational goals*, *performance indicators*, *business process* and *normative primitives* for the representation of enterprise models. The use of such abstractions is supported by the need to have holistic enterprise models to monitor the evolution of normative primitives and business processes towards the achievement of enterprise goals.

The second contribution consists of the initial steps of a methodology for semantically linking normative primitives to an underlying legacy database. For that, starting with the normative primitives extracted from the 2QUEN behavioural view, we have designed an upper schema with such primitives and a domain ontology that captures the domain entities and their social relations. The mapping relations between the three schemas (upper, domain and database schemas) have been developed accordingly. Our methodology reproduced the steps to build an 2OBDA architecture, an approach which has achieved a lot of success in the Semantic Web community for querying data sources, thus motivating us to reproduce the same benefits in our field. As the ultimate goal of our approach is to develop a framework to answer the compliance and performance requirements from businesses, this paper gives a first step towards this goal by connecting the normative primitives with data sources. This connection will allow one to semantically reconstruct event logs, thus enabling also to query real-data and answer the compliance and performance questions.

We may receive the criticism that Custard already creates a mechanism to monitor the progression of norm types across the stages of their lifecycle. However, by transforming it into a systematic approach, we believe to significantly help to understanding of the system, thus favouring its maintenance. Further, by decoupling the normative, domain and database specifications, we open the possibility for reusing the framework in other domains, by keeping the upper schema and adopting a new domain ontology and mappings. Finally, our approach also opens the possibility of using the upper schema as the human-friendly artefact to pose queries, thus avoiding to deal with the complexity of Custard SQL queries.

Our framework has two main shortcomings. The first one regards the absence of temporal treatment for the timestamped attributes from Custard information schema. This lack can be seen at transformation t_2 that uses a SPARQL query with filtering conditions to retrieve *DisclosureAuth* that have been accessed between 1 and 10 days after their request. Although the *creationTime* is a timestamped attribute, our approach does not use Allen's interval algebra (e.g. meets, overlaps, before, etc.) [22] to preserve the temporal semantics

of timestamped attributes. The second limitation is related to aggregation operations over the database. Custard makes available aggregation operations (e.g. SUM, AVG, MIN, etc.) to compute normative state transitions on the basis of closed-world assumption (i.e., the information about normative states is assumed to be complete). In contrast, traditional OBDA populates the domain ontology with (potentially incomplete) data from the database. Such differences affect the results of some aggregation operations (e.g., a COUNT operation has a different result if some database tuple is missing), and thus the inference of state transitions is equally impacted. Currently, aggregations and temporal issues are both under development in the context of 2OBDA approach and should be also incorporated in our approach in the future.

Future Work. Although our long-term goal is to use our framework for compliance and performance checking, the achievement of this long term goal would be improved if other features could be considered. First, although the upper schema contains normative primitives which are satisfied/violated in an isolated fashion, incorporating an ordering notion among the achievement of such primitives (i.e., a process model) would significantly enrich our approach. Second, developing a query language for querying the upper schema in terms of its abstractions would facilitate domain experts to express their queries. Third, our overall approach would be significantly strengthen with its application in an industrial use case for accessing data sources.

REFERENCES

[1] A. Maté, J. Trujillo, and J. Mylopoulos, "Conceptualizing and specifying key performance indicators in business strategy models," in *Conceptual Modeling*. Springer Berlin Heidelberg, 2012, pp. 282–291.

[2] J. Horkoff, D. Barone, L. Jiang, E. Yu, D. Amyot, A. Borgida, and J. Mylopoulos, "Strategic Business Modeling: Representation and Reasoning," *Software & Systems Modeling*, vol. 13, no. 3, pp. 1015–1041, 2014.

[3] N. M. Zaki, A. Awad, and E. Ezat, "Extracting accurate performance indicators from execution logs using process models," in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, 2015, pp. 1–8.

[4] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, M. Montali, and P. Torroni, "Expressing and verifying business contracts with abductive logic programming," *International Journal of Electronic Commerce*, vol. 12, no. 4, pp. 9–38, 2008.

[5] Y. F. Hernández-Julio, M. J. Paba, N. E. L. Narváez, H. M. Hernández, and W. N. Bernal, "Framework for the development of business intelligence using computational intelligence and service-oriented architecture," in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, 2017, pp. 1–7.

[6] R. K. M. Veneberg, M. E. Iacob, M. J. v. Sinderen, and L. Bodenstaff, "Enterprise architecture intelligence: Combining enterprise architecture and operational data," in *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*, 2014, pp. 22–31.

[7] G. Xiao, L. Ding, B. Cogrel, and D. Calvanese, "Virtual Knowledge Graphs: An overview of systems and use cases," *DataInt*, vol. 1, no. 3, pp. 201–223, 2019.

[8] E. Cardoso, "Strategic Reasoning for Enterprise Architectures: The SIENA Modeling Framework," Ph.D. dissertation, University of Trento, Trento, Italy, 2018.

[9] A. Chopra and M. Singh, "Custard: Computing norm states over information stores," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, ser. AAMAS '16. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1096–1105.

[10] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, "Linking data to ontologies," in *Journal on Data Semantics X*. Springer-Verlag, 2008, pp. 133–173.

[11] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev, "Ontology-based data access: A survey," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 5511–5519.

[12] R. Ferrario and N. Guarino, "Commitment-based modeling of service systems," in *Proceedings of the 3rd International Conference on Exploring Services Science (IESS'12)*. Springer Berlin Heidelberg, 2012, pp. 170–185.

[13] D. Calvanese, T. E. Kalayci, M. Montali, A. Santoso, and W. van der Aalst, "Conceptual schema transformation in ontology-based data access," in *Proceedings of the 21st Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW 2018)*, ser. Lecture Notes in Computer Science. Springer, 2018.

[14] M. P. Singh, "Formalizing communication protocols for multiagent systems," in *IJCAI-07*, 2007, pp. 1519–1524.

[15] E. Cardoso, J. Mylopoulos, A. Mate, and J. Trujillo, "Strategic enterprise architectures," in *The Practice of Enterprise Modeling - 9th IFIP WG 8.1. Working Conference, PoEM 2016, Skövde, Sweden, November 8-10, 2016, Proceedings*, 2016, pp. 57–71.

[16] F. Dalpiaz, E. Cardoso, G. Canobbio, P. Giorgini, and J. Mylopoulos, "Social Specifications of Business Processes with Azzurra," in *Proceedings of the IEEE 9th International Conference on Research Challenges in Information Science (RCIS'15)*. IEEE, 2015, best Paper Award, selected among 199 submissions.

[17] E. Cardoso, "Challenges in performance analysis in enterprise architectures," in *2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops*, 2013, pp. 327–336.

[18] M. Singh, "An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts," *Artificial Intelligence and Law*, vol. 7, pp. 97–113, 1999.

[19] I. C. I. Society, "Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams," *IEEE Std 1849-2016*, pp. 1–50, 2016.

[20] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, C. Pinkel, M. G. Skjæveland, E. Thorstensen, and J. Mora, "Bootox: Practical mapping of rdbs to owl 2," in *The Semantic Web - ISWC 2015*. Springer International Publishing, 2015, pp. 113–132.

[21] M. Lenzerini, "Data integration: A theoretical perspective," in *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '02. ACM, 2002, pp. 233–246.

[22] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communication. ACM*, vol. 26, no. 11, pp. 832–843, 1983.