Noname manuscript No.
(will be inserted by the editor)

Engineering Commitment-based Business Protocols with the 2CL Methodology

**Matteo Baldoni, Cristina Baroglio, Elisa Marengo,
Viviana Patti, and Federico Capuzzimati**

the date of receipt and acceptance should be inserted later

Abstract Enterprises must respect a number of regulations, with multilevel nature and which change along time. They must not only adapt their business interactions to the regulations and their changes but also evaluate the risks of violation of the new rules and to account for responsibilities. This work proposes a methodological framework for modeling and engineering business protocols, which gives primary position to the notions of commitment and responsibility, and supports the analysis of risks of violation when a new regulation is issued. We build on 2CL commitment-based protocols and introduce 2CL Methodology, a software engineering methodology for such protocols, which includes guidelines for specifying 2CL business protocols, for specialising them, and for composing a new 2CL protocol based on a set of given 2CL protocols. We developed a set of integrated software tools for the design and the analysis of 2CL protocols, with the aim of concretely supporting, on the one hand, designers in the task of identifying exposure to risks of violation, and, on the other hand, the management in the task of reasoning about accountability and of decision making. The proposal is evaluated by using a real-world case study from the banking sector.

Keywords Commitment-based business protocols, regulations, methodologies, risks of violations, accountability

1 Introduction

Business protocols are a means for specifying the interaction of a set of autonomous parties with heterogeneous software designs and implementations. They have a normative value in that parties are expected to behave according to the protocol. In many practical settings, the reality in which such parties operate is characterized by a *high degree of regulation*, and the business relationships are increasingly constrained by the regulative and legislative framework. This is, for instance, the case of banking and of trading services.

As new regulations are issued, there is the need of adapting business protocols to the new dictates, which usually restrict – e.g. by adding new commitments and new constraints – the possible interactions or require the combination of different protocols. Think, for instance,

Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti, and Federico Capuzzimati
Università degli Studi di Torino, Dipartimento di Informatica
E-mail: {matteo.baldoni, cristina.baroglio, elisa.marengo, viviana.patti, federico.capuzzimati}@unito.it

to the regulations concerning privacy and to their impact in the most disparate sectors. When this happens, it is important for those organizations, whose conducts are affected by the new regulation, to have the means for identifying their *exposure to risks of violation*. Concretely, organizations need to understand how new regulations impact on their business protocols, to reason about the risks connected to possible violations, and to ensure compliance to directives and laws. In other words, they need to be supported to answer to the following questions: What is the impact of the new regulations on the business protocols in use? Is the organization exposed to risks of violation, when behaving according to the current business protocols? How to graft a new regulation into a business protocol? Which changes to implement, considering that modifying a business protocol can be costly? In order to support answering such questions, it is necessary to provide organizations with *modeling and design tools* that allow dealing with business protocol changes in a natural way.

In this paper we introduce a *software engineering methodology* for the business protocol language 2CL, named 2CL Methodology. 2CL [6,4] allows specifying business protocols in a declarative way by means of social commitments. The commitment-based approach took hold within the research community working on multi-agent systems [58,20,24,11, 21]. It allows expressing interaction in terms of actions, whose social meaning is shared by the interacting parties. The execution of such actions affects the world and can either cause the creation of social commitments between the parties or the manipulation of already existing ones (e.g. by delegating, canceling, assigning them). A key feature of commitment-based approaches is that they naturally capture the contractual relationships among the partners rather than strictly encoding the order in which messages are to be exchanged. 2CL extends classical commitment-based protocols, like [56,57], with the possibility to express temporal *constraints among commitments*.

2CL Methodology includes specific guidelines for the *composition* and the *specialization* of protocols. The latter is used in the case in which a protocol must take in a new regulation or, as we say, a regulation is to be *grafted* upon an existing protocol [7,5]. The proposed methodology extends the Amoeba methodology, described in [17], in two ways: first, it allows tackling 2CL protocols; second, by exploiting the notion of constraint supplied by the 2CL language, it generalizes the notions of "data flow axiom" and "event order axiom", that Amoeba borrows from [19], in a way that realizes closure under the operations of composition (which does not hold for Amoeba, in that the result is not a protocol) and specialization (which is not provided in Amoeba). Briefly, data flow axioms specify the data flow that occurs during an interaction, while event order axioms specify temporal dependencies between the executions of the protocol actions.

As an example of usage, we apply the 2CL Methodology to a real-world case study, the Markets in Financial Instruments Directive (MiFID) which is a prominent example of EU regulation having an impact on trading procedures, and which is also one of the benchmarks of the ICT4LAW project (`http://www.ict4law.org`). We report a sales protocol that was legal before MiFID was issued, and we show how the proposed methodology supports the protocol designer in grafting the new regulation upon the old specification. The 2CL Methodology was evaluated by interviewing a group of volunteers of different expertise and experience after explaining them the methodology itself.

Finally, we describe an integrated set of software tools, developed for supporting the design and the analysis of 2CL business protocols, and in particular to decide which changes to implement when new requirements emerge. Broadly speaking, the tool identifies the risks the interaction could encounter as the possible violations that the players can perform, i.e. as the commitments that are not satisfied and constraints that are broken by some of the players. When the evaluation is done *a priori*, the designer can use the results to define some opera-

tional strategies, that affect the business protocol by, alternatively, preventing the occurrence of violations (*regimentation*) or by implementing alerting mechanisms (*enforcement*) [29]. When the identification of risks is performed *as the interaction occurs*, it allows an immediate intervention and, in some circumstances, also recovery from violations. To support the described tasks, the software provides the following functionalities: supporting the *design* of 2CL business protocols; *reasoning* about all possible interactions; highlighting the *risks of violations*. Specifically, the tool allows the *visualization and exploration* of the possible interactions, represented as a labelled graph, and supports risk analysis and cost-benefit analysis. The reasoning engine is an extension of Winikoff et al.'s enhanced commitment machine [53]. The implementation is done in *tuProlog*; the software interprets 2CL business protocols by means of a parser written in Java. The verification procedures that are implemented are described in [3] and rely on the operational semantics of 2CL which is given in the same paper.

Contributions. First, we defined the 2CL Methodology, a commitment-based business protocol engineering methodology which provides two fundamental features: the *composition of* 2CL *business protocols*, and their *specialization*. The latter allows tackling the issue of grafting new regulations inside business protocols. The methodology is based on Amoeba [17], which was refined and suitably extended. Second, we implemented a set of integrated software tools which features 2CL protocol design and analysis. Third, as a proof of concept we applied the proposal to a real-world case study: the MiFID Directive, which, in the European Union, applies to the offer of investment services in a financial domain.

Organization. The paper is organized as follows. Section 2 overviews the relevant literature concerning business protocol specification, together with the limits of current proposals, and reports practical motivations to this proposal. Section 3 briefly introduces the 2CL language with the help of examples and of UML diagrams. Section 4 describes the 2CL Methodology for engineering 2CL protocols. Section 5 introduces the MiFID case study and reports the application of the proposed methodology to it. Section 6 reports the results of the evaluation of the methodology and analyses them. Section 7 describes a set of integrated software tools for supporting the protocol design and analysis. Conclusions end the paper.

2 Background and motivations

Business interactions are often modeled, in the real world, by means of *business processes*. Companies adopt business process management systems (BPMS) to manage their flow of activity [51]. Traditionally, the models and languages for business processes that can be managed by such systems (e.g. BPEL, BPMN) are quite rigid, and this rigidity is a limit that prevents BPMS to tackle (rapidly) evolving processes [40]. As Pesic and van der Aalst further observe in [40], it is possible to find in the literature approaches for overcoming such a limit – e.g. the case-handling paradigm [52] and the adaptive workflow management systems [42,30] – but all such approaches exploit modelling techniques which rely on the notion of *procedural specification*: the specification defines which actions are executable at each moment; what is not explicitly foreseen is forbidden.

This procedural view makes business processes not suitable to easily take in new requirements because the composition techniques, that can be applied, besides imposing *unnecessary orderings* of the interactions with respect to what foreseen by the requirements,

cannot easily manage those *new activities*, whose execution is requested by the new requirements, and which are to be interleaved with the previously existing ones. As a consequence, workflow-like representations by and large require to rewrite the business process from scratch.

Pesic and van der Aalst proposed a shift of paradigm, passing from a procedural to a *declarative representation*. Specifically, they proposed *Declare* (formerly known as ConDec) [38,39,50] as a *declarative language* for business process representation. Briefly, Declare allows specifying "what" should occur rather than "how" making it occur. This change of perspective allows avoiding the over-specification of business processes, limiting the representation to what is mandatory as a set of relationships between tasks. While in the procedural approaches, e.g. in Petri nets, relationships between tasks capture a strict order of execution, in Declare relationships define *constraints of execution*, a softer notion. For instance, a constraint can capture the case where a task A must occur before a task B and before a task C but no order is given between B and C. This can be done without the need of explicitly prescribing the alternative sequences "B followed by C" and "C followed by B" but yielding to a more compact model. Relations between tasks are modeled in Linear Temporal Logic (LTL). Constraints can be created with the help of constrain templates, which also have a graphical representation. The enactment of Declare business processes is possible thanks to the fact that LTL formulas can be translated into automata. A deeper discussion of the approach and of the possibility of performing verification tasks on the executed processes can be found in [36,35].

Declare is very attractive because of the use of constraints for specifying relationships between tasks and activities, and because of the availability of a easy-to-use graphical language, however, Declare is *not oriented to cross-business processes*. Nevertheless, in the real world, tasks and activities may cut across enterprise boundaries and involve mutually independent partners, but there is no way for accounting for this characteristic in Declare: how to enact a task in this case? How having control?

Another declarative approach, which instead specifically tackles the problem of representing *cross-organizational* business processes, is the one by Telang and Singh [47]. As in the previous case, the importance of a flexible representation is underlined: what matters is the business intent, not the specific enactment. The approach relies on the *agent metaphor* [54] and on the notion of *commitment* [44], which is used to capture the business relationships between autonomous partners (the essence of business interactions), leaving aside the operational details. A commitment specifies that a party is a debtor towards a business partner (the creditor) in bringing about a certain condition of interest if a given context will hold. It has a *social* and *normative* (regulative) value. This kind of specification constrains the behavior of the involved organizations: the expectation is that the debtor will do as agreed. The way in which the debtor will act in order to achieve the condition is not specified.

Commitments naturally suit the representation of *business protocols* because they allow a flexible specification of the business intents and of the business relationships among the parties [18,47]. In other words, they give to the protocol actions and messages a meaning in terms of *contractual relationships* among the partners rather than a strict encoding of the order in which messages are to be exchanged. For instance, see [47], in a sales protocol what matters is that the client satisfies the commitment to pay and the merchant the commitment to deliver the goods, rather than how the interaction is executed. This change of perspective is important to enable a *flexible enactment* so as to allow the business parties, who are heterogeneous, autonomous, and basically self-interested, to find the way of interacting that better suits their characteristics and requirements. For this reason, commitments provide a solid basis for representing cross-organizational business protocols, in a way that respects

the partners' autonomy. Moreover, thanks to their *social semantics*, commitments are inferable from the observable behaviour, without any need to inspect the internal policies of the parties. With respect to Declare, they have the advantage of formally capturing the *responsibility* of specific partners in doing something, and of being verifiable by all the involved parties; thus, they allow introducing a notion of *violation*.

In [47] the authors explain how to build a commitment-based business model, based on a set of predefined commitment patterns, showing the high *modularity* of the approach. However, such business models amount to mere sets of commitments, constraining the behavior of the partners only so far as commitment satisfaction is concerned. This is the only regulative norm that is accounted for. Very often cross-business interactions require to express more sophisticate regulative norms: they must allow to specify sets of desired patterns of interactions. For example, in a trading setting, the commitment to pay is to be created before the commitment to ship the goods is created. Moreover, such patterns of interactions should have an explicit regulative flavor, because only this characteristic guarantees that either the parties will behave as described or a violation will be raised and detected. This is particularly true in those cases when regulations are expressed on observable events and on shared social expectations.

One further need is to have the capability of dealing with *changes* to the protocol, changes that are necessary when new requirements or new regulations are issued. In such cases, the model that is used should allow an easy modification of the business protocols. This need is even more important in contexts where requirements frequently change along time. In this case, there is the need of a *flexible* specification, where flexibility is to be understood as the capability of answering to the new requirements without the need of redesigning the business protocol from scratch [41]. This kind of modification of protocols often takes the form of a *specialization* or of a *composition*. Intuitively, we say that it is a specialization when further requirements are grafted into an existing business protocol, while it is a composition when the business protocol is obtained by combining two or more fully specified existing protocols. Let us consider the following example.

Example 1 – A new regulation is issued. Let us consider a telephone company and a new law on transparency of data management, which requires that clients must be informed about how their personal data will be used and accept this procedure before signing the contract. The new law has a strong impact on the sales procedures of the telephone company (both at desks and on-line): the procedure followed so far (explaining the economical issues and asking the client to sign the contract) is not legal anymore. The company needs to revise its sales procedure so as to include the new actions foreseen by the law (explaining the way in which personal data will be used and ask for acceptance) and take care that such actions are performed before the client signs the contract. Whether explaining such clauses before or after the economical issues is up to the company.

A flexible specification of the telephone company sales procedure would allow an easy specialization of it, taking in both the new activities and the activity orderings imposed by the regulation.

Our claim is that *commitments* and *constraints* supply the right abstractions for capturing the intents of the specification without being overly prescriptive: the former, by capturing contractual engagements between the interacting parties, the latter, by expressing agreements, norms, conventions, habits and such like on the evolution of the interaction. Moreover, we claim that constraints allow the representation of temporal relationships between action execution and of the data flow. Considering *Example 1*, on the one hand, it is easy to represent the interaction described in terms of commitments: the presentation of the clauses

concerning the uses of personal data as well as the clauses concerning economical issues correspond to commitments by the telephone company towards the client to accomplish the depicted conditions. On the other hand, the new law extends the original business protocol by adding new activities and by regulating the order in which these and the signature of the contract are to be executed. It is easy to capture the new ordering imposed by the law by exploiting temporal constraints.

Specifically, the proposal contained in this article builds upon the 2CL proposal for the representation of declarative business protocols [6]. 2CL extends classical commitment-based approaches, e.g. [56,57,53,12,8], by explicitly accounting for temporal constraints that are to be respected by the parties. Constraints are inspired by the Declare approach but their application is extended to commitments. As for Declare, 2CL is equipped with a graphical notation. Being declarative, 2CL allows the realization of compact models: they specify what is desired and undesired, leaving all that remains unconstrained. This is an advantage with respect to procedural approaches, characterized by a prescriptive nature which requires the specification of *all* the allowed evolutions.

In order to tackle the complexity of the analysis and the design of evolving business protocols, it is necessary to rely on a proper modeling methodology. To this aim, we propose an enhanced version of the Amoeba methodology [17]. Amoeba is specifically thought for realizing commitment-based business protocols, either designing them from scratch or by adapting already existing ones. The version that we propose is conceived for tackling not only commitments but also temporal constraints, the two chief components of 2CL specifications. It also includes steps that are specifically conceived for performing the *composition* and the *specialization* of business protocols, the latter of which is used for performing the grafting of regulations.

3 2CL Business Protocols

A 2CL business protocol models business interactions based on the two main notions of commitment and of constraint. The specification relies on the approach discussed in [4,6]. A 2CL protocol is an aggregation of various elements; Figure 1 provides an overview:

- *Role*: conceptualizes a possible actor in a protocol. Roles are played by specific actors when protocols are enacted;
- *Domain Element*: is an element of the *universe of discourse*;
- *Initial State*: is a set of items belonging to *Domain Element* and represents the initial state of an interaction that respects the business protocol;
- *Action*: represents a move that players of roles may execute. It conceptualizes the "counts as" [43,29] relationship between the move at issue and its social meaning, given in terms of modifications to the social state;
- *Constraint*: conceptualizes an interaction pattern that is to be respected.

Actions and constraints are both defined based on *facts* and *commitments* which are part of a set of *Domain Elements*. Facts are positive or negative propositions that do not concern commitments and which contribute to the social state (they are the conditions that are brought about); they are also used to represent action occurrences/events and they include $\top$. So, for instance, the occurrence of the action *pay-by-credit-card* can be represented by the simple fact *paid* when this is sufficient. On the other hand, a commitment to pay can be satisfied by all action executions that correspond to different forms of payment [6]. Commitments [44] are represented with the notation $\mathsf{C}(x,y,r,p)$, capturing that the agent x commits

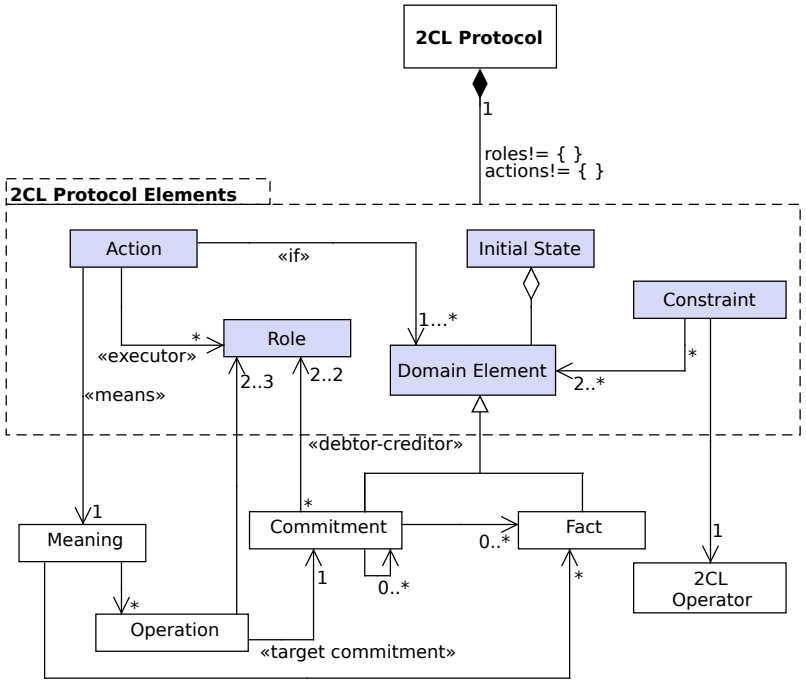

Fig. 1 UML diagram of the protocol specification. Coloured boxes denote protocol components. They are defined in terms of the other depicted elements.

to the agent y to bring about the consequent condition p when the antecedent condition r holds. Antecedent and consequent conditions generally are conjunctions or disjunctions of facts and commitments, see Figure 1. When r equals $\top$, we use the short notation $\mathsf{C}(x,y,p)$ and the commitment is said to be *active*. Commitments have a *regulative* nature, in that debtors are expected to behave so as to satisfy the engagements they have taken. This practically means that an agent is expected to behave so as to achieve the consequent conditions of the active commitments of which it is the debtor.

The business partners share a social state that contains commitments and facts. Some of these elements can hold since the beginning of the interaction. Every partner can affect the social state by executing actions, whose definition is given in terms of the standard operations on commitments, i.e. *create*, *cancel*, *release*, *discharge*, *assign*, *delegate*. Briefly

- $create(\mathsf{C}(x,y,r,p))$ is performed by x, and it causes $\mathsf{C}(x,y,r,p)$ to hold.
- $cancel(\mathsf{C}(x,y,r,p))$ is performed by x, and it causes $\mathsf{C}(x,y,r,p)$ to not hold.
- $release(\mathsf{C}(x,y,r,p))$ is performed by y, and it causes $\mathsf{C}(x,y,r,p)$ to not hold.
- $delegate(\mathsf{C}(x,y,r,p),z)$ is performed by x, and it causes $\mathsf{C}(z,y,r,p)$ to hold and $\mathsf{C}(x,y,r,p)$ to not hold.
- $assign(\mathsf{C}(x,y,r,p),z)$ is performed by y, and it causes $\mathsf{C}(x,z,r,p)$ to hold and $\mathsf{C}(x,y,r,p)$ to not hold.

As in [44], we postulate that discharge is performed concurrently with the actions that lead to the given condition being satisfied and causes the commitment to not hold. Delegate

and assign transfer commitments respectively to a different debtor and to a different creditor. For details see [44,56,10]. According to the commitment life cycle reported in [46], a commitment can be in one of the following states: null, conditional, base, active (grouping conditional and base), satisfied, or violated. Before a commitment is created, it is null. If the antecedent of a conditional commitment is brought about then the commitment is detached and its state becomes base. Conditional and base commitments are active. An active commitment is satisfied when its consequent condition is brought about. In this case the commitment is discharged. It is violated when it is a base commitment and it cannot be discharged because the consequent condition cannot be brought about. A survey comparing and commenting this and other commitment life cycles can be found in [9].

Action is meant to capture a physical event that is relevant for the interaction which is being modelled; they are not limited to utterances. Without losing generality, every action name is associated to one *Role* in the protocol (the possible executor). Each action is also associated to a *counts-as* relationship which captures the *Meaning* of the physical event in terms of modifications to the social state. These are given in terms of *Operation*s performed on *Commitment*s and in terms of *Fact*s that are achieved. Additionally, by means of a condition (**if**), expressed in terms of *Domain element*s, it is possible to specify the context in which the execution of the action acquires the specified social meaning. For the sake of readability, we write an action definition as:

$$action \textbf{ means } meaning \textbf{ if } context$$

Meaning is expressed as a a set of facts and operations on commitments. All actions have a social effect, either implicit or explicitly represented: the implicit effect is that their execution has an impact on existing commitments. In fact, when the antecedent or the consequent condition of a commitment contains a fact, representing an action occurrence, the commitment respectively is detached (its state becomes base) or discharged (its state becomes satisfied). Similarly, the execution of an action can activate the context of another one or help satisfying a constraint. Moreover, meaning can also explicitly include operations on commitments, like creation, delegation, or release, or facts. Such operations (or facts) appear after the reserved word *meaning*. In 2CL, the antecedent or consequent conditions of a commitment may contain fact labels. When a fact captures that an action occurred, by convention, the fact label is the same as the corresponding action name. The same happens for fact labels appearing in the meaning or in the context of some action. See, for instance, Example 2. When the only social effect of an action is its occurrence, for the sake of a simpler representation (explicitly specified) meaning becomes conventionally *none*. In order to be well-formed, a 2CL protocol must have a non-empty set of actions and a non-empty set of roles.

Example 2 – Purchase. Consider a purchase protocol which includes the following actions. Here *merc* denotes the merchant, while *cust* denotes the customer:

(a) $offer_{merc}$ **means** CREATE(C(*merc*, *cust*, *buy*, *give_item*))
(b) buy_{cust} **means** CREATE(C(*cust*, *merc*, *pay*))

Intuitively, the act of a merchant of offering some item to a client has the social meaning (shared by the client and the merchant) of creating a commitment, by which the merchant binds herself toward the client: she will give the item to the client if the client buys it. "Buying an item" is an action by which the client creates an unconditional commitment toward the merchant to pay for the item being bought. Figure 2 shows the contractual relationships between the two roles. Arrows are directed from the debtor towards the creditor.

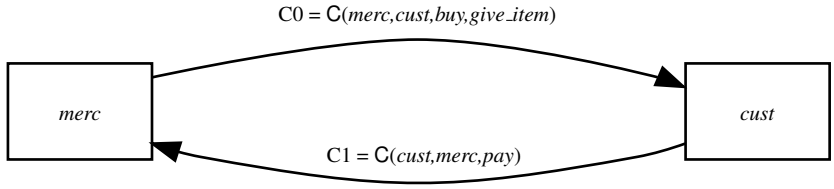

Fig. 2 Relationship diagram involving a merchant ad a customer, according to the actions reported in the Example 2.

The commitment which creates a relationship between a debtor and a creditor labels the corresponding arc.

Notice that if, in the meaning of action *offer* we replace buy by pay in the antecedent condition of the commitment C(merc, cust, buy, give-item), leading to C(merc, cust, pay, give-item) the business relationship between customer and merchant would be different. The commitment would become active only after payment. The merchant may decide to give the item to the customer even before payment occurs and, due to the commitment life cycle, the commitment would be satisfied. In this case, however, the customer would have no commitment to pay. If he/she does not pay, he/she would not be liable of a violation.

Besides actions, a 2CL protocol includes a *set of constraints*, that are expressed adopting the *Constraints among Commitments Language* (2CL) [4, 6]. This language supplies different kinds of *constraints* which allow the designer to characterize the legal evolutions of the interaction, by expressing mandatory and forbidden behaviours, without the need of listing the possible executions extensionally. Constraints relate two conditions, that are expressed as formulas of *Domain element*s. Accordingly, a constraint has the form:

$$dnf_1 \ \mathsf{op} \ dnf_2$$

where dnf_1 and dnf_2 are Disjunctive Normal Form formulas of facts and commitments and op is one of the operators supplied by the language. Table 1 reports the lists of 2CL

	Relation	*Type*	*Operator*	*Meaning*
Relational Operators	Correlation	pos.	$A \bullet\!\leftarrow B$	In an execution where A occurs, also B must occur but there is no temporal relation between the two.
		neg.	$A \bullet\!\not\leftarrow B$	If A occurs in some execution, B must not occur.
	Co-existence	pos.	$A \bullet\!\!-\!\!\bullet B$	Mutual correlation: both $A \bullet\!\leftarrow B$ and $B \bullet\!\leftarrow A$ hold.
		neg.	$A \bullet\!\not\leftarrow\!\bullet B$	Mutual exclusion of A and B: both $A \bullet\!\not\leftarrow B$ and $B \bullet\!\not\leftarrow A$ hold.
Temporal Operators	Response	pos.	$A \bullet\!\!\rightarrow B$	If A occurs, B must hold at least once afterwards (or in the same state). It does not matter if B already held before A.
		neg.	$A \bullet\!\not\rightarrow B$	If A holds, B cannot hold in the same state or after.
	Before	pos.	$A \rightarrow\!\bullet B$	B cannot hold until A becomes true. Afterwards, it is not necessary that B becomes true.
		neg.	$A \not\rightarrow\!\bullet B$	In case B becomes true, A cannot hold beforehand.
	Cause	pos.	$A \bullet\!\!\rightarrow\!\bullet B$	It is the conjunction of the base *response* and base *before* relations: $A \bullet\!\!\rightarrow B$ and $A \rightarrow\!\bullet B$.
		neg.	$A \bullet\!\not\rightarrow\!\bullet B$	It is the conjunction of the base *response* and base *before* negative relations: $A \bullet\!\not\rightarrow B$ and $A \not\rightarrow\!\bullet B$.

Table 1 2CL operators and their intuitive meaning.

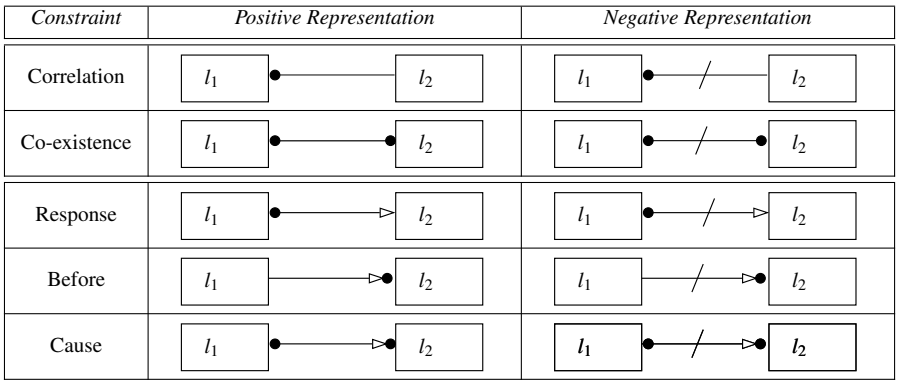

Fig. 3 Graphical representation of 2CL constraints.

operators together with their interpretation. Basically, the language foresees two kinds of constraints: *relational* and *temporal*. Relational constraints capture relationships on the co-occurrence of different conditions, allowing to express that the achievement of a certain condition forbids or requires, sooner or later, the achievement of another. So, for instance, $A \bullet\!\!- B$ (A correlates B) denotes that if condition A is achieved along an interaction, also condition B must occur. In different words, if at some point A holds in the social state, at some other point of the interaction B must hold; it does not matter whether B holds before, after, or at the same time as A. Temporal constraints, instead, capture a relative order at which different conditions should be achieved. They can be used to express, for example, that a certain condition should be acquired before another or it is required to hold at least once after the achievement of the former. For instance, $A \multimap B$ (A before B) denotes that A should be achieved before B, although other conditions can be achieved between the two. Notice that this is different than the procedural approach. Suppose, for example, that A and B are to be executed before C, no matter in which order. In 2CL it is possible to express this requirement by the constraint $A \wedge B \multimap C$, while in a procedural representation it would be necessary to explicitly account for the two alternative executions A,B,C and B,A,C. Moreover, the declarative representation supplied by 2CL allows agents to execute other actions at will, in between A, B, and C as long as the constraint is satisfied. In the procedural approach all the possible additional action executions should be explicitly listed.

The set of constraints of a business protocol expresses a set of *patterns* the interaction is desired to respect. Among the *possible* interactions, derivable from the actions specification, those that violate some constraint can be detected during the interaction and potentially sanctioned. In 2CL protocols, a *violation* is raised when a 2CL constraint is not satisfied or when, at the end of an interaction, some unsatisfied active commitments are left [3].

2CL allows representing constraints as a graph. Such graph captures the *flow of the interaction* that is implemented by the protocol, showing in an intuitive way the interactions that are allowed [4]. The arrows give the perception of a flow of activity although the graph does not account for actions but only for social state contents. This abstract representation is an expression of the no-flow-in-flow motto, introduced in [6]. The representation is inspired by ConDec [40] and by DCML [1] and it follows the graphical convention, which is reported in Figure 3. Each operator is represented with a different symbol, where: *(i)* the direction of the arrow (if present) denotes the *temporal nature* of the constraint, expressing a relative order on the achievement of two conditions. Arcs without arrows are used for representing

relational operators that do not specify temporal requirements; *(ii)* the position of the dot denotes the occurrence of which condition activates the constraint (*triggering condition*); *(iii)* negated operators are crossed by a line. For instance, consider the *before* constraint $a \rightarrow\!\!\!\!\bullet\ b$: the notation captures a temporal relation by which condition a should occur before condition b; the dot tells us that the triggering condition is b. Therefore, the constraint can be read as: *if* condition b is achieved, condition a must have occurred *before* it. Disjunctive Normal Form formulas involve commitments and facts. Basically, commitments and facts are represented as boxes. To define a conjunction, boxes are connected to a circle. Similarly, disjunctions are represented by connecting boxes to diamonds, exclusive disjunctions to bordered diamonds.

Example 3 – Constraints on Purchase.

The two actions reported in Example 2 lead to the creation of two commitments: one from the merchant toward the client to give her the item if she decides to buy it; the other from the client toward the merchant to pay for the item. The acts of paying and of giving the item can occur in any order. All that is requested is that both are executed, sooner or later, otherwise the respective commitment will be violated. Let us see how constraints can be used to further regulate the interactions by adding the following:

(c1) $\mathsf{C}(cust, merc, pay) \rightarrow\!\!\!\!\bullet\ \mathsf{C}(merc, cust, give\_item)$
(c2) $\mathsf{C}(merc, cust, give\_item) \bullet\!\!\!\!\leftarrow \mathsf{C}(merc, cust, send\_receipt)$

By constraint (c1), the purchase protocol additionally requires that before the merchant commits to give the items to the customer, this should have committed to pay for them. This constraint does not require the two conditions to be achieved one next to the other. If the protocol includes other actions (e.g. wrap the item in a gift box), it would be possible also to execute some of them in between. All that is requested is that the relative order between the achievement of the two conditions is respected. By constraint (c2) the protocol specifies that if the merchant takes the commitment of giving the items to the client, she is also expected to send, sooner or later, the receipt. Figure 4 shows the graphical notation for such constraints.

4 2CL Methodology

2CL protocols specify in a declarative way a set of possible legal interactions. Specifications result to be flexible and modular, thus fostering protocol reuse, speeding up their definition, and simplifying their maintenance. Nevertheless, the analysis that brings to the definition of a business protocol can be non trivial. In this section we present 2CL Methodology, a methodology that supplies guidelines to guide a protocol designer through the tasks of:

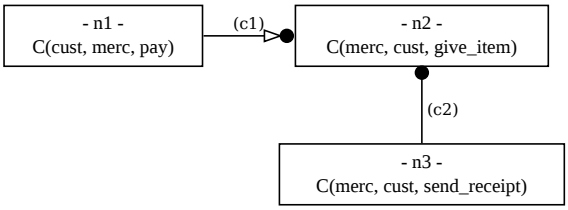

Fig. 4 Graphical representation of the Purchase Protocol's constraints reported in the Example 3.

(i) protocol specification; (ii) protocol composition; and (iii) protocols specialization. 2CL Methodology originates from the Amoeba methodology introduced by Desai *et al.* in [17] for commitments protocols. We adapted and extended it in order to suit for 2CL protocol specifications. Briefly, the main steps of the 2CL Methodology are:

M1. *Roles Identification*. Identify the set of roles involved in the interaction to be modelled. The identification of roles includes the identification of the actions they can execute.
M2. *Contractual Relationships Identification*. Identify the domain elements that are relevant for the interaction to be modelled; represent them in terms of facts and commitments; determine which of the individuated domain elements are to be included as content of the initial state.
M3. *Identify Actions Social Meanings*. Provide the definitions of the protocol actions in terms of their effects on the social state.
M4. *Identify Constraints*. Identify the temporal and relational requirements to be represented; express them in terms of 2CL constraints.
M5. *Reuse Protocols through Composition or Specialization*.

Such steps are followed for specifying new 2CL protocols and provide a guideline for both composition and specialization of 2CL protocols, as shown in the remainder of the section. The 2CL Methodology differs from Amoeba in steps M4 and M5. M4 tackles 2CL constraints, which are not used for the specification of protocols in Amoeba. For what concerns M5, Amoeba does not tackle specialization while, concerning composition, the object that Amoeba returns after the composition of two protocols is not a protocol, it is a protocol plus a set of additional axioms, which are necessary to capture both the data flow occurring in an interaction and temporal dependencies between the protocol actions. On the contrary, 2CL Methodology produces new protocols because it relies on constraints which are part of the definition of 2CL protocols, and have a regulative nature. A composed (or specialised) protocol can be further specialized or further composed with other protocols.

4.1 Protocol Specification

When defining a 2CL protocol, the designer should identify a set of roles, a set of domain elements, the content of the initial state, a set of actions, and a set of constraints. In the following we describe how such steps can be addressed, by exploiting the following example:

Example 4 A merchant performs on-line sales, asking clients to pick up their items at the shop. When a purchase is done, the merchant commits towards the client to retrieve the bought items and to bring them to the shop; the client commits towards the merchant that once they will be informed that the bought items are available at the shop, they will go there to pick them up.

***M1:* Roles Identification.** The set of roles of a protocol is obtained, first, by identifying the set of participants to the interaction that it is of interest to represent; second, by abstracting from players the roles they play in the protocol.

In our reference example we individuate two actors, playing respectively the roles of *merchant (merc)* and *customer (cust)*. For instance, the customer can execute *order* (some item) and *withdraw* (an order).

***M2:* Contractual Relationships Identification.** This step is structured in three substeps which basically aim at determining:

M2.1 the domain elements (or universe of discourse) on which the protocol relies;
M2.2 the contractual relationships and conditions of interest that are involved in the interaction;
M2.3 the contractual relationships and conditions of interest that exist prior to the interaction and that belong to the initial state.

In order to define the domain elements, the analyst is required to identify the contractual relationships the different roles are involved in during the interaction, and how they can be manipulated. Considering the reference example, the merchant engages in bringing the items to the shop once the order is done and, by ordering some item, the customer takes the engagement of paying for them. In case the customer changes her mind by cancelling the order, the merchant's engagement is released. For what concerns the customer, instead, she is expected to pick up the items from the shop when they are ready.

The identified contractual relationships are, then, represented by the analyst in terms of commitments and of operations on commitments. Those individuated for our example can be represent by the commitments:

(a) C(*merc*, *cust*, *order*, *prepare_item*);
(b) C(*cust*, *merc*, *item_ready*, *pick_up_item*);
(c) C(*cust*, *merc*, *item_ready*, *pay*).

Other conditions of interest are represented as facts. So, for instance, if it is of interest to record that the merchant has sent copy of the receipt to the customer, one could use the fact *sent_receipt*.

Some of the domain elements can be assumed to hold before the interaction starts. In case of commitments they capture contractual relationships a participant commits to when accepting a role in the protocol. In case of facts, they can be understood as conditions of interest assumed to hold since the beginning. The analyst should decide which of the domain elements to include in the starting state. By accepting the role of merchant, for instance, an agent accepts the engagement of preparing the items once they are ordered. Therefore, the commitment C(*merc*, *cust*, *order*, *prepare_item*) is expected to hold from the start.

M3: **Identify Actions Social Meanings.** This step of the methodology aims at identifying the social meaning of the protocol actions. In order to accomplish this task, the analyst considers the set of commitments and facts individuated at the previous step. By following the information she has extracted on their creation, satisfaction and manipulation, she determines which actions have an impact (and which) on the set of domain elements. Accordingly, she associates to each action a meaning, given in terms of operations on commitments or assertion of facts. Finally, the analyst determines whether the specified meaning holds only in a given context [29]. If so, the condition is added to the action definition.

In our example, given the commitments individuated at the previous step, the protocol should foresee the action *order*, by means of which a customer commits to the merchant to pick up the items at the shop when they are ready and commits to pay for them.

order **means** CREATE(C(*cust*, *merc*, *item_ready*, *pick_up_item*)),
CREATE(C(*cust*, *merc*, *item_ready*, *pay*)).

Moreover, the customer is allowed to change her mind. To this aim, the protocol should include the action *withdraw* by means of which the merchant's commitment of preparing the item is released and the customer's commitment of picking them up is cancelled. This meaning of the action holds only in case the items are not ready yet. If this is not the case, the customer cannot withdraw from its engagement of picking up the items:

withdraw **means** RELEASE(C(*merc*, *cust*, *order*, *prepare_item*)),
CANCEL(C(*cust*, *merc*, *item_ready*, *pick_up_item*)) **if** ¬*item_ready*

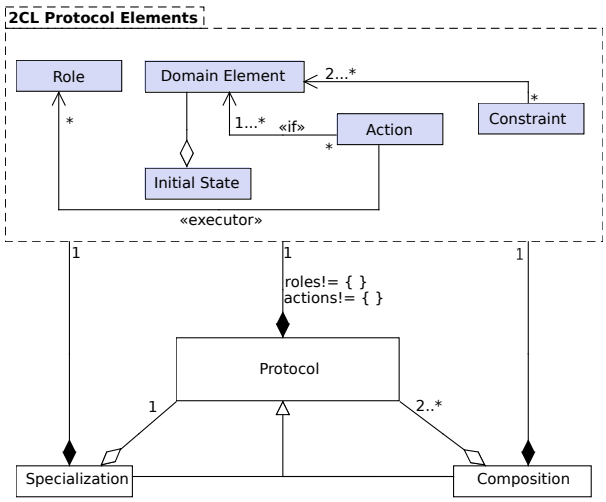

Fig. 5 UML diagram of protocol composition and specialization.

***M4:* Identify Constraints.** In this step of the methodology, the analyst is expected to identify a set of temporal requirements, or co-occurrence relations, the interaction is desired to respect and to find a suitable representation in terms of 2CL constraints. While accomplishing this task, the analyst should not confuse those requirements, that are to be expressed in terms of constraints, with those that are to be captured as "if" conditions inside the action definitions. While the latter restricts the context in which an action acquires a certain meaning, constraints specify a "requirement" on the use of the actions. When an action is executed out of the context, which is specified in its "if" condition, the other agents will not associate to its occurrence the meaning that is reported in the action definition. When an action is executed in a way that violates a constraint, instead, a violation is raised.

Considering our example, the merchant is free to prepare items even before they were ordered (as usually happens in the shops), while a customer can pick them up at any time. However, it is of interest for a merchant that when a customer arrives at the shop to pick up some item, she has already paid for them or, at least, has taken the commitment to pay. This requirement can be expressed by means of the constraint:

$$\mathsf{C}(cust, merc, pay) \vee \ pay \rightarrowtail pick\_up\_item$$

4.2 Protocol Composition

Composition (see Figure 5) combines different specifications so as to reach a more complex design objective. When combining different specifications, however, there is the need to define how they are connected with one another. The main issues are: how to define the way in which the composed business protocols are intertwined with one another? How to define a mapping between the elements of different business protocols? Which elements are to be added and which should be modified? In the following we call *component* protocols the starting 2CL protocols that the analyst wants to combine. The result is called *composite* protocol. We use the following example for explaining the steps.

Example 5 – Setting up a cross-business interaction. Suppose that the merchant from Example 4 agrees with a shipping company for supplying a new service to the customers: delivery at home. Both the merchant and the shipping company have their protocols to interact with their respective clients, but when they start their joint venture they need to combine them in order to supply the delivery at home service.

Here, the person who bought the items to be delivered may be identified by the merchant as the "client" and, by the shipping company, as the "addressee". Similarly, the act of shipping may be identified as "deliver" by the shipping company and "send" by the merchant. These examples show the need for composition to account for *mappings* between elements belonging to different protocols. Such mappings can be realized by exploiting standard techniques for ontology matching, e.g. [25]. Moreover, composition should allow for the definition of additional interactions between the parties, that were not in the original protocols and that act as "glue" between them. For instance, in order to allow the shipping company to be informed about the shipping details and to allow the client to possibly track delivery, there is the need to add new interactions for exchanging the necessary information.

***C1:* Roles Identification.** *(i) Identify the set of roles of the composite protocol; (ii) define the correspondences with the roles of the component protocols.*

The step is similar to what described for protocol specification. The characteristic feature is the additional need of defining proper mappings between the roles that are already defined in the component protocols and those identified for the composite protocol. This is done by the analyst who applies a proper renaming. If needed, the analyst can introduce new roles that are not included in the component protocols or even remove some roles. In the case of our example, "client" and "addressee" might be substituted by "customer". Thus, on the whole, the protocol wiil include three roles: *merchant* (shortened as "merc" in the following), *customer* ("cust"), and *shipping company* ("sc").

C2: Contractual Relationships Identification. *(i) Identify the set of contractual relationships and conditions of interest to be represented in the composite protocol; (ii) define the content of the initial state.*

Also for this step it may be necessary to apply a renaming of facts, in order to univocally identify facts which have different names in different component protocols and to distinguish facts which have the same name with different meanings (e.g. *withdraw* may represent the *withdrawal* of money from a bank account in one protocol, and the abort of a contract in another protocol). The analyst is also required to individuate, if any, further contractual relationships or conditions of interest to be taken into account. For instance, one additional contractual relationship would be that the shipping company should commit to deliver the goods to a customer who paid the merchant for them. For what concerns the initial state, it can be obtained as the union of the initial states of the component protocols, after the described renaming is applied.

C3: Identify Actions Social Meanings. *(i) Consider the set of actions given in the component protocols; (ii) identify additional actions' definitions or changes to be applied.*

The set of actions of the composite protocol is based on the union of the sets of actions of the component protocols, which, however, could be affected by some modifications and by proper renaming. In order to identify which definition need to be modified and how, for each pair of action and commitment, coming from different component protocols, the analyst should determine whether the former has some impacts, which is not captured by its current definition, on the latter. If this is the case, the analyst provides a new definition

for the action, so as to fill the gap. Once the resulting set is obtained, the analyst should individuate additional actions to introduce and for them she has to provide a suitable social meaning.

C4: Identify Constraints. *Identify the set of constraints by considering: (i) constraints of the component protocols; (ii) constraints capturing additional requirements.*

First, the analyst considers the union of the sets of constraints defined in the component protocols and associates to each of them a unique identifier. Afterwards, she decides if some constraint should be substituted with a new, stronger or weaker one, that better captures a desired requirement. Due to the fact that constraints can be used to represent temporal and non-temporal relationships among conditions, constraints help to solve the fundamental problem of specifying how the different component protocols intertwine with one another. This can be done by relating a condition (e.g. a commitment to do something), which is addressed by a protocol, with a condition addressed by another protocol, by exploiting one of the constraint operators foreseen by 2CL. In the example, it could be necessary to express that: (a) before the merchant sends the receipt, the shipping company should have delivered the items; (b) once the shipping company has received the order from the merchant, the client is no more allowed to withdraw from the order; (c) the merchant should take the commitment of sending the item in a certain region, only after having obtained the commitment from the shipping company to perform the delivery. These conditions are captured by the following 2CL constraints:

(a) $deliver \rightarrow\!\!\bullet\ send\_receipt$
(b) $commission \bullet\!\!\not\rightarrow withdraw$
(c) $\mathsf{C}(sc, merc, deliver) \rightarrow\!\!\bullet\ \mathsf{C}(merc, cust, deliver)$

C5: composition and testing. *Combine the elements obtained from the previous steps into a new protocol specification.*

Basically, the analyst should merge the specifications coming from the component protocols, after having applied the renamings and replacements, which were identified along previous steps of the methodology. Finally, she adds the additional definitions she has introduced.

4.3 Protocol Specialization

Specialization (see Figure 5) addresses the problem of *adapting* an existing specification to different requirements. This is, actually, the case when *grafting a regulation* onto a protocol. As an instance, the telephone company in Example 1 needs to revise its interaction with the client in order to account for the new law on personal data management. Practically, the introduction of new requirements entails the definitions of new activities that are to be interleaved with the previously existing ones. The telephone company should introduce an action for *informing* a client on the way her personal data will be used, and an action by which the client *accepts* it. There is also the need of specifying that both actions must be performed before the contract is signed.

Behind the term "adaptation" there is the desire to change a protocol specification so that it meets the requirements of a different context. In this process those characteristics which define the identity of the original protocol [41] are to be preserved, specifically: the design objective the protocol was defined to achieve, and the set of interacting roles it encompasses. This is the main aspect of specialization that makes it differ from composition. This latter,

indeed, is an operation that produces a new protocol with a different set of roles and a different purpose, i.e. with its own new identity w.r.t. those of its components. Considering Example 1, the new protocol adopted by the telephone company still aims at ruling the sale procedures and it still involves the original interacting parties.

Following these considerations, the definition of specialization that we propose takes inspiration from the notion of specialization adopted in the *object oriented paradigm*. Specifically, specialization allows both to *extend* a protocol by introducing new definitions (e.g. for roles, activities, and constraints) and to *override* some of the already existing ones. Everything that is not redefined is inherited *as-is* from the original protocol. Overriding an action definition allows, for instance, the specification of additional meanings. By overriding constraints, instead, it is possible to weaken/strengthen some requirements. Going back to Example 1, the action of signing a contract should account for the effect of accepting the terms for personal data management, which originally were not foreseen. This result can be achieved by overriding the existing definition. Moreover, the new business protocol may include the additional role of "data manager" (the person in charge of storing the clients' personal data).

Given a 2CL protocol (*base protocol* in the following), the main aspects to consider for tackling specialization are: which elements are to be added? How do they graft on the original specification? Which elements need to be modified and which are the effects of these changes?

S1: **Roles Identification.** *(i) Identify the set of participants to be accounted for in the specialized protocol; (ii) determine which of these are not represented in the base protocol.*

The analyst determines which of the identified roles are new and, thus, to be added and whether there are new actions to include. In the example, the interacting parties that the analyst individuates are the telephone company (*tc*), the client (*c*) and the *data manager* (*dm*). Let us suppose that the base protocol already accounts for the roles of "telephone company" and "client": during the interaction the client can ask for some service supplied by the company, eventually providing her personal data for allowing the company preparing the contract; finally she can sign the contract; the company allows the client to withdraw from the contract, by means of the action *del_client* which means that the contract is aborted. The role "data manager" is to be added, together with its actions. For instance, among the actions that can be performed by the data manager there are *store_data* and *delete_data*, used respectively to inform the telephone company that the data have been stored in and deleted from the database. For what concerns the other two roles, the telephone company is expected to *inform* the client on how her data will be managed, and to *notify* her that data have been cancelled; the client can decide to *accept* the conditions of the contract.

S2: **Contractual Relationships Identification.** *(i) Identify new contractual relationships and conditions of interest to be taken into account; (ii) identify changes on the content of the initial state.*

Taking into account new requirements usually yields the introduction of new contractual relationships among the parties. To do so, the analyst first considers the new roles (identified in the previous step) and determines whether they are involved in contractual relationships with the other roles. Afterwards, she considers the remaining roles and determines whether the new requirements call for them being engaged in new commitments. Additional conditions of interest may also be identified. In the example, the introduction of the new law on data management requires that the telephone company commits to provide the necessary documentation on how personal data are managed and to notify the client when her personal

data are removed from the database. This engagements can be represented respectively by the commitments $\mathsf{C}(tc, c, document\_supplied)$ and $\mathsf{C}(tc, c, delete\_data, notify\_deleted\_data)$. Afterwards, the analyst reconsiders the initial state, and refines its content by accommodating the new requirements. The obtained initial state overrides the one of the base protocol. In the example, it is reasonable to expect that the telephone company is engaged to supply the documentation about how personal data will be used from the very start. Hence, the commitment $\mathsf{C}(tc, c, document\_supplied)$ will belong to the initial state of the business protocol.

S3: **Identify Actions Social Meanings.** *(i) Identify the social meaning of the activities to be added; (ii) identify changes to the social meaning of the base protocol actions.*

For each activity to be introduced in the specification, the analyst is expected to provide a suitable definition, capturing the desired social meaning. It could also be the case that the specification of some of the actions, defined in the base protocol, needs to be modified. There are many reasons why action updates may be necessary. For instance, an effect that is relevant in a certain context may be irrelevant in another context. In order to change an action, it is sufficient to provide a new definition that *overrides* the old one.

The meaning of action *inform* is that the necessary documentation has been given to the client (*document_supplied*). By performing the action of *acceptance* the client agrees with the privacy policy and takes the commitment of sending her personal data to be stored, that however is created only in case the company has provided the documentation. Otherwise, it would not be clear which terms are being accepted:

(a) *inform* **means** document_supplied.

(b) *acceptance* **means** CREATE(C(*c*, *tc*, *provide_personal_data*)) **if** document_supplied.

The action *del_client* supplied by the base protocol is as follows:

(c) *del_client* **means** contract_aborted.

With the introduction of the regulation, the telephone company must also commit to notify the client when her personal data will be actually deleted by the data manager. Thus, the base protocol definition for the action is overridden by the following one:

(c) *del_client* **means** contract_aborted, CREATE(C(*tc*,*c*,*delete_data*,*notify_deleted_data*))

The other actions of the base protocol (e.g. *provide_personal_data*), are inherited as-is. Therefore, there is no need to explicitly account for them in this step of the methodology.

S4: **Identify Constraints.** *(i) Identify new constraints to be introduced; (ii) identify changes to the set of constraints defined in the base protocol specification.*

Specializations often need to specify new requirements on how the interaction should be carried on and where the added activities find place with respect to those of the base protocol. Considering the example, it is clearly stated that *before* the company can supply its services (this is part of the base protocol), it must obtain from the client, the commitment to provide her personal data (this is added by the specialization). This requirement can be captured by expressing the constraint $\mathsf{C}(c, tc, provide\_personal\_data) \multimap supply\_service$. In some cases it could be necessary to modify the constraints which belong to the base protocol, e.g. by relaxing them. This is, once again, possible by overriding their definitions.

S5: **Specialization and Testing.** The last step of the methodology consists in the definition of the new specification, which is obtained by applying to the base protocol all the changes individuated in the previous steps of the methodology. Everything that is not affected by changes is inherited as is from the base specification. More precisely, starting from the base protocol, the analyst: (i) introduces the new roles identified by the specialization; (ii) intro-

duces the new contractual relationships and, if necessary, substitutes the initial state specification; (iii) substitutes the definitions of the actions that have been redefined, and adds the definitions of the new actions; (iv) introduces the new constraints and substitutes those that are redefined. The result of these operations is a new 2CL protocol obtained as specialization of another one.

Once the resulting specification is obtained, it is necessary to analyse it for determining whether it realises exactly the desired model of the interaction. For instance: are the action definitions correct? Does the set of constraints capture exactly the set of interactions that are intended to be legal? If constraints are not properly defined, indeed, it could be the case that the allowed interactions include some false positive (interactions that are represented as legal but that actually should be not) or exclude some of those desired to be legal. In case the analyst discovers some misbehaviour in the specification, the methodology can be iterated until the desired result is achieved. As a support to this task, Section 7 describes a set of tools that allow different kinds of operations and analyses. Among them, it is possible to draw the graph of 2CL constraints, thus providing a perception of the flow imposed by the specification (even though constraints do not capture a rigid flow of execution). Moreover, it is possible to generate and visualise the graph of possible interactions that are allowed by the actions definitions, graphically annotating those that lead to violations. A practical use of this graph is presented in the analysis of the case study.

5 Case Study: The Markets in Financial Instruments Directive

Let us now introduce the real-world case study, that we used to test the framework: the Markets in Financial Instruments Directive number 2004/39/CE [33], issued by the European Commission. The key characteristic is the need of integrating business protocols with new regulations. In particular, since one of the main concerns of the directive is the protection of investors, it introduces new regulations that financial service agencies must follow. In this section we show how the financial product selling protocol can be represented as a 2CL business protocol and how the introduction of MiFID can be addressed as a *Specialization* of it. Both tasks will be addressed by applying the 2CL Methodology. Thanks to such a methodology, a business analyst is guided through the process of individuating the elements to be introduced (e.g. new activities, constraints) and how they find place in the context of a sale protocol. As a first step, let us apply the 2CL Methodology for protocol specification, described in Section 4.1, in order to define a basic sale. Afterwards, we will apply Specialization to graft the Directive on the individuated protocol.

5.1 Pre-MiFID sale business protocol.

The sale protocol captures the interaction that should occur among a potential investor, a financial promoter and a bank, for the purchase of a financial product. The role of the financial promoter is to present some products to the client. When the client accepts to buy a product, the stipulation of the contract is made with the bank. When the contract is signed, the financial promoter should be notified.

Step M1: Roles Identification. The roles are deduced from the actors who take part to the interaction. The sale protocol includes three parties: an investor (*inv*), a financial promoter (*fp*), and a bank (*bank*). The actions the various roles can execute are:

propose_solution: this action allows a financial promoter to present an investor a financial product;
accept_proposal: this action is to be read as the beginning of the agreement between the investor and the financial promoter. By accepting a proposal, the investor states the interest in signing a contract;
introduce_investor: this action represents the formal introduction of the client to the bank. It is executed by the financial promoter;
issue_contract: by means of this action the bank; communicates that the sale contract is ready;
sign_contract: by means of this action the investor signs the contract;
countersign_contract: this action is performed by the bank. By signing the contract, the bank accepts the engagement of taking care of the investment of the client, until its end. The bank is also expected to notify the financial promoter, so that the work that the latter carried on can be paid;
notify: This action is used to notify to the financial promoter that the bank has stipulated the contract with the investor.
invest: this action represents the end of the contract. By means of it the bank states that the investment was made effective, thus extinguishing its engagement with the investor.
withdraw: by means of this action the investor can quit the contract. It acquires the desired meaning if the contract was countersigned and if the investment had not been made effective yet. In this case the client can retract the engagements he is involved in, and he renounces to the investment.

Step M2: Contractual Relationships Identification. In the context of the sale of financial services, when the initiator accepts a proposal from the financial promoter, she is actually taking the engagement of signing the contract when this will be ready. This engagement is captured by the commitment $\mathsf{C}(inv, bank, issue\_contract, sign\_contract)$ and it binds the investor unless the latter decides to rescind from the contract. Another contractual relationship involves the bank and the financial promoter: when the bank countersigns a contract, that was signed by the investor, the bank should notify the financial promoter. We express it as a commitment as follows: $\mathsf{C}(bank, fp, sign\_contract, notify)$. Moreover, the bank is bound to make the investment effective. This engagement is created, once again, when the bank countersigns the contract: $\mathsf{C}(bank, inv, sign\_contract, invest)$. The relationships that this set of commitments generates among the interacting parties is graphically represented in Figure 6. For what concerns the initial state, none of the individuated commitments or facts is assumed to hold since the beginning of the interaction, therefore, the initial state is assumed to be empty.

Step M3: Identify Actions Social Meanings. The actions of the sale protocol are reported hereafter together with a description of their intended meaning, which is provided based on how commitments can be manipulated (e.g. created, released, cancelled) by the agents.

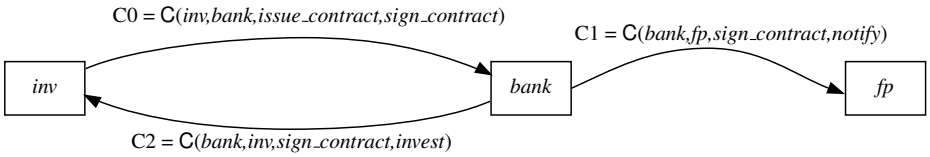

Fig. 6 Relationship diagram of the roles involved in the sale protocol.

Physical Event	**Executor**	**Social Meaning**	**Context**
(a) propose_solution	fp	*none*	true
(b) accept_proposal	inv	CREATE(C(*inv*, *bank*, *issue_contract*, *sign_contract*))	propose_solution
(c) introduce_investor	fp	*none*	true
(d) issue_contract	bank	*none*	introduce_investor ¬ withdraw
(e) sign_contract	inv	*none*	issue_contract ¬ withdraw
(f) countersign_contract	bank	CREATE(C(*bank*, *fp*, *sign_contract*, *notify*)) CREATE(C(*bank*, *inv*, *sign_contract*, *invest*))	¬ withdraw issue_contract
(g) notify	bank	*none*	sign_contract countersign_contract
(h) invest	bank	*none*	sign_contract ¬ withdraw
(i) withdraw	inv	RELEASE(C(*bank*, *inv*, *sign_contract*, *invest*)) RELEASE(C(*bank*, *inv*, *invest*)) CANCEL(C(*inv*, *bank*, *sign_contract*))	countersign_contract ¬ invest ¬ withdraw

Table 2 Action definitions for the financial services sale protocol.

Table 2 reports the social meaning of each protocol action, the conditions under which the social meaning holds, and the role entitled for the execution of each action.

- A proposal can be made at any time, by executing the action *propose_solution*, which has only an implicit social meaning (*(a)* in Table 2).
- Let us consider the commitment $\mathsf{C}(inv, bank, issue\_contract, sign\_contract)$ and determine which of the identified actions impact on it. This commitment is created when the investor accepts a proposal for a certain financial product. This effect, however, is achieved only in case a proposal has already been made (i.e. when there is something to accept), otherwise it is not clear what the investor is accepting. Therefore, we define the action *(b) accept_proposal* as:

$$accept\_proposal \textbf{ means } \text{CREATE}(\mathsf{C}(inv, bank, issue\_contract, sign\_contract))$$
$$\textbf{if } propose\_solution$$

- A contract can be issued (action *(d), issue_contract*) by the bank only if the investor was introduced by the financial promoter (*(c) introduce_investor*), otherwise the bank does not have the necessary information concerning the investor. At this point the client can sign the contract (action *(e), sign_contract*).
- The commitments $\mathsf{C}(bank, fp, sign\_contract, notify)$ and $\mathsf{C}(bank, inv, sign\_contract, invest)$ are taken by the bank by countersigning the contract (action *(f), countersign_contract*). These effects can be achieved only when the contract is ready and they make sense

only if the investor has not already changed idea by withdrawing from the contract. These conditions can be expressed as reported in the last column of Table 2.
- When the investor and the bank have stipulated a contract (the former by signing it and the latter by countersigning the contract) a notify (action *(g), notify*) can be sent to the financial promoter.
- The investment (action *(h), invest*) is an action performed by the bank. It represents the achievement of the investment agreed by the parties by signing the contract. Therefore, in order to be effective, it must be the case that the investor signed the contract and he/she did not execute a withdraw.
- A withdraw (action *(i), withdraw*) can be performed by the investor if the contract was signed by the bank and if the investment has not been executed yet. The meaning of this action is the release of the engagements taken from the bank to reach the investment and the cancellation of the commitment from the investor to sign the contract (in case this has not been done yet).

Step M4: Identify Constraints. By means of constraints the analyst can specify a set of requirements that interactions should respect. Concerning the sale protocol, they can be summarised as follows:

(c1) After the investor commits to sign a contract, the financial promoter is required to introduce the investor to the bank, so that the interaction can continue towards the definition of the investment.
(c2) If the investor took a commitment to sign the contract, given that the bank prepares it, then the bank is expected to actually prepare the contract.
(c3) If the contract prepared by the bank is signed by the client, then it is required that the former becomes, sooner or later, bound to make the investment happen.
(c4) Before the investment is achieved the bank should at least commit to notify it to the financial promoter.

The above requirements can be represented in terms of 2CL constraints, let us see how.

- Requirement *(c1)* captures both a temporal and a conditional condition: if something is achieved, then something else must hold and only after. Therefore, the right 2CL operator for capturing this constraint is the *cause*, resulting to the following definition:

$$(c1)\ \mathsf{C}(inv, bank, issue\_contract, sign\_contract) \bullet\!\!\rightarrowtail introduce\_investor$$

- Also the second requirement captures a temporal condition, but its nature is slightly different: the requirement states that in case the investor takes the commitment to sign the contract, after an unspecified amount of time then the contract must be ready. Notice that having a contract prepared does not impose that the preparation would occur in the future. Actually, the bank might even have it prepared before the investor takes a decision. This kind of requirement is expressed by the *response* constraint:

$$(c2)\ \mathsf{C}(inv, bank, issue\_contract, sign\_contract) \bullet\!\!\rightarrow issue\_contract$$

Notice that satisfying this constraint corresponds to requiring that the conditional commitment of the investor will become active sooner or later. Notice that it is not correct, in this case, to make the investor directly commit to sign the contract by using $\mathsf{C}(inv, bank, sign\_contract)$. The reason is that a commitment of this kind would not be safe for the investor: he or she does not have any guarantee that the bank will, actually, prepare the contract to be signed. In case the bank does not, this unconditional commitment would make the investor liable of a violation because he or she will not be able to

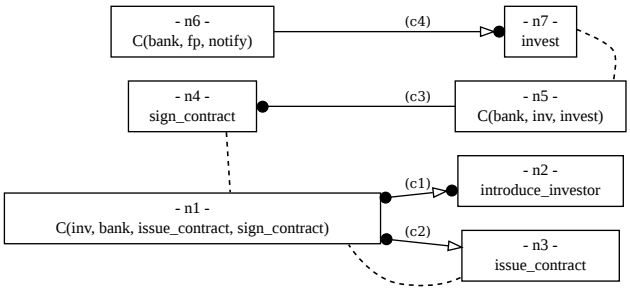

Fig. 7 Graphical representation of the 2CL constraints from the sale protocol. Dashed lines highlight relationships between events/facts and commitments.

satisfy the commitment. Instead, by using constraint *(c2)*, the bank would be responsible for the violation.

– *(c3)* does not express any temporal requirement on the acquisition of the different conditions. Therefore, *correlate* is a suitable operator for representing the constraint:

$$(c3)\ sign\_contract \bullet\!\!- \mathsf{C}(bank, inv, invest)$$

– Finally, the last requirement expresses that if a certain condition is achieved, then another one has to be achieved before. This kind of temporal requirement is expressed by the *before* operator. The resulting constraint is:

$$(c4)\ \mathsf{C}(bank, fp, notify) \rightarrowtail invest$$

The individuated constraints can be graphically represented as in Figure 7. This graph provides an intuition of the flow that constraints create. Dashed lines are added to highlight the relationships that tie facts/events and those commitments in which they are involved as part of the antecedent or consequent conditions. A precise view of the allowed interactions is, instead, given by the labelled graph of the possible interactions, see Figure 8, which is described below.

Analysis of the interaction. Figure 8 reports the labelled graph of the possible interactions that can be enacted when applying the business protocol which was built so far. The graph is obtained by means of the tool that we will describe in Section 7, by exploiting a particular feature that allows to generate the graph representing all the legal interactions and the possible causes of violation. When the occurrence of a physical event violates a constraint, the further interactions that can be generated from that state are not explored. The use of this tool and the analysis of the graphs it generates support the analyst in discovering errors in the specification, and provide a graphical representation of the boundaries within which agents do not incur in risks of violation. The graphical convention we adopted in the graph is reported in Table 3. In words: (i) a state of violation is represented as a (red) diamond, with an incoming dashed (red) arrow. The label of the arrow specifies which constraint is violated; (ii) a state in which there is a pending condition (i.e. a condition that, due to some constraint, is required to be achieved) is (yellow-)coloured. Again, the label of the incoming arc specifies which is the pending constraint; (iii) a state with a single outline, independently from the shape, is a state that contains unsatisfied commitments; (iv) a state with a double outline, independently from the shape, does not contain active commitments. Some states may combine different representations. For instance a (yellow-)coloured diamond with single outline is a state where there are unsatisfied active commitments, where a constraint is violated and where there is a pending condition.

	Commitment		No Commitments	
	Pending Condition	No Pending Condition	Pending Condition	No Pending Condition
Violation				
No Violation				

Table 3 Legend of states representation in the labelled graph.

Figure 8 helps drawing some considerations. When the interaction stops at state number 3, no violation occurs. In words, the business protocol allows the investor to refuse a proposal by the financial promoter, just by not accepting it. Instead, when the interaction stops at states 5,6 or 8, a violation is raised. In these cases, indeed, there are pending conditions to be achieved, imposed by a constraint. In the first two cases, the bank still has to issue the contract, as required by constraint *(c2)* (see the label on the arc); in the second case, the bank still has to take the commitment to make the agreed investment effective (since the investor

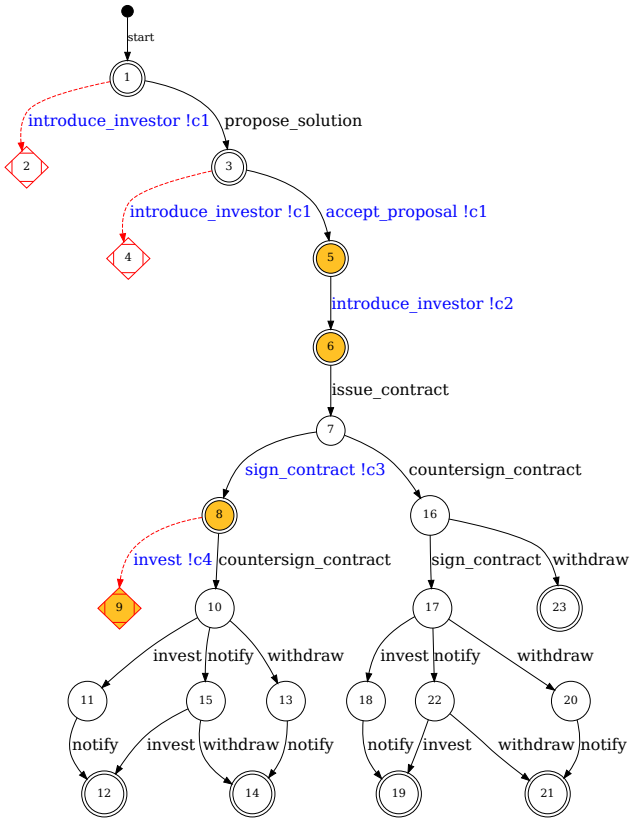

Fig. 8 Partial graph of the interactions of the pre-MiFID sale protocol. It reports all the legal interactions and the possible causes of violation.

has signed the contract), which is required by constraint *(c3)*. States 2, 4 and 9 represent the violation of a constraint. In particular, states 2 and 4 represent a violation of constraint *(c1)* that is due to the fact that an investor is introduced before he or she took the commitment to sign the contract (i.e. by accepting a proposal). State 9 represents the violation of constraint *(c4)* that requires the bank to take the commitment to notify the investment to the financial promoter.

5.2 Grafting of MiFID.

Let us now describe how the regulation introduced by the MiFID can be grafted on the sale protocol described in the previous section. Specifically, as a case study we consider the regulation that applies to the offer of investment services off-site. This is the case when a bank promotes and sells of financial products with the help of external collaborators (financial promoters or intermediaries). According to MiFID the proposal of products and the definition of a contract must foresee the following activities:

- **Identification**: the client must be identified by an identity card or equivalent document;
- **Qualification**: the intermediary (financial promoter) supplies all the foreseen documentation about his/her professional qualification and the rules that he/she must stick to;
- **Profiling**: the intermediary must profile the client, gathering information about the balance sheet, knowledge about financial subjects, investment aims. This phase requires the filling of the MiFID form, which explicitly specifies the resulting category of client and which is to be signed also by the client;
- **Selection**: the proposed financial products must agree with the client's profile. This requires that financial products are classified w.r.t. the different profiles of risk;
- **Evaluation**: the proposal is evaluated through a simulation: if it is adequate an order is filled and signed both by the client and by the intermediary, otherwise the product is discarded;
- **Verification**: the documentation is sent to the investment trust, which must check that there are no errors or missing data. In this case, the documentation is corrected and sent back to the intermediary, otherwise the contract is sent to the client;
- **Withdrawal**: the client can decide to cancel an order.

The requirements prescribed by MiFID graft onto the previously existing financial product sale protocol. What happens if an intermediary buys a financial product for a client, violating some of the constraints imposed by the MiFID? The *sale is valid*, the client results to be the owner of the product. This happens because MiFID does not define *sale* (sale is defined by a different regulation) but dictates how the interaction with the client should be carried on by adding a new layer of regulations on top of existing ones. So, the violation of some constraint does not affect the sale directly but creates both a *risk of sanction* and a *risk of exposure* for the intermediary. This is witnessed by a sentence by the Italian Supreme Court (*Cassazione civile a sezioni unite*, num. 26724 and 26725 [28]) which decided that in case of violations, like the above, if the client was economically damaged he/she can ask for a compensation and, in the most serious cases, for the cancellation of the contract between the client and the intermediary. This will be transparent to the seller, who will not be involved in the quarrel and will have no consequences (specifically he/she will not have to give money back).

Step S1: Roles Identification. The actors foreseen by the Directive are a financial promoter, a bank and an investor. Therefore, MiFID does not add new roles to those identified in the

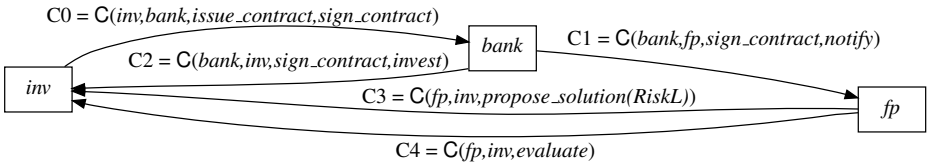

Fig. 9 Relationship diagram of the roles involved in the sale protocol enriched with the relationships introduced by the MiFID.

sale protocol. The actions contained in the original sale protocol are not modified by MiFID with the exception of *propose_solution*, which must now account for the risk level of the proposed product, which must match the risk level of the investor. MiFID, however, requires to add to the business protocol the new actions, that are described in words at the beginning of this section. Specifically, *interview* aims at identifying the investor and at supplying the necessary documentation; *profile* classifies the investor into a proper risk category and creates the commitment, of the financial promoter, to evaluate a suitable solution; *evaluate* allows discharging this commitment: the financial promoter performs some simulation in order to find a solution that fits the investor requirements and communicate these results. By means of this action, the financial promoter commits to propose a solution that is characterized by the desired level of risk. Classification of financial products according to the different levels of risk is performed by means of the action *classify_products*. Finally, the bank must *verify* that the documentation does not contain errors and that no data are missing.

Step S2: Contractual Relationships Identification. MiFID requires the analyst to account for contractual relationships that were not foreseen in the sale protocol. Specifically, it requires the financial promoter to commit to evaluate, by means of simulation, a certain proposal in order to verify that it respects the level of risk (which can be *low*, *medium* or *high*) that was accepted by the investor and to communicate the results of these evaluation. This engagement is represented as $\mathsf{C}(fp, inv, evaluation)$; it is created when the financial promoter profiles the investor. Besides evaluation, the financial promoter is expected to propose financial products at a certain level of risk to the investor, a relationship that is captured by the commitment $\mathsf{C}(fp, inv, propose\_solution(RiskL))$. Figure 9 shows how commitments relate the roles of the business protocol. For what concerns facts, MiFID only requires that an investor is provided of all the necessary documentation (*document_supplied*). Last, MiFID does not require any contractual relationship or condition to hold since the beginning of the interaction.

Step S3: Identify Actions Social Meanings. The social meaning of the actions is reported in Table 4. The action *profile(RiskL)*, by which the financial promoter communicates to the investor his/her risk level resulting from the profiling, creates two commitments, one to perform the evaluation of the products and the other to propose a product whose risk level matches that of the investor. The protocol accounts for a new action *propose_solution(RiskL)* (overriding the one of sale) that, differently from the one of the plain sale protocol, accounts for the risk level of the proposed solution. In order for this action to effectively substitute the old one, which did not account for the level of risk, it is necessary to add, as part of its social meaning, the fact *propose_solution* which is part of the "if" condition of action (b), *accept_proposal*, in Table 2.

Physical Event	Executor	Social Meaning	Context
(j) interview	fp	document_supplied	true
(k) profile(RiskL)	fp	CREATE(C(*fp*, *inv*, *evaluate*)) CREATE(C(*fp*, *inv*, *propose_solution(RiskL))*)	true
(l) classify_products	fp	*none*	true
(m) evaluate	fp	*none*	classify_products
(n) verify	bank	*none*	¬ withdraw sign_contract
(o) propose_solution(RiskL)	fp	*propose_solution*	*true*

Table 4 Actions introduced by the MiFID and their definitions.

Step S4: Identify Constraints. MiFID defines also when the different tasks should be performed. In particular:

(c5) The investor should have received all the necessary documentation before the bank commits to perform an investment. This is because the investor must be aware of the consequences of the bank taking such an engagement.
(c6) The investor should receive the necessary documentation before committing to sign a contract. This is to be sure that the investor is aware of the terms of the contract and of the consequences of signing it.
(c7) When the investor is introduced to the bank, the financial promoter should provide also the documentation proving his/her identity. This is to avoid misbehaviours from the financial promoter, proposing investments without a real investor.
(c8) A contract must be verified before the bank countersigns it.
(c9) The investor must be profiled before the financial promoter can perform a simulation and find a solution, that is tailored to the investor's needs.
(c10) A proposal must be evaluated before offering it to the investor.

The 2CL constraints capturing the above requirements are:

(c5) $document\_supplied \rightarrow\!\bullet\, \mathsf{C}(bank, inv, invest)$
(c6) $document\_supplied \rightarrow\!\bullet\, \mathsf{C}(inv, bank, issue\_contract, sign\_contract) \vee \mathsf{C}(inv, bank, sign\_contract)$
(c7) $interview \rightarrow\!\bullet\, introduce\_investor$
(c8) $verify \rightarrow\!\bullet\, countersign\_contract$
(c9) $profile(RiskL) \rightarrow\!\bullet\, evaluate$
(c10) $evaluate \rightarrow\!\bullet\, propose\_solution(RiskL)$

All constraints specify relationships between the achievement of conditions of the sale protocol and of MiFID. Figure 10 shows the graphical representation of the constraints for the sales protocol specialized with the MiFID.

Step S5: Specialization and Testing. The application of the results of the previous steps to the sale protocol specializes it so that it accounts for MiFID. Figure 11 reports the legal interactions between the investor, the financial promoter and the bank and the possible causes of

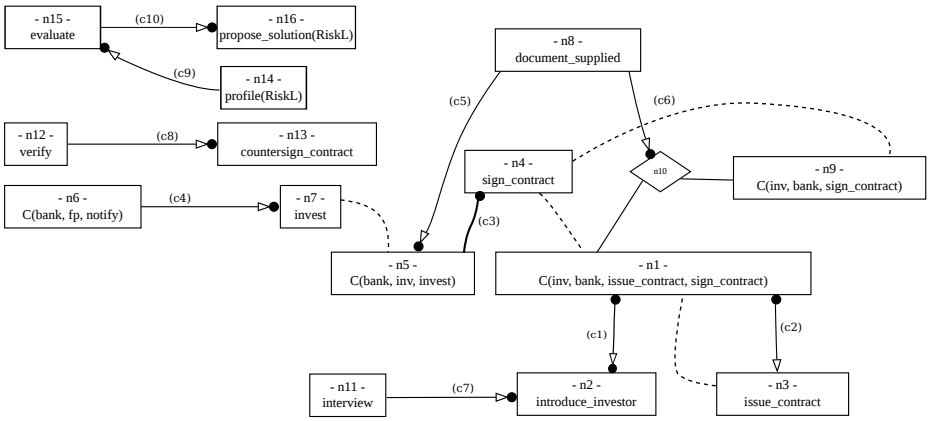

Fig. 10 Graphical representation of the constraints from the sale protocol after the introduction of MiFID. Dashed lines highlight relationships between events/facts and commitments.

violation[1]. Nodes and arcs are colored in ways that show whether the corresponding actions and states fulfill the requirements of the sale protocol, enriched by the MiFID Directive.

Arcs that are highlighted by means of an ellipse represent the activities that were originally foreseen by the sale protocol. MiFID grafts upon them. Thus, for instance, starting the interaction by proposing a solution is illegal, after the introduction of MiFID (state 2). This is not surprising, since the directive requires the execution of additional steps with the aim of providing specific guarantees on the sale protocol. The investor cannot accept a proposal until he receives the required documentation (state 11), and the financial promoter cannot perform the evaluation if he has not profiled the investor yet (state 41). Finally, notice that the investor cannot be introduced (to the bank) before the interview.

This kind of graph is a valuable tool for the analyst, who can validate the business protocol and discover possible design mistakes. In order to avoid them, the analyst could decide the introduction of enforcement or regimentation strategies. For instance, let us consider constraint (c7). It requires that before the investor can be introduced to the bank, the financial promoter must have performed the interview. Suppose that the sale is performed with the support of a software, used both by the bank and by the financial promoter. A simple way for regimenting this constraint is to require the introduction of the identity card (or equivalent) number of the investor, in order for the sale to proceed. Of course, when evaluating the adoption of similar solutions, the analyst is expected to consider the costs of modifying the adopted application software.

6 2CL Methodology evaluated by users

Different aspects need to be taken into account when evaluating the effectiveness of a methodology. Besides testing the expressiveness of the language and the applicability of the methodology with the help of the MiFID case study, we designed a questionnaire for collecting feedback on our approach. We collected feedback's from twenty-five persons, all of them working in the area of computer science (and most of them having a degree in computer science): eight senior researchers, nine Post-Docs and eight Ph.D students. 60% of

[1] Other graphs, as well as other examples, can be found at `http://di.unito.it/2CL`.

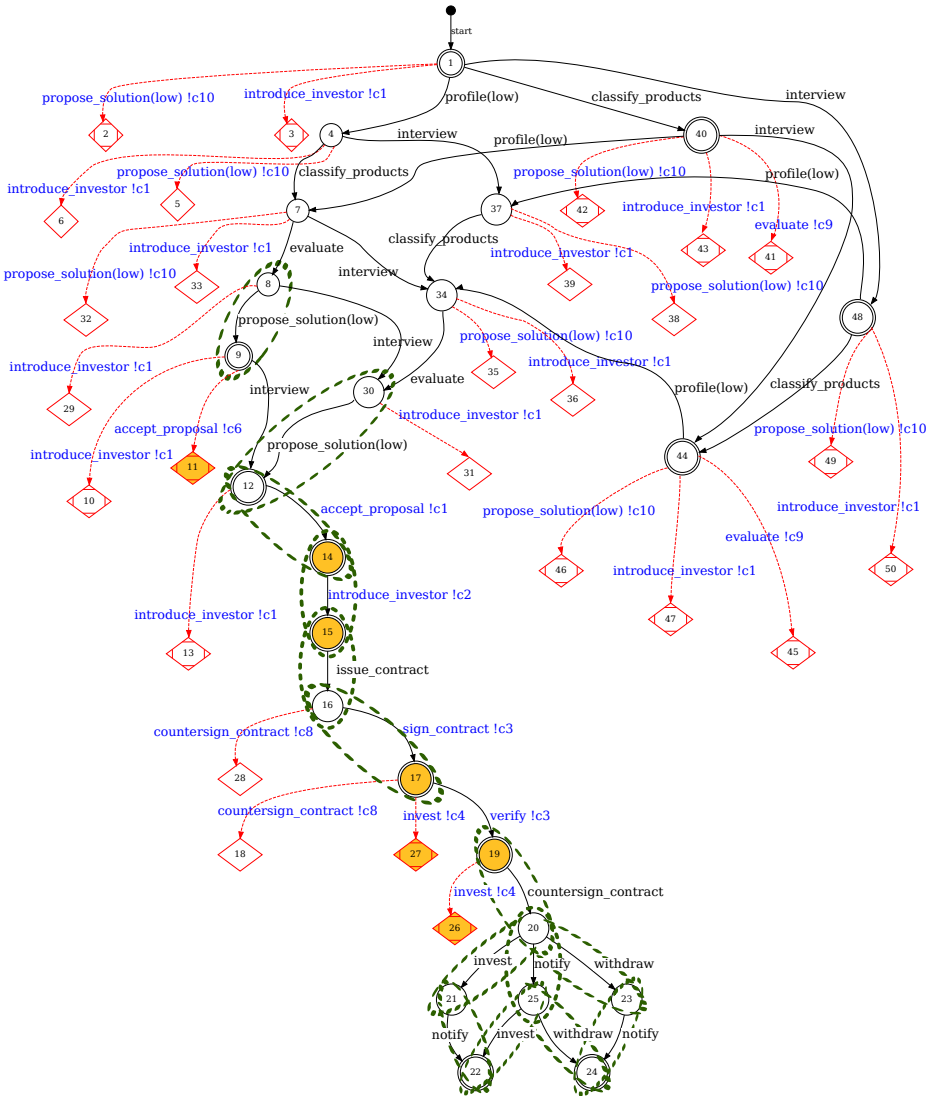

Fig. 11 Graph of possible executions of the sale protocol, specialized with MiFID. Ellipses highlight the actions of the sale protocol. The actions foreseen by MiFID are interleaved with them.

volunteers have knowledge on MAS, 28% of participants have knowledge on commitment protocols. First, we introduced to the volunteers the basic notions about MAS, commitments, and 2CL protocols; then, we presented the use of 2CL Methodology by modelling step by step the case study of OECD Guidelines [37,32], on the protection of privacy and transborder flow of personal data. Finally, we showed the labelled graph of the case study, produced by the tool described in Section 7.

At the end of the presentation we asked participants to fill in a questionnaire, that was designed by taking inspiration from [14,15]. It is structured in three parts. The first two aim at gathering information about the volunteer, i.e. his/her position and background on

	Clarity												*Adeq.*	*Easy of use*			*Unamb.*
	Intuitive Notation	Symbols well defined	Methodology well explained	Effective as Guide	Easy of identification in an informal specification				Support of the methodology				Sufficient elements	Symbols easy to understand	Easy to draw	Spec. easy to understand	Role of symbols well def.
					Ro	CC	Act	Cst	Ro	CC	Act	Cst					
Mean	4.00	4.00	4.12	3.91	4.30	3.52	3.52	3.57	4.13	3.57	3.48	3.43	4.35	4.32	4.10	4.00	4.30
Mode	4	4	4	4	5	4	3	4	4	4	4	3	4	5	4	4	5
Variance	0.50	0.60	0.44	0.17	0.77	0.90	1.08	0.98	0.75	0.89	0.72	0.71	0.33	0.98	0.80	0.30	0.77
Standard Deviation	0.70	0.80	0.67	0.42	0.88	0.95	1.00	0.99	0.87	0.95	0.85	0.84	0.57	0.99	0.90	0.50	0.88
Relative Standard Deviation	0.17	0.20	0.16	0.11	0.20	0.27	0.29	0.28	0.21	0.27	0.24	0.25	0.13	0.23	0.20	0.10	0.20

Table 5 Results of the questionnaires about 2CL Methodology.

MAS; the third part aims at evaluating the methodology and consists of questions related to *(i)* clarity and understandability, *(ii)* adequacy and expressiveness, *(iii)* ease of use, and *(iv)* unambiguity. Participants had the possibility to write also *(v)* general comments, which did not fit the questions or expressed suggestions for future developments. Questions were answered by expressing scores in a 5-point Likert-like scale (5 is the best score). Some questions included the space for writing additional comments or explanation.

Table 5 reports the results that were computed on the set of collected filled questionnaires[2]. Columns correspond to questions. For each question, we reported the following values: *(i)* the *mean* score, which provides the average score value; *(ii)* the *mode*, showing the most frequent score value; *(iii)* the standard deviation; *(iv)* the relative standard deviation. The last two values are reported for understanding whether the mean score can be considered a significant value, i.e. whether it can be perceived as a representative evaluation for a certain aspect of the methodology. Actually, the mean values that we obtained for the various questions range from 3.43 through 4.35. All these values can be considered as representative since the relative standard deviation is, in all cases, (much) lower than 0.5.

Clarity and Understandability. The first two questions concern 2CL: whether the relevant concepts have simple and intuitive notations and if symbols and syntax are well explained. The subsequent two questions refer to the methodology, the way it is explained and how effective it is in guiding a protocol designer. In general, the *mean* for these questions is really close to 4 and, indeed, most of the evaluations are 4 or 5 (as shown also by the relative standard deviation reported in the table). This suggests that the symbols of the language and their syntax are clear, as well as the steps of the methodology. The last two questions aim at gathering impressions on how easy is to identify and represent roles, commitments, actions and constraints, starting from an informal specification, and applying the steps of

[2]The collected questionnaires are available at `http://di.unito.it/questionnaire`.

methodology. Roles are perceived as the easiest part to be designed (the mode is 5). For what concerns the identification of 2CL constraints, the mean value we obtained is 3.43. the fact that this result is slightly inferior to the others is, actually, not surprising because this is the part of the specification which requires the highest degree of competence on 2CL, and all of the volunteers had no knowledge on the language before participating to this evaluation.

Adequacy and Expressiveness. The aim of the second part of the questionnaire is to understand which important aspects are perceived as missing in 2CL. All the participants but one (who was neutral) commented the language as highly expressive. Indeed, the mean is 4.35 and the relative standard deviation is very low. We collected also some suggestion on which additional elements it would be interesting to capture; among the others, further logical constraints and an explicit representation of time.

Easy of Use. This part of the questionnaire evaluated whether the approach is perceived as easy to understand, considering three different aspects: *(i)* whether symbols and their use are easy to understand; *(ii)* whether it is easy to sketch a 2CL specification by hand, and *(iii)*, given a 2CL specification, whether it would be difficult to "read" it. For all these questions we got very high values: the mode is 5 in the first two cases and 4 in the last one.

Unambiguity. The last part of questionnaire regards clarity of each element of the notation (roles, commitments, actions, and constraints). The feedback express a quite positive evaluation of this aspect, but some perplexity arose about the properties of and the differences between constraints and commitments. These comments were very useful to improve and extend the description of these components inside 2CL Methodology.

7 2CL Tools

This section briefly sketches the tool, that we developed for performing the analysis of business protocols, and which is based on the formalization proposed in [3]. Such a formalization concerns the creation of compact and annotated graphs, which provide a global view of the possible interactions, showing which are legal and which cause constraint (or commitment) violations. The aim is to enable the verification of exposure to risk on the graph of the possible executions, and to support taking decisions about how to behave (or how to modify the protocol) in order to avoid such a risk. The system that we realized is an Eclipse plug-in (java-based and open source IDE). The source code is available at the URL `http://di.unito.it/2CL`. The functionalities that the system supports can be grouped into three components: *design*, *reasoning* and *visualization*.

Design Component. The design component provides the tools that are necessary for defining a business protocol. It supplies two editors, respectively for actions and constraints definition. The former is basically a text editor, where actions can be specified following the syntax reported in Section 3. The definition of protocol constraints, instead, is graphically given by means of the editor depicted in Figure 12. Constraints are represented by drawing facts as labelled boxes and connecting them with 2CL arrows (following the graphical representation of Table 1) or with logical connectives so as to design more complex formulas. The graphical representation has a corresponding textual representation (either XML or plain text). Starting from the plain text, a Prolog program is generated by means of a recursive descending parser, written by means of Antlr, where 2CL actions are mapped into Prolog predicates as in [53].

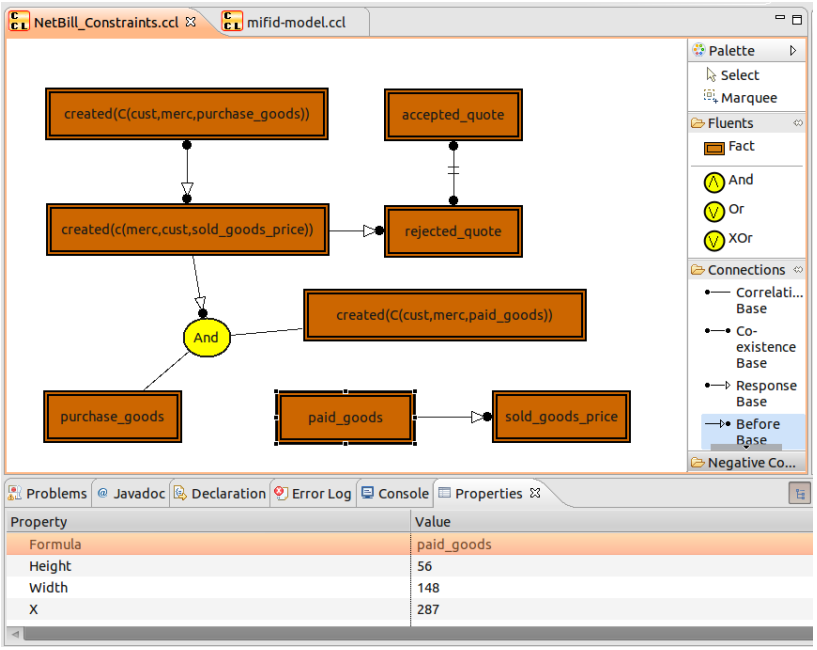

Fig. 12 Overview of the editor for constraints specification. The palette on the right allows the user to select the elements to introduce in the graph (facts, logical connectives and 2CL arrows).

Reasoning Component. The reasoning component consists of a program that is able to generate all the possible interactions, which are allowed by the protocol actions, and to label them as legal or not, according to the protocol constraints. The program is realized in *tuProlog*[3] and it builds upon the *enhanced commitment machine* source code by Winikoff *et al.* [53]. In particular, we adopt the same implementation of commitments life-cycle and the same mechanism to determine state transitions (i.e. the application of the protocol actions). Such a framework, as pointed out in [45], lacks a compilation procedure for turning a protocol specification into a finite state machine and does not support non-terminating protocols. Despite these limitations, the framework offers an easy way to visualize the unfolding of a protocol execution. With respect to Winikoff *et al.*'s implementation, we introduce as a difference the verification of 2CL constraints, performed as verification of properties on single states [3]. The output of the Prolog reasoner is an annotated graph of all the possible interactions (an annotated reachability graph). Interactions are obtained based on protocol action definitions only. Annotations, instead, account for all the regulative aspects, concerning both commitments and constraints. So each state of the graph has a set of labels that capture, if any, the violation of some constraints, the presence of pending conditions, and the presence (or absence) of unsatisfied active commitments. Pending conditions are due to constraints that are only partially verifiable in the state at issue: for instance, in the case of coexistence when only one of the two conditions which should occur in an execution already occurred. A pending condition becomes a violation when the interaction stops before the constraint condition is fully satisfied. On the whole, the graph will include both legal states and states in

[3] http://www.alice.unibo.it/xwiki/bin/view/Tuprolog/

which violations occur. We realised also a variant of the program which expands only states where no violations have been detected. In this way the resulting graph is more compact and, therefore, easier to understand.

The specification of the protocol, on which the Prolog program relies, is given as a text file and it is obtained as output of the design editors. It is transformed into a Prolog file by a parser written in Java. The parser generates also different kinds of graphs showing how constraints tie the protocol actions.

Visualization Component. All the graphs produced by the reasoning component (for instance, those used in this article) can be visualized as images. The labelled graph, in particular, is represented by adopting some graphical conventions that we reported at the end of Section 5. Particularly interesting, for the analysis of the labelled graph, are the functionalities supplied by the *Graph Explorer* tool. It is realized in Java and it relies on iDot (Incremental Dot Viewer[4]), an open source project that uses the *prefuse*[5] visualization framework for Dot graph display. The Graph Explorer supplies different functionalities, like the visualization of the shortest path given a source and a target state, and the visualisation of legal (or illegal) paths only. The user can add or delete a node in a path; search a state starting from its label; and search all the states that contain a certain fact or commitment. Moreover, the tool allows the exploration of the graph one state at a time, by choosing which node to expand. Another feature of the Graph Explorer is the possibility to build the graph incrementally, as well as to expand only violation-free nodes or all nodes.

The tool we described is a useful support in the analysis of business protocols. As explained, the labelled graph represents all the possible interactions where each state is labelled according to the evaluation of the protocol constraints. Highlighting the possible violations amounts to alerting the user about a risk. Exploring the labelled graph by means of the Graph Explorer can emphasize whether performing a certain sequence of actions results in a violation and, in this case, if there is a way to return on a legal path. For what concerns the designer, it is not always easy, when specifying a protocol, to individuate which constraints to introduce but, with the help of the tool, it becomes easy to identify undesired behaviours and revise the constraints so as to avoid them. Moreover, the tool and the labelled graph support the analysis of a business protocol, by helping the identification of situations where it may be necessary to perform some regimentation or enforcement.

8 Conclusion

This paper proposes the 2CL Methodology for designing business interactions ruled by protocols that are expressed in 2CL [6]. Being based on the notions of commitment and responsibility, the proposal supports the analysis of risks of violation when new requirements are to be taken into account and also, as a special but relevant case, when new regulations graft onto previously existing business protocols. The approach relies on a *commitment-based* representation of business protocols. Over the past decade, commitments have emerged as a leading basis for declaratively specifying interaction protocols. In these approaches the social state is used by agents as a basis for decision making, as they interact while pursuing their individual goals, and provide abstractions that allow to capture the contractual relationships among the partners instead of prescribing a strict ordering of the messages

[4] `http://code.google.com/p/idot/`

[5] `http://prefuse.org/`

[47]. This is crucial when modeling cross-business interactions in real-life scenarios [19, 17, 5]. The resulting support for flexible enactment, and autonomy is one of the strong points of commitment protocols. The declarative nature and the modularity of 2CL facilitate the specialization and the composition of business protocols, and so to adapt them to different contexts or to different regulations.

2CL Methodology builds upon Amoeba [17], a methodology specifically designed for commitment-protocols. As a difference with Amoeba, 2CL Methodology handles 2CL protocols, which include the possibility of specifying *constraints* among commitments, ruling the evolution of the social state. As another difference with Amoeba, 2CL Methodology supplies a kind of composition that, starting from 2CL protocols, produces a 2CL protocol and supplies also guidelines for realizing the specialization of protocols. Instead, 2CL Methodology shares with Amoeba the advantages of using commitments. In particular, commitments allow giving to the encoded interactions a precise business meaning, which is lacking in most existing approaches. See [17] for a deeper discussion and comparison with other agent-oriented software engineering methodologies.

The methodology was, first, evaluated by applying it to a real-world case study, MiFID, which was one of the benchmarks for the project ICT4LAW (see also [16]). The industrial partners of the ICT4LAW project recognized MiFID as a challenging testbed for approaches aimed at supporting the design of business processes which comply to regulations changing over time. Other partners in the project, such as D'Aprile et al. [16], used the same testbed for proving the efficacy of a commitment-based framework in the task of automatically verifying the compliance of business processes to laws. The results that we report in this paper prove the effectiveness of the proposed approach in modeling the "grafting" of the MiFID regulation into a business protocol. The subsequent evaluation made by interviewing a group of volunteers confirmed the expressiveness of 2CL and the effectiveness of 2CL Methodology.

Recently, Comma, a novel methodology for commitment-based business modeling was proposed [48]. Comma shares with Amoeba (and with 2CL Methodology) the same underlying notion of commitment, however, while Amoeba and 2CL Methodology are methodologies for designing commitment-based business *protocols* and, therefore, at their heart they guide the analyst through the steps for defining the social meaning of the protocol actions (and messages), Comma specifies business processes as a set of commitment-based patterns of contractual relationships, whose operational counterpart is given in terms of message sequence charts, and guides the analyst in finding the activities that are necessary to accomplish the desired evolution of the process. Patterns are intended as reusable and composable schemas that the business analysts can use to design the contractual relationships for a desired business scenario [49]. Along this line, an interesting direction for future research can be to study how 2CL constraints could be used to enrich Comma business processes with further regulative aspects. Further about patterns, Telang and Singh [47] identified a set of common patterns of interaction, represented them in terms of commitments, and proposed an approach that uses the patterns for building business protocols. Broadly, the idea is that the analyst takes from a catalogue of patterns those which fit his or her specification needs, and uses them as the building blocks of the new protocol. Along this line, Chopra and Singh [13] proposed a set of commitment patterns which capture common business patterns, reporting, for each of them, the robustness requirements it meets. Robustness requirements are used by the protocol designer in the phases of selection and of composition which lead to the specification of a new business protocol.

Declarative approaches are recognized to suit well the representation of business protocols [50, 18]. However, various authors [45, 34] claim that in many cases they are not so

much intuitive. To overcome this limitation, we decided to realize a set of (graphical) tools for supporting the analysis of business protocols. Specifically, the tools include editors for specifying 2CL commitment protocols and interfaces for visualizing graphically the constraints and take advantage of the perception of a flow given by the partial temporal ordering [4]. For what, instead, concerns the point of view of the involved agents, the tools allow the visualization of the possible interactions, even interactively one piece at a time, thus supporting the identification of the violations which could possibly occur. This aspect is particularly important when new requirements arise and each party has to understand their impacts on the interaction, e.g. if any of the previously performed behaviors is not allowed anymore. The implementation of the generation of the annotated graph of the possible interactions is proved to be correct with respect to the operational semantics of the generalized commitment machine presented in [3].

Let us remark that the 2CL methodology currently supports the design of protocols not of agents. On the whole, our intuition is that agents not necessarily will be designed ad hoc for the system after the protocol is designed but, hopefully, it will be possible to re-use already existing agents equipped with the necessary capabilities. When an agent accepts to play a role in a commitment-based protocol, it accepts the social meanings of the actions of the protocol. As a consequence, when it creates a commitment, the social expectation is that the commitment will be satisfied, but the plan that will be enacted to do so is totally up to the agent and to its capabilities. To the best of our knowledge, there are not many proposals that tackle the problem of realizing agents, given a commitment-based protocol. The proposal in [2] is aimed at making a step further in this direction.

Other related approaches focus on formal methods for the automatic verification and reasoning about commitment protocols. Along this line, Mallya and Singh [31] describe a semantic approach to interaction protocols, by defining an algebra for commitment protocols. Such an algebra provides a conceptual basis for reasoning about protocols in terms of traditional software engineering notions such as refinement or aggregation, and includes the operators *merge* and *choice* and a subsumption relation for protocols. The former operator requires the interleaving of the runs of the merged protocols, the latter basically provides the union of the runs of two protocols. Based on the algebra, the authors show how to compare protocols, and how to reason about them in terms of aggregation and refinement. The formal framework enables the automatic verification of properties of the designed protocols but it does not cope with the issue of embedding the abstractions, that are supported by the algebra, into a methodology for protocol design nor the challenge to develop software tools for supporting designers in the task of tailoring existing protocols so that they meet given requirements.

Yolum's proposal [55] aims at checking at design time whether a commitment protocol is correct with respect to a predefined set of generic properties, which are consistency, effectiveness and robustness. A protocol is consistent when it does not bring to inconsistent states. It is robust when it offers more than one way to achieve the desired results. It is effective when it can be enacted and ended successfully. The desired properties are expressed in event calculus and algorithms are supplied for verifying whether a protocol satisfies them. The framework that we proposed does not support the verification of similar properties yet but it would be interesting to develop a tool for their verification.

In [27] Gerard and Singh formulate a notion of *refinement* of one protocol by another. Then, they develop a tool called *Proton*, which exploits CTL and the well-known MCMAS model checker in order to verify protocol refinements. Proton refinement is different than 2CL Methodology specialization. A sub-protocol is a refinement of a super-protocol when all of its executions traces are also execution traces of the super-protocol, and their roles

must be the same. The specialization of a protocol, instead, allows the introduction of new roles and does not foresee the same limitation on execution traces. The protocol is enriched with all the roles, actions, constraints specified in a regulation. In Proton, moreover, the legal executions of actions are constrained by means of guards. Guards are similar, in their intent, to 2CL constraints but the languages used to specify them are different. In the case of Proton, they are conditions on the current state of the executions, while in the case of 2CL they are (temporal) conditions that concern the path that brought to a state.

El-Menshawy et al. [22,23] and Bentahar et al. [8] focus on the problem of verifying the conformance of commitment-based protocols to given desirable properties, by proposing several methods, that are based on model checking techniques. In particular, in [22] El-Menshawy et al. define a formal semantics for social commitments and their operations (i.e. *withdraw*, *fulfill*, *violate*, *release*, *assign*), based on the $ACTL^{*c}$ logic, i.e. CTL^{*} branching-time temporal logic extended with modalities for social commitments and operations. Within this logical framework, a specification for commitment-based protocols is proposed, and model checking techniques are used in order to automatically verify protocols against temporal properties, such as *fairness*, *safety*, *liveness* and *reachability*. Generic properties presented by Yolum [55] can be related to such temporal properties, enabling the use of the proposed model checking techniques to verify correctness of a commitment-based protocol at design time.

Recent works [26,25,9] concern run-time reasoning on commitment-based specifications of interaction, and the related crucial issue of *monitoring*, i.e. checking run-time whether the commitments, that are created during an on-going interaction, are fulfilled (or violated). As highlighted in [9], "commitments lend themselves well to external inspection and monitoring, but in concrete terms, effective monitoring tools are still missing", and monitoring of commitment-based specifications still represents a challenging open issue for the multi-agent community. As a difference with the 2CL model, where the focus is on (temporal) constraints relating commitments, the specifications proposed by Fornara and Colombetti [26] and by Chesani et al. [9] allow the specification of temporal properties, such as starting points and deadlines, *inside* the commitments. Fornara and Colombetti 's [26] model includes concepts like commitment, role and norm, which are useful for specifying artificial institutions. In this model, obligations and prohibitions are intended as positive and negative commitments, which are created by agent communicative acts or by the activation of norms related to an agent's role. Their approach relies on the use of semantic web technologies. In particular, since OWL 2 DL and SWRL are used to specify obligations and prohibitions, standard state-of-the-art reasoners can be exploited to implement run-time monitoring.

Finally, the framework in [9] is, instead, based on event calculus ($\mathscr{EC}$). As El-Menshawy [22], it supplies a formal semantics for commitments and their operations. Technically, commitment-based models are formalized in a logic programming setting, by first-order $\mathscr{EC}$ axioms. Such specifications have an operational counterpart. In particular, the authors propose to use reactive event calculus ($\mathscr{REC}$) to implement and execute commitment-based specifications of the interaction, and to perform run-time monitoring of the commitments.

Acknowledgements This research was partially funded by "Regione Piemonte" through the project ICT4-LAW. The authors would like to thank the anonymous reviewers for the valuable comments, which helped improving the quality of this paper.

References

1. M. Baldoni, C. Baroglio, I. Brunkhorst, N. Henze, E. Marengo, and V. . Patti. Constraint Modeling for Curriculum Planning and Validation. *International Journal of Interactive Learning Environments*, 19(1):83–123, 2011.
2. M. Baldoni, C. Baroglio, and F. Capuzzimati. 2COMM: a commitment-based MAS architecture. In M. Cossentino, A. El Fallah Seghrouchni, and M. Winikoff, editors, *Proc. of the 1st International Workshop on Engineering Multi-Agent Systems, EMAS 2013, held in conjuction with AAMAS 2013*, pages 17–32, St. Paul, Minnesota, USA, May 2013.
3. M. Baldoni, C. Baroglio, F. Capuzzimati, E. Marengo, and V. Patti. A Generalized Commitment Machine for 2CL protocols and Its Implementation. In *Post-Proc. of the 10th Int. Workshop on Declarative Agent Languages and Technologies X, DALT 2012, Revised Selected and Invited Papers*, number 7784 in LNAI, pages 96–115. Springer, 2013.
4. M. Baldoni, C. Baroglio, and E. Marengo. Behavior-Oriented Commitment-based Protocols. In *Proc. of ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 137–142. IOS Press, 2010.
5. M. Baldoni, C. Baroglio, E. Marengo, and V. Patti. Grafting Regulations into Business Protocols: Supporting the Analysis of Risks of Violation. In *Forth International Workshop on Requirements Engineering and Law (RELAW 2011), held in conjunction with the 19th IEEE International Requirements Engineering Conference*, pages 50–59, Trento, Italy, August 30th 2011. IEEE Xplore.
6. M. Baldoni, C. Baroglio, E. Marengo, and V. Patti. Constitutive and Regulative Specifications of Commitment Protocols: a Decoupled Approach. *ACM Transactions on Intelligent Systems and Technology, Special Issue on Agent Communication*, 4(2), 2013.
7. M. Baldoni, C. Baroglio, V. Patti, and E. Marengo. Supporting the Analysis of Risks of Violation in Business Protocols: the MiFID Case Study. In *Information Systems: Crossroads for Organization, Management, Accounting and Engineering*, pages 545–553. Springer, 2012.
8. J. Bentahar, M. El-Menshawy, H. Qu, and R. Dssouli. Communicative commitments: Model checking and complexity analysis. *Knowl.-Based Syst.*, 35:21–34, 2012.
9. F. Chesani, P. Mello, M. Montali, and P. Torroni. Representing and monitoring social commitments using the event calculus. *Autonomous Agents and Multi-Agent Systems*, 27(1):85–130, 2013.
10. A. Chopra. *Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing*. PhD thesis, North Carolina State University, Raleigh, NC, 2009.
11. A. K. Chopra, A. Artikis, J. Bentahar, M. Colombetti, F. Dignum, N. Fornara, A. J. I. Jones, M. P. Singh, and P. Yolum. Research directions in agent communication. *ACM Transactions on Intelligent Systems and Technology*, 4(2), 2013.
12. A. K. Chopra and M. P. Singh. Constitutive interoperability. In L. Padgham, D. C. Parkes, J. P. Müller, and S. Parsons, editors, *AAMAS (2)*, pages 797–804. IFAAMAS, 2008.
13. A. K. Chopra and M. P. Singh. Specifying and applying commitment-based business patterns. In *Proc. of AAMAS*. IFAAMAS, 2011.
14. K. H. Dam. *Evaluating and Comparing Agent-Oriented Software Engineering Methodologies*. PhD thesis, Applied Science in Information Technology, School of Computer Science and Information Technology, RMIT University, Australia, 2003.
15. K. H. Dam and M. Winikoff. Comparing Agent-Oriented Methodologies. In *AOIS*, volume 3030 of *Lecture Notes in Computer Science*, pages 78–93. Springer, 2003.
16. D. D'Aprile, L. Giordano, A. Martelli, G. Pozzato, D. Rognone, and D. Theseider Duprè. Business Process Compliance Verification: An Annotation Based Approach with Commitments. In *Information Systems: Crossroads for Organization, Management, Accounting and Engineering*, pages 563–570. Springer, 2012.
17. N. Desai, A. K. Chopra, and M. P. Singh. Amoeba: A Methodology for Modelling and Evolving Cross-Organizational Business Processes. *ACM Transactions on Software Engineering and Methodology*, 19(2), 2009.
18. N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Interaction Protocols as Design Abstractions for Business Processes. *IEEE Trans. Software Eng.*, 31(12):1015–1027, 2005.
19. N. V. Desai, A. K. Chopra, M. Arrott, B. Specht, and M. P. Singh. Engineering Foreign Exchange Processes via Commitment Protocols. In *IEEE Int. Conf. SCC 2007*, pages 514–521, 2007.
20. B. Dunin-Keplicz and R. Verbrugge. Evolution of collective commitment during teamwork. *Fundamenta Informaticae*, 56(4):329–371, 2003.
21. M. El Menshawy, J. Bentahar, W. El Kholy, and R. Dssouli. Reducing model checking commitments for agent communication to model checking ARCTL and GCTL*. *Autonomous Agents and Multi-Agent Systems*, August 2012.

22. M. El-Menshawy, J. Bentahar, W. El Kholy, and R. Dssouli. Verifying conformance of multi-agent commitment-based protocols. *Expert Syst. Appl.*, 40(1):122–138, 2013.
23. M. El-Menshawy, J. Bentahar, H. Qu, and R. Dssouli. On the verification of social commitments and time. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011*, pages 483–490. IFAAMAS, 2011.
24. N. Fornara and M. Colombetti. A Commitment-Based Approach To Agent Communication. *Applied Artificial Intelligence*, 18(9-10):853–866, 2004.
25. N. Fornara and M. Colombetti. Representation and monitoring of commitments and norms using OWL. *AI Commun.*, 23(4):341–356, 2010.
26. N. Fornara and M. Colombetti. Specifying and enforcing norms in artificial institutions: A retrospective review. In *Declarative Agent Languages and Technologies IX - 9th International Workshop, DALT 2011, Revised Selected and Invited Papers*, volume 7169 of *Lecture Notes in Computer Science*, pages 117–119. Springer, 2012.
27. S. N. Gerard and M. P. Singh. Formalizing and verifying protocol refinements. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(2), 2013.
28. G. Gibilaro. Cassazione Civile Sentenza, Sez. SS.UU., 19/12/2007, n. 26724 e 26725. Intermediazione finanziaria, nullità del contratto e risarcimento del danno, 2007.
29. A. J. I. Jones and M. Sergot. *On the characterization of law and computer systems: the normative systems perspective*, pages 275–307. John Wiley & Sons, Inc., 1994.
30. P. J. Kammer, G. A. Bolcer, R. N. Taylor, A. S. Hitomi, and M. Bergman. Techniques for Supporting Dynamic and Adaptive Workflow. *Computer Supported Cooperative Work*, 9(3/4):269–292, 2000.
31. A. Mallya and M. P. Singh. An algebra for commitment protocols. *Autonomous Agents and Multi-Agent Systems*, 14(2):143–163, 2007.
32. E. Marengo. *2CL Protocols: Interaction Patterns Specification in Commitment Protocols*. PhD thesis, Università degli Studi di Torino, Research Doctorate in Science and High Technology, Specialization in Computer Science, 2012.
33. Directive 2004/39/EC of the European Parliament and of the Council of 21 April 2004 on markets in financial instruments.
34. T. Miller and J. McGinnis. Amongst first-class protocols. In *Proc. of Eng. Societies in the Agents World VIII*, volume 4995 of *LNCS*, pages 208–223. Springer, 2008.
35. M. Montali. *Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach*, volume 56 of *LNBIP*. Springer, 2010.
36. M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative Specification and Verification of Service Choreographiess. *ACM Transactions on the Web (TWEB)*, 4(1), 2010.
37. Organisation for Economic Co-operation and Development. OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. Available on-line, 1980. http://www.oecd.org/.
38. M. Pesic. *Constraint-Based Workflow Management Systems: Shifting Control to Users*. PhD thesis, Eindhoven University of Technology, 2008.
39. M. Pesic, H. Schonenberg, and W. van der Aalst. DECLARE: Full Support for Loosely-Structured Processes. In *Proc. of the 11th IEEE International Enterprise Distributed Object Computing Conference*. IEEE Computer Society, Washington, DC, USA, 2007.
40. M. Pesic and W. M. P. van der Aalst. A Declarative Approach for Flexible Business Processes Management. In *Business Process Management Workshops (BPM 2006)*, volume 4103 of *LNCS*, pages 169–180. Springer, 2006.
41. G. Regev, I. Bider, and A. Wegmann. Defining Business Process Flexibility with the help of Invariants. *Software Process: Improvement and Practice*, 12(1):65–79, 2007.
42. M. Reichert and P. Dadam. ADEPT$_{\text{flex}}$-Supporting Dynamic Changes of Workflows Without Losing Control. *J. Intell. Inf. Syst.*, 10(2):93–129, 1998.
43. J. Searle. *The construction of social reality*. Free Press, New York, 1995.
44. M. P. Singh. An Ontology for Commitments in Multiagent Systems. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
45. M. P. Singh. Formalizing Communication Protocols for Multiagent Systems. In *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1519–1524, Hyderabad, India, January 2007. AAAI Press.
46. M. P. Singh, A. K. Chopra, and N. V. Desai. Commitment-Based Service-Oriented Architecture. *IEEE Computer*, 42(11):72–79, Nov. 2009.
47. P. R. Telang and M. P. Singh. Abstracting Business Modeling Patterns from RosettaNet. In *Service-Oriented Computing: Agents, Semantics, and Engineering*, 2010.
48. P. R. Telang and M. P. Singh. Comma: a commitment-based business modeling methodology and its empirical evaluation. In *Proc. of Int. Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2012*, pages 1073–1080. IFAAMAS, 2012.

49. P. R. Telang and M. P. Singh. Specifying and Verifying Cross-Organizational Business Models: An Agent-Oriented Approach. *IEEE Transactions on Services Computing*, 5(3):305 –318, 2012.
50. W. van der Aalst, M. Pesic, and H. Schonenberg. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - Research and Development*, 23:99–113, 2009.
51. W. M. P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, 2002.
52. W. M. P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, 2005.
53. M. Winikoff, W. Liu, and J. Harland. Enhancing Commitment Machines. In *In Proc. of DALT 2004*, volume 3476 of *LNCS*, pages 198–220. Springer, 2004.
54. M. J. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.
55. P. Yolum. Design time analysis of multiagent protocols. *Data and Knowledge Engineering*, 63(1):137–154, 2007.
56. P. Yolum and M. P. Singh. Commitment Machines. In *Intelligent Agents VIII, Proc. of ATAL*, volume 2333 of *LNCS*, pages 235–247. Springer, 2001.
57. P. Yolum and M. P. Singh. Designing and Executing Protocols using the Event Calculus. In *Proc. of the 5th Int. Conf. on Autonomous Agents*, pages 27–28, 2001.
58. P. Yolum and M. P. Singh. Flexible protocol specification and execution: applying event calculus planning using commitments. In *Proceedings of AAMAS*, 2002.