Service Choreography, SBVR, and Time*

Nurulhuda A. Manaf, Sotiris Moschoyiannis, and Paul J. Krause
Department of Computer Science, University of Surrey
Guildford, Surrey, GU2 7XH, UK
{n.amanaf, s.moschoyiannis, p.krause}@surrey.ac.uk

We propose the use of structured natural language (English) in specifying service choreographies, focusing on the *what* rather than the *how* of the required coordination of participant services in realising a business application scenario. The declarative approach we propose uses the OMG standard *Semantics of Business Vocabulary and Rules* (SBVR) as a modelling language. The *service choreography* approach has been proposed for describing the global orderings of the invocations on interfaces of participant services. We therefore extend SBVR with a notion of time which can capture the coordination of the participant services, in terms of the observable message exchanges between them. The extension is done using existing modelling constructs in SBVR, and hence respects the standard specification. The idea is that users - domain specialists rather than implementation specialists - can verify the requested service composition by directly reading the structured English used by SBVR. At the same time, the SBVR model can be represented in formal logic so it can be parsed and executed by a machine.

1 Introduction

There is increasing interest in developing distributed applications that involve stand-alone services from different organisations on the web. However, the coordination of the interactions between the underlying services in building such applications remains a challenge. Sustained efforts by the web services community have culminated in the *service choreography* approach [34] which is concerned with describing the conversation across different participating services (global perspective) as well as the *service orchestration* approach [24] which describes the interaction scenario from an individual service's viewpoint. Service choreography in particular, is intended to capture the coordination of the participant services, in terms of the observable message exchanges between them. This is given mostly in terms of the orderings of these interactions during the execution of the corresponding business activity.

The orderings of the interactions (invocations on interfaces) between the underlying services is key in coordination as they capture the dependencies between participant services and thus the correctness of the application design. Well known issues (e.g., see [4]) that involve the orderings of interactions include *deadlock* and *race conditions* (a situation where two or more messages are competing to arrive first, so while the appear to be ordered in a given execution they are effectively unordered). In the context of service choreography, verification additionally comes in the form of *conformance* and *realisation* [33, 5]. Moreover, if choreographies are to be equipped with transactional guarantees [22], meaning that a series of compensations are performed upon failure, the ordering of the interactions is doubly important.

Declarative approaches in the Business Rules realm [30], [8] focus on *what* rather than *how*. The 'what' and the 'how' of a solution to a computing problem are quite different. The 'what' refers to the properties of a solution whereas the 'how' refers to the steps followed to achieve the solution. Declarative

*This work was partly funded by the UK Research Council EPSRC, under the project *Evolution and Resilience of Industrial Ecosystems* (ERIE) , Contract No. EP/H021779/1.

José Proença and Massimo Tivoli (Eds.):
14th International Workshop on Foundations of
Coordination Languages and Self-Adaptive Systems (FOCLASA15)
EPTCS 201, 2015, pp. 63–77, doi:10.4204/EPTCS.201.5

programming focuses on specifying the 'what' and using a general-purpose engine for reaching the goal. An example declarative language is SQL, which specifies properties of data but not the way to retrieve it. The latter is left to the database management system (DBMS) implementation [7]. Imperative programming focuses on the 'how', bypassing the need to define the properties of the required solution since programmers can guarantee the desired properties by directly controlling the algorithm. Java and C are generally considered imperative languages.

With respect to specifying service choreography, the declarative approach can express business requirements intuitively, e.g., see [8, 10]. The Business Rules manifesto [30] builds the business requirements on the premise that rules or policies in a business application scenario should be expressed declaratively in natural-language sentences for the business audience. A rule is distinct from any enforcement defined for it. A rule and its enforcement are separate concerns. Also, rules apply across processes and procedures. In addition, the issue of understandability in expressing and modelling complex business requirements is important especially if we want the provision for the domains specialists to validate the specification against their business models. Note that domains specialists (business analysts, stakeholders) rather than implementation specialists are best positioned to validate against the business activities taking place in practice. In our approach, the choreography specification is expressed in terms of statements like the following:

It is obligatory that each **rental car** *is owned by* at least one **branch**

In addition, a declarative approach typically starts with an unconstrained view of the specification and gradually constrains it, by means of adding rules, as the intended behaviour of the service choreography becomes more clear. This is in contrast to the more traditional imperative approach, which tends to be more restrictive and sometimes results in introducing artificial decision points, or forcing premature decision points, for the practitioner or over- / under-specifying what actually happens [21].

Work on formal semantics in this area has focused more on the imperative (or procedural) approach and service orchestration and less so on the declarative approach and service choreography. Existing work, e.g., [21, 15, 2] that takes a declarative approach tends to focus on reasoning about consistency of the rule set, which of course is an important aspect of verification, but have not looked into explicitly capturing the orderings, in terms of observable message exchanges in a choreography. In addition, and to the best of our knowledge, none of the current proposals for a declarative approach to service choreography has attempted to provide the end-user with something close to natural language.

In this paper, we propose a declarative approach which builds on using the *Semantics of Business Vocabulary and Rules* (SBVR) [26] for the specification of service choreographies. SBVR is a standard maintained by the Object Management Group (OMG) and uses structured natural language, which makes it specifically understandable by humans. The business rule given earlier is actually written in SBVR and makes use of two Fact Types (cf Section 2). There is no explicit notion of time in SBVR. In order to capture the global ordering constraints on observable actions (invocations) in a service choreography we describe the use of sequencing of Fact Types in an SBVR model, together with the modelling constructs of *objectification* and *actuality*. This allows us to specify, for example, that a product is received by the customer only after it is delivered by the shop.

The remainder of this paper is structured as follows. Section 2 contains a brief account of the business rules approach and SBVR. In Section 3, we build an SBVR model for a case study, an Online Photo Shop, which focuses on the ordering of service interactions. This forces us to look at temporal aspects and thus Section 4 describes our handling of time ordering within SBVR. Sections 5 gives a brief account of related work. Some concluding remarks and ideas for future work are included in Section 6.

2 The Business Rules approach and SBVR

Several specification or modelling languages for specifying interactions between services in a business application scenario are available to practitioners, with varying levels of adoption, such as the *Business Process Model Notation* (BPMN) [25], *Web Services Choreography Description Language* (WS-CDL) [34], *Web Services Business Process Execution Language* (WS-BPEL) [24]. These languages require training to read and write and hence may not lend themselves naturally to be used by the end-user directly.

The OMG standard *Semantics of Business Vocabulary and Rules* (SBVR) [26] is gaining ground as a basis for system specification. By inception, SBVR is intended to provide a way to capture specifications in natural language and represent them in formal logic so they can be machine processed. Users are able to verify the specification directly by reading the structured natural language used by SBVR which can then be parsed and executed by a machine. In line with the Business Rules Approach [30], it follows the doctrine: "*Rules build on facts, and facts build on concepts as expressed by terms. Terms express business concepts; facts make assertions about these concepts; rules constrain and support these facts.*"

As argued in [19], while SBVR is a meta-model with models natively expressed as logical formulations, its most common serialization is SBVR Structured English. Terms (e.g., **branch**), Fact Types (e.g., **rental car** *is owned by* **branch**), and rules (e.g. It is obligatory that each **rental car** *is owned by* at least one **branch**) are combined into models. An example of an SBVR model can be seen in Figure 1. It refers to the Rental Car case study included in its spec document[26].

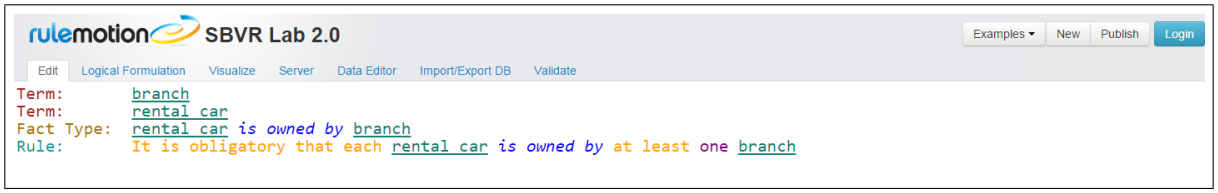

Figure 1: Part of an SBVR model for the specification of the Rental Car case study

The rule in Figure 1 is written using our web-based SBVR editor [18] maintained by Rulemotion[1] and is a representation of higher-level facts that use the deontic constraint, obligatory on the constraint defined by the rule. The quantifications, each and at least one show the restriction of rental car belonging. Furthermore, *is owned by* is the designation for the Fact Type in Table 1. Fact Type is constructed based on identified Terms (a noun concept, **rental car** and **branch**). Thus, the combination of deontic constraint, quantifier, terms and fact type will yield a constructive rule. This type of rule can be used by domain specialists (e.g., business analysts) in defining the business model or activity to be performed with a choreography.

[32] provides a syntax and semantics for the logical formulations of SBVR, a *first-order deontic-alethic logic (FODAL)*. It is an extension of first-order logic which is a combination of standard-deontic logic (SDL) and normal modal logic, S4. The syntax of *FODAL* [32] includes a set of propositional connectives ($\neg$, $\wedge$), a universal quantifier ($\forall$), an infinite set $\mathscr{P}$ (predicate symbols), an infinite set $\mathscr{V}$ (variable symbols), and, modal operators ($\Box$ (necessity) and $\boldsymbol{O}$ (obligation)) for alethic and deontic respectively. To formalise SBVR rules, *FODAL* follows the first-order modal formulae which are specified by the rules :

- Every atomic formula is a formula.

[1] With thanks to Rulemotion, the editor *SBVR Lab 2.0* is available at `http://sbvr.co`

- If X is a formula, so is $\neg X$.
- If X and Y are formulas, then $X \wedge Y$ is a formula.
- If X is a formula, so are $\Box X$ and $\boldsymbol{O}X$.
- If X is a formula and v is a variable, then $\forall v X$ is a formula.

The usual definition is used for the existential quantifier ($\exists$) and other propositional connectives ($\vee$, $\rightarrow$, $\leftrightarrow$). However, there are additional modal operators defined for possibility ($\Diamond$), permission ($\boldsymbol{P}$), and prohibition ($\boldsymbol{F}$) which are "It is possible that ϕ" is logically equivalent to "It is not necessary that not ϕ" ($\Diamond\phi \equiv \neg\Box\neg\phi$), "It is permitted that ϕ" is logically equivalent to "It is not obligatory that not ϕ" ($\boldsymbol{P}\phi \equiv \neg\boldsymbol{O}\neg\phi$), and "It is forbidden that ϕ" is logically equivalent to "It is obligatory that not ϕ" ($\boldsymbol{F}\phi \equiv \boldsymbol{O}\neg\phi$).

In addition, [32] provides a *Kripke* semantics for *FODAL* as well as the proofs of its sound and complete axiomatisations with respect to the semantics [31]. The axioms of *FODAL* implies the combination of the axiom systems for the propositional modal logics ***S4*** and a serial relation of a deontic modality behaviour (***KD***) as well as the interaction between alethic and deontic modalities. The *FODAL* axioms as in [32] are shown as follows:

(Tautologies *S4*) Any FOL substitute-instance of a theorem of ***S4***

(TAutologies *KD* Any FOL substitute-instance of a theorem of ***KD***

(Vacuous $\forall$) $\forall x\phi \equiv \phi$, provided x is not free in ϕ

($\forall$ Distributivity) $\forall x(\phi \rightarrow \psi) \rightarrow (\forall x\phi \rightarrow \forall x\psi)$

($\forall$ Permutation) $\forall x \forall y \phi \rightarrow \forall y \forall x \phi$

($\forall$ Elimination) $\forall y(\forall x\phi(x) \rightarrow \phi(y))$

Necessary *O* $\Box\phi \rightarrow \boldsymbol{O}\phi$

Even though *FODAL* is undecidable, [32] identifies a decidable fragment of *FODAL* logic. This is the set of atomic modal sentences with at most two variables, all predicate symbols with at most unary, and the set of atomic modal sentences in which are applied to subformulas from the guarded fragment of firs-order logic.

In the context of service choreography, it is important to capture the ordering of observable actions (service interactions) as discussed before. SBVR does not include a notion of time and therefore with respect to time and ordering, OMG has supplemented it with the *Date-Time Vocabulary* (DTV) [27]. To be more precise, DTV expresses the specification in the form specified in Annex C of SBVR as defined by OMG [26]. It was introduced to encapsulate the SBVR rules that involve concepts such as date and time (excluding real-time processing) which are frequently used in everyday business activities across a wide range of business scenarios.

Two types of time are considered in DTV. Type 1 refers to a time period, an explicit time interval. Type 2 uses temporal concepts to define a relationship between *situation kinds* and *occurrences*. These are used to represent the potential real activities or events that occur multiple times in a business environment. In our work on choreography specification we apply Type 2 from DTV as well as the SBVR verb concept *objectification* [26] in order to express the ordering of exchanged messages and corresponding Fact Types, e.g., ***A*** before ***B*** . This will be further discussed in Section 4.

3 Service Choreography Specification using SBVR

In this section we describe how an SBVR model can be built for the service choreography involved in the Online Photo Shop case study, which was originally studied in [21]. We identify the need for expressing the ordering relationship between observable events and propose a way to express such orderings in an SBVR model.

3.1 Online Photo Shop: a case study

We look at the case study of an Online Photo Shop in [21] which provides services for placing orders and printing photographs (and other products) to customers. The business scenario involves a multi-party conversation between several services; *Customer, Photo-Shop, Order, Print,* and *Deliver*. All services are provided by the Online Photo Shop entity, hence in this case they all belong to one organisation. The conversation respects the policies underlining the business activities involved, as described in [21].

However, we notice that there are certain problems with the specification of the Online Photo Shop as given in [21], such as ambiguities in defining an activity (e.g., 'open order' and 'register') while some prescribed orderings on activities are questionable (e.g., customer may 'pay' before or after Photo Shop performs 'charge'). We have amended the specification slightly to steer away from such issues. This will allow us to focus on using SBVR to specify the choreography rather than elaborating the specification itself. Below is the description of responsibilities by the web services *Customer, Photo-Shop,* and *Order*.

Service Customer *provides a service for customer to:*

- *register an account at photo shop by entering data, such as name, address, credit card number and preferred way of delivery through activity "register";*
- *pay for ordered products via activity "pay for";*
- *receive ordered products via activity "receive".*

Service Order *allows the customer to order photos and posters via activity "photo" and "poster" respectively by uploading files and selecting wanted formats or to order photo albums by selecting the preference photo album via activity "album".*

Service Photo-Shop *provides:*

- *the products; photos, posters, and albums. It ensures the shop records the customers data via activity "update";*
- *a service to print ordered photos and posters via activity "print";*
- *a service to deliver products (photo, poster, or album) to customer via activity "deliver".*

As a flavour of the kind of changes we made to the case study presented in [21], the activity "update" has been introduced in place of "open order" in the 1st constraint shown in Table 1 so as to maintain the semantic meaning of the activities. The 5th constraint is needed to bind the product that is ordered by the customer to said customer, for each payment. Also, the "pay for" activity is now prescribed to take place before the execution of the "deliver" activity.

3.2 SBVR model for the Online Photo Shop service choreography

The informal specification of the business scenario can be addressed in a way similar to how entities, attributes and relationships are drawn from a textual specification as done in information modelling and

1. The shop will not ”update” the customer’s data before the customer executes activity ”register”. When the customer executes activity ”register”, the shop will update its data via activity ”update”.
2. After the customer orders photos and posters via activity ”photo” and ”poster” respectively, the shop prints ordered products via activity ”print”.
3. Each ordered product (photo, poster or album) through activity ”photo”, ”poster”, or ”album” has to be delivered via activity ”deliver”. The shop will not ”deliver” before at least one product is ordered.
4. Customer can ”receive” products only after the shop executes ”deliver”. All ordered products must be received by the customer through activity ”deliver”.
5. Customer has to ”pay for” each ordered product (photo, poster or album) made by the customer.
6. Customer has to ”pay for” each ordered product before the shop delivers the ordered products via activity ”deliver”.

Table 1: Amended global constraints for the Online Photo Shop

relational database design [8]. Hence, nouns are candidate terms and verbs connecting these nouns together are candidate (binary) fact types. Using this as a rule of thumb, the following Terms (Figure 2) and Fact Types (Figure 3) have been extracted for the Online Photo Shop (Section 3.1).

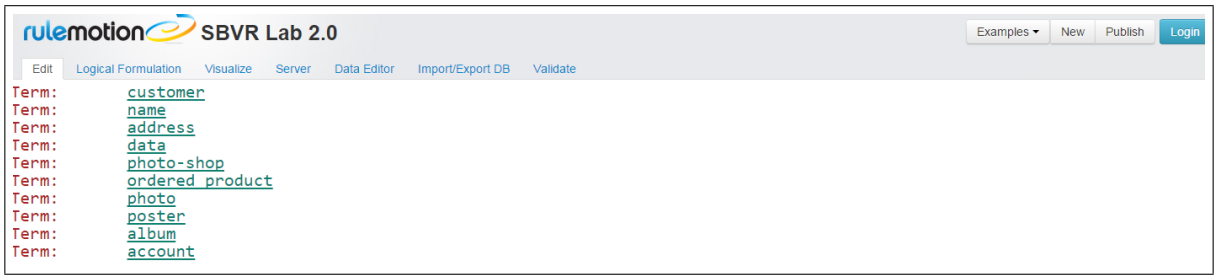

Figure 2: Terms in the Online Photo Shop case study

As discussed in Section 2, Term as a noun concept is applied to Fact Type to show the concept that is the meaning of the noun. Fact Type is a combination of one, two or more Terms. The Fact Types formed for the Online Photo Shop case study are shown in Figure 3. For example, the Fact Type, ’**customer** *receives* **ordered product**’ shows **customer** as a role that specifically characterises **ordered product** role by their involvement in the activity, while *receives* represents the verb concept of the factual relationship between the two terms. Similarly, the Fact Type, ’ **name** *is of* **customer**’ shows **name** and **customer** as noun concepts, where **name** is an attribute to characterise **customer** while *is of* is a verb concept for the Fact Type.

The Terms and the Fact Types make up the *Business Vocabulary* in an SBVR model.

It might be worth noting that there is a synonymous form for Fact Types in SBVR [26], which allows to identify Fact Types that convey the same meaning. For example:
Fact Type: **name** *is of* **customer**; Synonymous Form: **customer** *has* **name**

The synonymous form is useful when it comes to verifying the rule set in the SBVR model - whether all rules have been captured; whether any rules are in conflict.

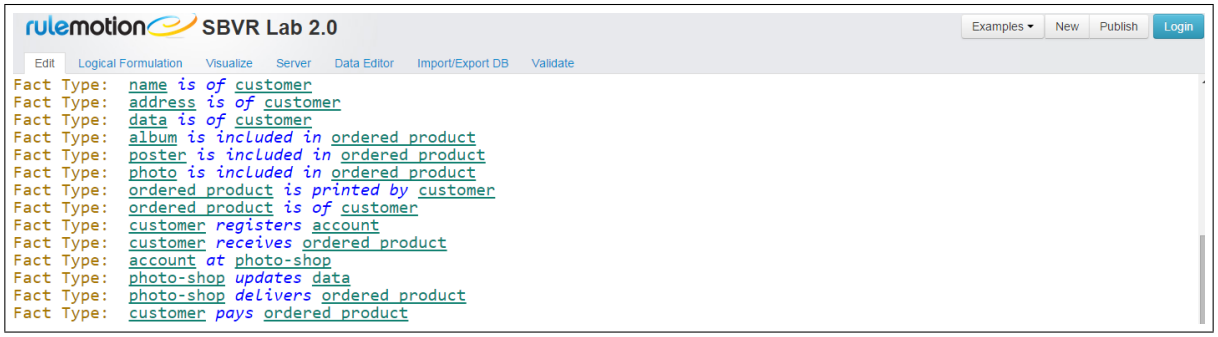

Figure 3: Fact Types in the Online Photo Shop case study

Fact Types are then used to construct the rules in the model as shown in Figure 4. As mentioned in Section 2, rules are expressed using appropriate quantification, logical operations (if applicable) and modalities. In what follows we use the rules from Table 2.

Figure 4: Rules in the Online Photo Shop case study

Rule
It is obligatory that the **photo-shop** *delivers* the **ordered product** that *is of* each **customer** and at least one **ordered product** that *is of* the **customer**

Table 2: Business rules for the Online Photo Shop (the *Rules* in an SBVR model)

Semantic formulations are used in SBVR to structure the meaning of rules and come in two flavours; logical formulation and projection [26]. All rules in the previous examples are structured according to the specialisations of logical formulation, e.g., logical operations, quantification, etc. An instantiation formulation also is one of the logical formulations that bind a concept (e.g., individual concept) to a bindable target (variable or an individual concept from logical formulation) to formulate the meaning. This is vital in the rules to express an accurate meaning.

For example, the specification in the case study (Table 1) attempts to prevent the situation where the photo shop attempts to deliver a product that no customer has ordered. To this effect, the logical formulation: It is obligatory that the **photo-shop** *delivers* the **ordered product** that *is of* each **customer** has been added to the set of rules in the SBVR model. According to this rule, the ordered product (photo/poster/album) which is a concept (delivered by the photo shop) binds to the ordered product

which is an individual concept (ordered by the customer). Additionally, that which is located after the designation for a noun concept, **ordered product** here, and before the designation for verb concept, *is of*, is used to restrict that keyword on the previous designation based on facts about them.

Moreover, Table 2 shows the rule that is supposed to represent the ordering of the activities. The rule however does not capture the dependency between the activities, i.e., that activity 'deliver' by the photo shop must occur after activity 'order' by the customer of at least one product. Thus, this is why a notion of ordering between fact types in SBVR is proposed, in the next section, to capture the type of rules which prescribes the ordering of the underlying service interactions in a business scenario.

4 A notion of time in business rules: ordering of service interactions

We have seen that with respect to coordination it is necessary to capture the ordering of service interactions in order to encapsulate the important properties of domain in a choreography. In the Introduction, we outlined some reasons why the ordering of service interactions is important. Without a notion of time, e.g., precedence, it is not straightforward to construct a rule which prohibits certain anomalies coming into view such as race conditions. For instance, the temporal ordering "precedence" need to be placed in between the Fact Type: **customer** *pays for* **ordered product**, and the Fact Type: **ordered product** *is delivered by* **photo-shop**. In view of that example, it seems appropriate to look into expressing ordering of Fact Types in SBVR without changing the OMG standard or introducing special primitives particular to our approach. The idea is to express ordering in terms of dependency between certain messages (*causality*), and by implication also choice (*conflict*) and *concurrency*, which should be possible, especially if a true concurrency semantics is pursued as done in [22].

It transpires that such a notion of time, i.e., the ordering of service interactions in a choreography, is closely related to Type 2 in the *Date-Time Vocabulary* (DTV) [27], which is a supplementary specification to SBVR by OMG. Type 2 of Time aspects in DTV concerns a relationship between situation kinds and occurrences. The construction based on Type 2 in DTV (pp. 183-215 in [27]) draws upon the concept of "state of affairs" in SBVR which refers to an event, activity, situation, or circumstance that is *actual* (in fact, defined as an *actuality* in SBVR spec [26]). The actuality itself is an instance of a verb concept. For example, the proposition 'a **customer** *pays for* an **ordered product**' has the actuality (state of affairs), 'an **ordered product payment** that *is of* a **customer**' which is formed out of the verb concept *pays for* while both the proposition and state of affairs satisfy the subclause in SBVR spec [26] which say that "it is necessary that each **proposition** *corresponds to* exactly one **state of affairs**".

However, DTV does not agree with that necessity because the proposition that corresponds to a state of affairs, as stated in the example, refers to one event only. On the contrary, the objective of DTV is to represent real states of affairs that occur multiple times. For this reason, DTV introduces a "situation kind" in place of state of affairs. A situation kind refers to a type of situation, event or activity that may occur multiple times. It is related closely with occurrence, so a typical example of a situation kind that occurs in actual situation at some place and time. For instance, the situation kind, 'an **ordered product payment** that *is of* a **customer**' refers to the activity of (binding) a customer paying their ordered product, which may occur multiple times in future.

Therefore, we apply a temporal ordering of situation kinds which is specifically based on the template '**situation kind**$_1$ *precedes* **situation kind**$_2$' that further defines in SBVR that 'each **occurrence** *of* **situation kind**$_1$ *precedes* each **occurrence** *of* **situation kind**$_2$'. This allows comparing the ordering of two situation kinds. The following rule makes use of this temporal ordering: 'It is obligatory that each **customer** *pays for* each **ordered product** that *is of* the **customer** *precedes* that **ordered product** *is*

delivered by the **photo-shop**'. This is in fact the rule in the SBVR model of the choreography that captures the precedence constraint this section opened with.

There is also a temporal ordering of occurrences in DTV [27] which can prescribe '**occurrence**$_1$ *precedes* **occurrence**$_2$'. In this Type 2 of time aspects in DTV, the "occurrence" is defined as the occurrence interval (specific time interval). Hence, it is not directly related to the notion of time considered here.

Our approach to ordering Fact Types in SBVR includes the use of the construct of *objectification* in SBVR [26]. This verb concept is used to specialise the "state of affairs" which in turn specialises "situation kinds" and "occurrence". Objectification may fill verb concept roles that range over a "situation kind" as well as an "occurrence". For example, the verb concept objectification as state of affairs for 'an **ordered product payment** that *is of* a **customer**' is defined as 'a **customer** *pays for* an **ordered product**'. It may be used with the verb concept '**photo-shop** *needs* **situation kind**' and also with the verb concept '**photo-shop** *records* **occurrence**'. Thus, to express the ordering of service interactions in a choreography, we use a notion similar to Type 2 in DTV [27] and apply the *objectification* construct of SBVR [26]. An example rule that uses this notion of time ordering from the Online Photo Shop is given by:

'It is obligatory that each **customer** *pays for* each **ordered product** that *is of* a **customer** *precedes* that **ordered product** *is delivered by* the **photo-shop**'

This rule has the following two propositions or Fact Types: '**customer** *pays for* **ordered product**' and '**ordered product** *is delivered by* **photo-shop**'. Therefore, the situation kinds, 'an **ordered product payment** that *is of* a **customer**' and 'the **ordered product delivery** *by* the **photo-shop**' are an actuality denoted by the verb concept objectification of the two propositions or Fact Types.

By considering the vocabulary structure in SBVR: '**situation kind**$_1$ *precedes* **situation kind**$_2$' and taking 'an **ordered product payment** that *is of* a **customer**' to be **situation kind**$_1$, and 'the **ordered product delivery** *by* the **photo-shop**' to be **situation kind**$_2$ the expression of the rule is given as:

'It is obligatory that an **ordered product payment** that *is of* a **customer** *precedes* the **ordered product delivery** *by* the **photo-shop**'

This ability to express temporal constraints in business rules has been applied to our case study to show ordering of service interactions. All the rules in a choreography are transformed to the SBVR Logical Formulation (Ch. 10 in [26]) as shown in Figure 5.

Figure 5 shows the representation of the Logical Formulation of the stated rule. It represents the relationships between the obligation formulation, the atomic formulations, the instantiation formulation, and the objectification in the rules based on the Simplified Syntax for Logical Formulations in Annex F of DTV [27]. The instantiation formulation in the logical formulation can be seen in Figure 5 whenever the first variable '**ordered product**' binds to the concepts '**ordered product payment**' and '**ordered product delivery**' to show both concepts are referring to the same ordered product that was ordered and paid for by the customer. Then, the rules are translated into first order logic, following the SBVR Logical Formulation [26]. This transformation opens up the space for reasoning (verification) as well as model transformations between different tools.

An example of the translation for one rule from our case study is given in Table 3. The first order logic expression $o\forall x\forall y\exists^1 z(B(P(y,x)\wedge Q(x,y),T(x,z)))$ says that "for all products, x, for all customers, y, there exists at most one photo shop, z, such that y pays for x, and x is of y, precedes x is delivered by z". This is half way between the logic and the structured English of the business rule.

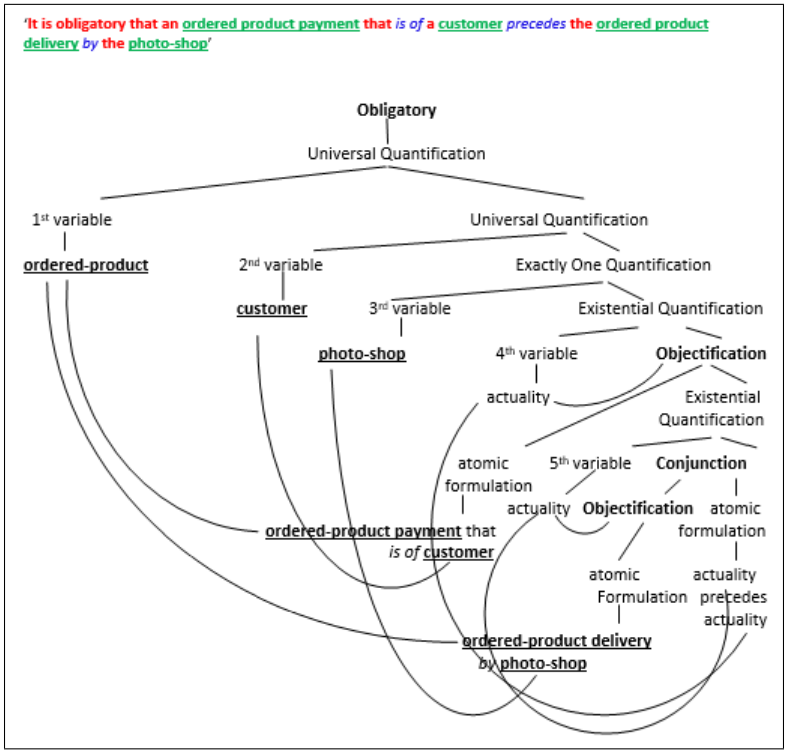

Figure 5: SBVR Logical Formulation

Declaration	**Business Rule**	**First Order Logic**
x is an ordered product; y is a customer; z is a photo-shop; $P(y,x)$: y pays for x; $T(x,z)$: x is delivered by z; $Q(x,y)$: x is of y; $B(P(y,x) \wedge Q(x,y), T(x,z))$: $P(y,x) \wedge Q(x,y)$ precedes $T(x,z)$	It is obligatory that each **customer** *pays for* each **ordered product** that *is of* a **customer** *precedes* that **ordered product** *is delivered by* the **photo-shop**	$o\forall x \forall y \exists^1 z(B(P(y,x) \wedge Q(x,y), T(x,z)))$

Table 3: SBVR First Order Logic

5 Related Work

The work on *DecSerFlow* proposed in [21] provides a declarative language together with a logical framework for reasoning while the work described in [12] uses the implementation of the *Business Process Execution Language for Web Services* (BPEL4WS) (see the specification document [24]) and its semantics for choreography specification. The representation of a choreography is given in the form of graphical specification of service flows which can be mapped onto Linear Temporal Logic (LTL) [13] and in the form of XML data format definition which is then translated to Finite State Process (FSP) process algebra, respectively, thus allowing to model the required behaviour. This enriches the expressiveness and allows to perform interoperability and verification tasks, including conformance checking and deadlock detection [4]. [12] also provides a tool, LTSA-WS for checking the correctness of the service interactions in terms of whether they correspond to those specified in the requirements. While these approaches provide reasoning capability, *DecSerFlow* is a proprietary language while [12] uses both an informal and a formal language that require training to read and write for specifying software services. Hence, both comprise a learning curve for practitioners (business analysts or stakeholders).

Furthermore, [2] also place emphasis on coordination of service interactions that correlate to the choreography specification - a model derived from BPMN 2.0 [25] is implemented. On the other hand, [9] provides a declarative approach and applies UML activity diagram (as illustration purposes) for describing and capturing the ordering constraints between interactions, yet for service interface adaptation. Additionally, [6] provides an integrated tool support for the specification and validation and verification for adaptation contracts. In other words, [9] and [6] focus on discovering mismatches between behavioural interfaces which this different with our focus as discussed earlier in Section 1.

The approach built around SBVR [26] which we propose in this paper for modelling service choreographies uses a structured natural language standard maintained by OMG, which comes with a logical formulation (recall Section 4) which can be exploited for reasoning about correctness. This means that an SBVR model can be parsed and is machine readable. We have more to say on this in the concluding section of the paper (Section 6).

In introducing a notion of time, understood in terms of ordering of Fact Types, we chose not to come from the angle of the *Object Constraint Language* (OCL) [28] or Allen's temporal logic [1]. This is because although these approaches deal with time, OCL introduces Collections to manage an *OrderedSet* and a *Sequence* which uses elements to represent the occurrence of objects. Both constructs contain a collection where the elements are ordered. However, the *OrderedSet* contains unique elements which means no duplicates of the elements exist while the elements in Sequence may be part of a sequence more than once. For example, consider a class Employee with an attribute 'age'. Collection contains three employees such as *employee1.age = 24*, *employee2.age = 27* and *employee3.age = 27*. Thus,
Expression : self.employees $\rightarrow$ sortedBy(age);
Result : Sequence: *employee1, employee2, employee3*

In [1] a temporal logic was developed to represent knowledge of properties, events, and actions using one primitive object, namely the time period, and one primitive relation 'Meets' (*m* and *n* meet if and only if *m* precedes *n*). One can describe *m* precedes *n*, but *m* and *n* represent two time periods. Hence, both OCL and Allen's temporal logic are not suitable to address the ordering of Fact Types inside a rule.

It might be instructive to also compare the way a choreography can be expressed in *DecSerFlow* [21] and in SBVR. For example, *succession(A,B)* is used in *DecSerFlow* to constrain the Online Photo Shop rules: the shop will not 'open order' before the customer executes activity 'register'. Hence, the 'succession' constraint replaces parameter 'A' with activity 'register' and parameter 'B' with activity 'open order'. It is then converted to a logic expression (in Linear Temporal Logic (LTL)) in the following

DecSerFlow template:

succession(register, open order) = response(register, open order) precedence(register, open order)

In contrast, in SBVR the corresponding rule will be

It is obligatory that each **customer** *registers* at most one **account** *at* the **photo-shop** *precedes* the **photo-shop** *updates* the **data** that *is of* the **customer** .

We believe that this global constraint is more understandable to domain specialists (business analysts) but also humans in general.

6 Conclusion and Future Work

In this paper, we presented a declarative approach to the coordination of distributed applications comprising stand-alone services. We proposed the use of SBVR for choreography specification and demonstrated how *objectification* and *actuality* can be exploited in impressing temporal aspects between Fact Types that appear in a rule. This notion of time ordering is reminiscent of Type 2 in DTV [27], a supplementary specification document to SBVR, also advocated by OMG, and is useful in capturing the global constraints in a multi-party conversation involved in a service choreography.

SBVR can be used to support the development of ontologies and business models using structured natural language. It is widely used to cope with complex requirements of business operations with a language that is easily understandable by business analysts (domain specialists) rather than systems analysts (implementation specialists). We are currently exploring the possibility of integrating SBVR with the work on participatory modelling, where we use Fuzzy Cognitive Maps (FCM) and techniques from network analytics in identifying strategic intervention points in complex networks. This is looking into the Humber region, UK, (one of the UK's most important energy hubs) as a case study where local authorities and various groups of stakeholders engage in moving from a fossil fuel economy to a bio-based economy [29]. The FCM map is built over a one-day workshop with stakeholders and the whole process could be helped by using SBVR to capture expertise and dependencies in the complex network of the Humber region in a way that is understandable to business, policy makers and researchers alike.

As discussed in Section 4, the SBVR Logic Formulation prescribed in the OMG specification document for SBVR [26] can be used to transform an SBVR model into first order logic, which can be useful for reasoning by looking at the *FODAL* approach that proposed by [32] as discussed in Section 2. In the context of coordination, and service choreography, temporal aspects will need to be handled and possibly in a distributed manner as done in Mdtl [11] which we have used for reasoning about distributed and concurrent interactions before, or as done for decentralized self-adaptive systems in [14] where behavioural properties are specified using timed computation tree logic (TCTL). Hence, with respect to choreography verification we would typically be looking at Computational Tree Logic (CTL) or Linear Temporal Logic (LTL) [13] . LTL models time as a sequence of states, extending infinitely into the future. However, it does not allow to quantify explicitly over paths. CTL allows to reason about sequences of events that capture the semantics. The CTL syntax includes a parse tree, a quantifier equivalent to logical formulation kinds, and both CTL and LTL syntaxes denote a set of atomic propositions which is similar to the first order logic proposition used in the logical foundation for the SBVR model here.

In terms of implementation, apart from the SBVR editor [18] discussed earlier (Section 2) we also

implemented an SBVR to SQL comipler [20]. Using the Logical Formulation SBVR-LF [34], the complier maps business rules onto SQL queries to be executed on a (relational) database, as shown in Figure 6. The relational database is also automatically generated by the model. The work in [17] demonstrates

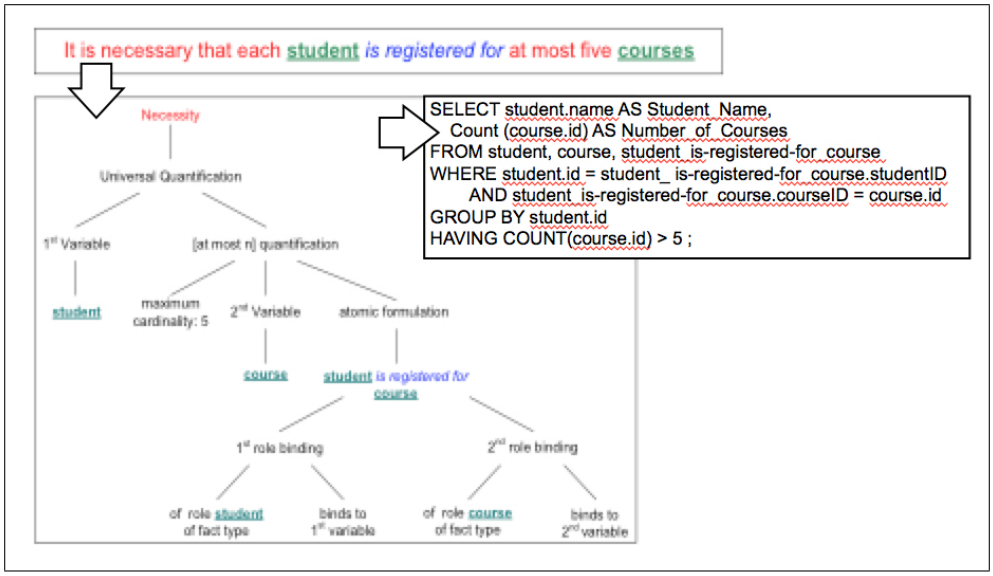

Figure 6: SBVR rules transformed to SQL queries [23]

how information systems can be generated directly from SBVR. Therefore, SBVR can be used to formulate complex data queries in a way that provides a higher level of abstraction than SQL (or any other query language) as shown in [23].

In previous work we have used *vector languages*, a model of true concurrency, to equip service choreographies with transactional guarantees, in the so-called *transaction languages* [22]. A natural extension of this work is to investigate the use of transaction languages, which is tuple-based formalism that captures the ordering of observable actions in a given choreography, to activate rules in the overlaid SBVR model of the service choreography. So transaction languages effectively play the role of the *blackboard*, in the sense of the work in [15], while the SBVR rules which are amenable to business analysts are used to constrain the generated implementation and seamlessly force it to adhere to the choreography specification [2].

One other possible direction for future work then will focus on analysing the complete set of behaviours (all possible outcomes) by exploiting the logical underpinning of the SBVR model of the choreography. This would target reasoning and choreography verification tasks such as *realisation* and *conformance*. In addition, another possible future extension has to do with the (correctness of the) transformation from natural language to SBVR [3]. Controlled Language (CL) is used in [16] to ensure correctness and deadlock-freedom. This would allow our approach to extend to natural language for specifying choreographies and would appeal to a wider business audience.

References

[1] James F. Allen & George Ferguson (1994): *Actions and Events in Interval Temporal Logic*. *J. Log. Comput.* 4(5), pp. 531–579, doi:10.1093/logcom/4.5.531.

[2] Marco Autili & Massimo Tivoli (2014): *Distributed Enforcement of Service Choreographies*. In: *Proceedings Int'l Workshop on Foundations of Coordination Languages and Self-Adaptive Systems, FOCLASA*, pp. 18–35, doi:10.4204/EPTCS.175.2.

[3] Imran Sarwar Bajwa, Mark G. Lee & Behzad Bordbar (2011): *SBVR Business Rules Generation from Natural Language Specification*. In: *Proceedings of AAAI Spring Symposium: AI for Business Agility*, pp. 2–8.

[4] Matteo Baldoni, Cristina Baroglio, Viviana Mascardi, Andrea Omicini & Paolo Torroni (2010): *Agents, Multi-Agent Systems and Declarative Programming: What, When, Where, Why, Who, How?* In: *A 25-Year Perspective on Logic Programming: Achievements of the Italian Association for Logic Programming, GULP*, pp. 204–230, doi:10.1007/978-3-642-14309-0_10.

[5] Tevfik Bultan & Xiang Fu (2007): *Specification of Realizable Service Conversations Using Collaboration Diagrams*. In: *In Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications, SOCA*, pp. 122–132, doi:10.1109/SOCA.2007.41.

[6] J. Camara, G. Salaun, C. Canal & M. Ouederni (2009): *Interactive Specification and Verification of Behavioural Adaptation Contracts*. In: *Proceedings of the 9th International Conference on Quality Software*, doi:10.1109/QSIC.2009.17.

[7] C. J. Date (2000): *What not How - The Business Rules Approach to Application Development*. Addison-Wesley.

[8] C. J. Date (2004): *An Introduction to Database Systems*. Addison-Wesley.

[9] Marlon Dumas, Murray Spork & Kenneth Wang (2006): *Adapt or Perish: Algebra and Visual Notation for Service Interface Adaptation*. In Schahram Dustdar, JosLuiz Fiadeiro & AmitP. Sheth, editors: *Business Process Management*, Lecture Notes in Computer Science, pp. 65–80, doi:10.1007/11841760_6.

[10] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo A. Reijers, Barbara Weber, Matthias Weidlich & Stefan Zugal (2009): *Declarative versus Imperative Process Modeling Languages: The Issue of Understandability*. In: *Enterprise, Business-Process and Information Systems Modeling, 10th International Workshop, BPMDS, and 14th International Conference, EMMSAD*, pp. 353–366, doi:10.1007/978-3-642-01862-6_29.

[11] Juliana Küster Filipe & Sotiris Moschoyiannis (2007): *Concurrent Logic and Automata Combined: A Semantics for Components*. In: *Proceedings Int'l Workshop on Foundations of Coordination Languages and Self-Adaptive Systems, FOCLASA*, Elec. Notes in Theort. Comp. Sci., pp. 135–151, doi:10.1016/j.entcs.2007.03.008.

[12] Howard Foster, Sebastian Uchitel, Jeff Magee & Jeff Kramer (2006): *LTSA-WS: A Tool for Model-based Verification of Web Service Compositions and Choreography*. In: *Proceedings of the 28th International Conference on Software Engineering*, doi:10.1145/1134285.1134408.

[13] Michael Huth & Mark Dermot Ryan (2004): *Logic in Computer Science - Modelling and Reasoning about Systems*. Cambridge University Press, doi:10.1017/CBO9780511810275.

[14] M. Usman Iftikhar & Danny Weyns (2012): *A Case Study on Formal Verification of Self-Adaptive Behaviors in a Decentralized System*. In: *Proceedings 11th International Workshop on Foundations of Coordination Languages and Self-Adaptive Systems, FOCLASA*, pp. 45–62, doi:10.4204/EPTCS.91.4.

[15] Jean-Marie Jacquet, Isabelle Linden, & Mihail-Octavian Staicu (2013): *On the Introduction of Time in Distributed Blackboard Rules Jean-Marieq*. In: *Proceedings 12th International Workshop on Foundations of Coordination Languages and Self-Adaptive Systems, FOCLASA*, pp. 144–203, doi:10.1007/978-3-642-45364-9_13.

[16] François Lévy & Adeline Nazarenko (2013): *Formalization of Natural Language Regulations through SBVR Structured English - (Tutorial)*. In: *Theory, Practice, and Applications of Rules on the Web - 7th International Symposium, RuleML*, pp. 19–33, doi:10.1007/978-3-642-39617-5_5.

[17] A. Marinos & P. Krause (2009): *What, Not How: A Generative Approach to Service Composition*. In: *3rd IEEE International Conference on Digital Ecosystems and Technologies*, pp. 115–120, doi:10.1109/DEST.2009.5276716.

[18] Alexandros Marinos, Pagan Gazzard & Paul J Krause (2011): *An SBVR Editor with Highlighting and Autocompletion*. In: *Semantic Web Rules - International Symposium, RuleML*, pp. 111–118.

[19] Alexandros Marinos & Paul J. Krause (2009): *An SBVR Framework for RESTful Web Applications*. In: *Rule Interchange and Applications, International Symposium, RuleML 2009*, 5858, LNCS, pp. 144–158, doi:10.1007/978-3-642-04985-9_15.

[20] Alexandros Marinos, Sotiris Moschoyiannis & Paul J Krause (2010): *An SBVR to SQL Compiler*. In: *Proceedings of the RuleML-2010 Challenge, at the 4th Int'l Web Rule Symposium*, 649. Available at `http://ceur-ws.org/Vol-649/paper7.pdf`.

[21] Marco Montali, Maja Pesic, Wil M. P. van der Aalst, Federico Chesani, Paola Mello & Sergio Storari: *Declarative Specification and Verification of Service Choreographies*. *ACM Transactions on the Web, TWEB* 4(1), pp. 3:1–3:62.

[22] Sotiris Moschoyiannis & Paul J. Krause (2015): *True Concurrency in Long-running Transactions for Digital Ecosystems*. *Fundamenta Informaticae* 138(4), pp. 483–514, doi:10.3233/FI-2015-1222.

[23] Sotiris Moschoyiannis, Alexandros Marinos & Paul J. Krause (2010): *Generating SQL Queries from SBVR Rules*. In: *Semantic Web Rules - International Symposium, RuleML*, pp. 128–143, doi:10.1007/978-3-642-16289-3_12.

[24] OASIS (2007): *Web Services Business Process Execution Language Version 2.0*. Technical Report. Available at `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`.

[25] OMG (2013): *Business Process Model and Notation (BPMN)*. Technical Report. Available at `http://www.omg.org/spec/BPMN`.

[26] OMG (2013): *Semantics Of Business Vocabulary And Business Rules (SBVR), V1.2*. Technical Report. Available at `http://www.omg.org/spec/SBVR/1.2/PDF`.

[27] OMG (2015): *Date-Time Vocabulary (DTV)*. Technical Report. Available at `http://www.omg.org/spec/DTV/`.

[28] OMG (2015): *Object Constraint Language (OCL)*. Technical Report. Available at `www.omg.org/spec/OCL/`.

[29] Alexandra S. Penn, Christopher J. Knight & et al. Georgios Chalkias (2015): *Extending Participatory Fuzzy Cognitive Mapping with a Control Nodes Methodology: A Case Study of the Development of a Bio-based Economy in the Humber Region, UK*. In Steven Gray, Michael Paolisso, Rebecca Jordan & Stefan Gray, editors: *Environmental Modeling with Stakeholders*, Springer. In press.

[30] R. G. Ross (2003): *The Business Rules Manifesto, Version 2*. Technical Report, Business Rules Group.

[31] Dmitry Solomakhin (2011): *Logical Formalization of Semantic Business Vocabulary and Rules*. Master's thesis.

[32] Dmitry Solomakhin, Enrico Franconi & Alessandro Mosca (2013): *Logic-based Reasoning Support for SBVR*. *Fundamenta Informaticae* 124(4), doi:10.3233/FI-2013-848.

[33] Jianwen Su, Tevfik Bultan, Xiang Fu & Xiangpeng Zhao (2007): *Towards a Theory of Web Service Choreographies*. In: *Web Services and Formal Methods, 4th International Workshop, WS-FM*, pp. 1–16, doi:10.1007/978-3-540-79230-7_1.

[34] W3C (2006): *Web Services Choreography Description Language*. Technical Report. Available at `http://www.w3.org/TR/ws-cdl-10-primer/`.