CME: A Tool for Designing Business Models based on Commitment Patterns

Stefano Lanza, Simone Vallana, Cristina Baroglio

Università degli Studi di Torino, Dipartimento di Informatica
stefano.lanza@studenti.unito.it, simone.vallana@studenti.unito.it,
cristina.baroglio@unito.it

Abstract. Commitment-based patterns are an emerging technology, originally developed in the area of Multi-Agent Systems, for representing business models. In order to be used effectively, a similar approach calls for the development of graphical tools, supporting the designer in the task of knowledge representation. This paper presents CME, an Eclipse plugin developed at this very aim. CME supports the application of the Comma methodology for the design of business models, based on commitment-based patterns. It also allows the creation of new commitment-based patterns and pattern libraries.[1]

1 Introduction and Motivation

Patterns, as introduced by Telang in [6], are frequently used schemas of engagement which specify the roles that are involved in an interaction and the commitments in which they are involved. They can be used as building blocks in the construction of complex interactions models. By relying on commitments [4], patterns allow capturing, in a natural way, both the business intents, and the responsibilities each role has. At the same time, they support a declarative representation which does not prescribe specific courses of action but lets agents free to decide which actions to take as long as they respect their commitments. This approach, therefore, respects the autonomy of agents, advised in the Multi-Agent systems research area.

Commitments (and therefore patterns) reside at a *social level*: agents share a common interpretation of actions which are relevant to the interaction (e.g. message utterances). Each action "counts as" a social meaning. So, for instance, in a commercial transaction there is a mutual agreement that in case a client takes an object he or she has a commitment to pay for it; on the other hand, if the client gives money for an object the merchant has a commitment to give (or to send) the object at issue to the client. This is an example of a pattern. Notice that at this level there is no specification of who should act first, this depends on the circumstances and on the agents themselves. No procedure is encoded. This level of representation allows agents to adapt more easily to contingent conditions,

[1] This work is the third year degree thesis of Stefano Lanza and Simone Vallana, whose advisor is Cristina Baroglio.

to different environments, and to take advantage of occasions because it leaves them free to decide their actions.

Another advantage is that responsibilities are clearly stated inside commitments. Each commitment, in fact, has a "debtor" agent, which is the agent who is responsible for the achievement of the commitment consequent condition. This responsibility is accepted by the agents and, for this reason, in case a commitment is violated, it is possible to take action against the debtor agent. As such, this approach is very different than more traditional approaches for specifying interaction, e.g. UML interaction diagrams and UML collaboration diagrams, which basically rely on capturing the flow of messages.

Commitment-based patterns supply the building blocks by which it is possible to compose (complex) business models. To implement this vision, it is necessary to define guidelines that the designer can follow in order to build a business model. In [5], Telang and Singh propose *Comma*, a methodology for the design of business models, based on commitment-based patterns. The methodology supports the analysis of cross-organizational scenarios, the identification of the involved roles, of their tasks, and of their interaction, based on predefined libraries of patterns of business relationships. An example of cross-organizational scenario is the invitation to tender, where bidders are called to make offers that are then evaluated by the organization which issued the call. In this context, for instance, when issuing the call, the organization, which is acting as an initiator of the interaction, commits to evaluate the proposals by the bidders, who, in turn, commit to execute what requested in the call if their proposal is accepted. Another example is outsourcing, where an organization outsources part of its functions towards some clients to another organization.

The representation of patterns is inherently graphical but to the best of our knowledge there is a *lack of tools*, that are specifically thought for supporting the designer in the construction of patterns as well as in their use. This work tries to fill the gap by proposing the *Commitment Model Editor* (CME for short), an Eclipse plug-in, which allows trained designers to build commitment-based pattern libraries as well as to build pattern-based business models. Patterns and business models are represented internally in a homogeneous way, allowing business processes to be easily turned into new patterns if needed. The paper is organized as follows. Section 2 reports background knowledge on commitments and patterns. Section 3 explains the main characteristics of CME with the help of a running example. Final remarks end the paper.

2 Commitments and Commitment-based Patterns

Commitments [4] are represented with the notation $C(x, y, r, p)$, capturing that the agent x (*debtor*) commits to agent y (*creditor*) to bring about the consequent condition p when the antecedent condition r holds. When r equals *true*, the short notation $C(x, y, p)$ is used, and the commitment is said to be *active*. Commitments have a *regulative* nature: agents are expected to behave so as to achieve the consequent conditions of the active commitments of which they are the debtors. In that case, commitments are *satisfied.* The business partners share a *social state* that contains commitments. These can be manipulated by means of the standard operations *create*, *cancel*, *release*, *assign*, *delegate*, *discharge.* Cancel and delegate are performed by the debtor, release and assign by the creditor. Discharge is performed when the consequent condition is achieved.

Commitment-based patterns are recipes for capturing recurrent business scenarios [6] – Telang and Singh in their works discuss various patterns extracted from RosettaNet [1]. A pattern consists of a set of *roles* and a set of *commitments.* Graphically, roles and commitments can be represented as the nodes of a directed graph. Arcs are either directed from a role to a commitment or from a commitment to a role. In the former case, the role is the debtor of the commitment, in the latter it is the creditor. So, for instance, in Figure 2, *Tender Issuer* is the creditor of commitment $C0$, while *Subcontractor* is the debtor. Patterns can be grouped into *pattern libraries*, which are then made available to the designers.

The *Comma methodology* encompasses the following steps:

1. Given a scenario description, extract subscenarios, each matching a pattern from the Comma pattern library;
2. Identify the roles for each subscenario, based on business functions (e.g., Shipper) that remove any ambiguity;
3. For each subscenario, identify business tasks (e.g., payment) that a role executes. Typically tasks are specified as actions executed by the participants;
4. Introduce into the business model a pattern corresponding to each subscenario. Rename the pattern characters with the roles from Step 2, and introduce the tasks from Step 3 as the antecedents and consequents of the appropriate commitments.
5. For each pattern, introduce its *message sequence chart* (MSC) into the operational model. Rename the roles and messages in the MSCs to align them with those determined in Steps 2 and 3.

CME was developed to support the first four steps of Comma. A default library of patterns was developed which includes part of the RosettaNet patterns by Telang and Singh.

3 CME Implementation and Example

CME is a graphical tool for the design of commitment-based business processes. Even tough it was developed as an Eclipse plugin, it is a stand-alone application system which does not require to have Eclipse installed in order to be used. The pivotal aspect, on which the tool is centered, is the possibility of *creating* and *using commitment-based patterns.* Indeed, the designer can both create pattern libraries, that can be used by third parties through the same tool, and create business models, by using available patterns. We describe the system and its features with a the help of examples.

Pattern Libraries. CME allows creating and using *pattern libraries*. Many libraries can be used at once. Users can exploit CME wizards to configure the paths at which the desired libraries can be found in the file system. A tab on the right of the interface allows exploring the patterns inside the selected libraries. Pattern libraries are written in XML. The format respects the structure given by the grammar:

LIBRARY $\longrightarrow$ *name author description* PATTERN_LIST
PATTERN_LIST $\longrightarrow$ PATTERN PATTERN_LIST | ε
PATTERN $\longrightarrow$ *name description* R_LIST C_LIST
R_LIST $\longrightarrow$ ROLE R_LIST |ε
C_LIST $\longrightarrow$ COMMITMENT C_LIST | ε
ROLE $\longrightarrow$ id *name description*
COMMITMENT $\longrightarrow$ id *name description state debtor creditor antecedent consequent*

As mentioned, the default pattern library used by CME contains patterns from RosettaNet and developed by Telang and Singh [1, 6]. In order to test the system, we developed a little pattern library containing patterns for the most common procedures for tenders which are applied by the Italian public administration.

Patterns. Patterns are given through the sets of roles and of commitments they include. Edges are implicitly defined by the debtor/creditor specifications. Commitments have two identifiers: one is numerical, used internally by the system, the other can be assigned by the designer and is human-oriented. The graphs are drawn by exploiting available libraries for graph representation.

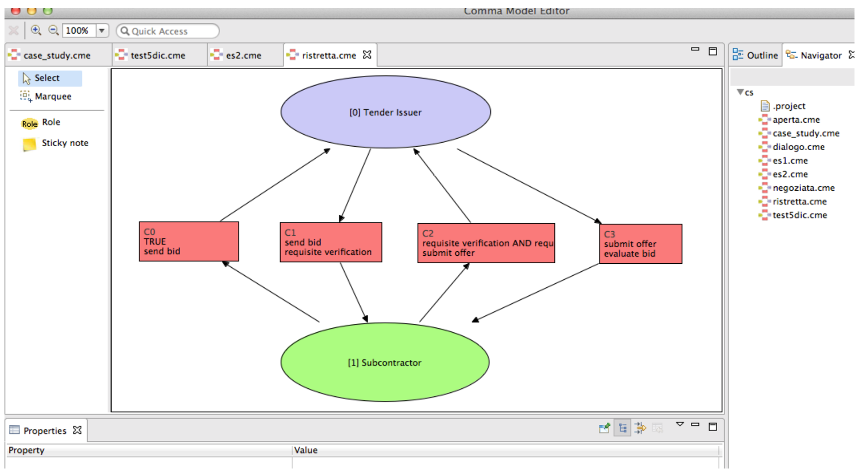

Fig. 1. Pattern for restricted procedure: ovals are roles, rectangles commitments.

As an example of pattern, Figure 1 reports the *restricted procedure* for tenders [2, 3] as it is represented inside CME. This standard procedure foresees that a *Tender Issuer* (contracting agent) calls for bids from some *Subcontractors*. Each subcontractor who answers the call is evaluated and also its proposal is evaluated. The ovals in the picture are the roles of the pattern; rectangles represent commitments. Antecedent and consequent conditions are written on separate lines. CME does not require conditions to be written by following a specific grammar. In other terms, it is not commitment-language specific.

Interface. When CME is run, an interface containing three frames is shown: The central frame contains a GEF editor that allows the graphical specification of patterns (or business models). The user can directly insert roles and sticky notes, commitments are inserted with the help of a wizard. The lower frame lists and allows the modification of all the properties of the currently selected Node object. The right frame contains views: these allow the management of pattern libraries and supply both a navigator and an outline view of the edited model. Another characteristic of the system is that it is possible to work on many business models at the same time: each one will have an own tab within the interface.

When a business model consists of many pattern instances, it is not easy to grasp the structure of the plain graph because this may include a large number of roles, of commitments, and of edges. In order to help the designer, we developed also a "zoom-out" functionality which shows the business model as a graph containing two kinds of nodes: the defined roles and the instantiated patterns. If a role is used inside a pattern, then an edge connects the corresponding role node to the pattern instance node.

Business models vs. Patterns. In general, a business model can compose several patterns and the identity of such patterns is kept in the internal representation adopted by CME. On the contrary, a pattern is a plain graph of roles and of commitments. It is, therefore, easy to convert a business model into a pattern because it is sufficient to discard part of the information stored within the business model. CME supplies a proper *export* operation which allows to convert a business model into a pattern.

3.1 Example of application of Comma with CME

We now show a little example of application of Comma, underlining how the steps are helped by the CME tool. Let us suppose, that the Municipality of Torino needs to call for bids for the management of the cleansing services of its offices, and that it will assign the task of evaluating the requisites of the bidding companies to an external Audit Authority. Let us repeat the Comma steps and see what the designer does in this case. The result is reported in Figure 2.

1. *Identify subscenarios*: there are two subscenarios:
   (a) the interaction of the municipality with bidders,
   (b) the interaction of the municipality with the Audit Authority.
   The patterns to be applied are respectively:

   (a) the pattern for restricted procedures (from the pattern library that we developed), which foresees two roles: *Tender Issuer* and *Subcontractor*;
   (b) the pattern for commercial transaction (from the default pattern library, containing patterns from RosettaNet), which foresees the roles: *Partner1* and *Partner2*.
2. *Identify roles*: there are three roles (Public Administration, Audit Authority and Cleaning Service), one of them (Public Administration) glues the two applied patterns and matches two different pattern roles (Tender Issuer and Partner1). Therefore, we rename the roles *Tender Issuer* and *Partner1* by *Public Administration*, the role *Subcontractor* by *Cleaning Service*, and the role *Partner2* by *Audit Authority*. Such roles can be created in CME directly by the user without the need of relying on specific wizards.
3. *Identify actions per role*: here the designer needs to get into the depths of how evaluation and selection are made. We will not list all actions exhaustively but, as examples, in Figure 2, the Cleaning Service should "send bids" and "submit orders"; instead, "requisite verification" is one of the Audit Authority's actions;
4. The business model will contain a single instance of each of the identified patterns, to which a proper renaming was applied. To this aim, it is possible to use the uploaded libraries and select the desired patterns. A guided procedure allows the user to perform the necessary renaming, thus connecting patterns to roles.

The resulting business model binds the agents which will play it, e.g., the Municipality of Torino as a Public Administration. It is worthwhile to comment the commercial transaction pattern: here, there seems to be a loop (if you provide the service I pay, if you pay I provide the service). However, recall that this kind of specification is not a procedure. The two peers can autonomously decide when to act. If the Audit Authority acts first, $C4$ is satisfied, while $C3$ becomes detached, and the Municipality is bound to pay. Similarly, if the Municipality pays first.

4 Possible usage and final remarks

In this paper, we introduced CME, a graphical editor for commitment-based patterns and pattern libraries. Commitment-based patterns, and therefore also CME, are particularly useful in the design of cross-organizational business models, where the interaction of the parties cannot rely on allowing direct, mutual access to their information systems, but, rather, calls for the specification of the boundaries and the expectations of the interaction at a higher and normative level.

It is, indeed, a current trend for companies to focus their internal resources on the core business delegating to third parties other functions. Similarly, Public Administrations more and more often need to interact with other authorities and organizations. When such organizations maintain relations with plenty of other actors, it is easy that some of the commitments foreseen by the contracts are

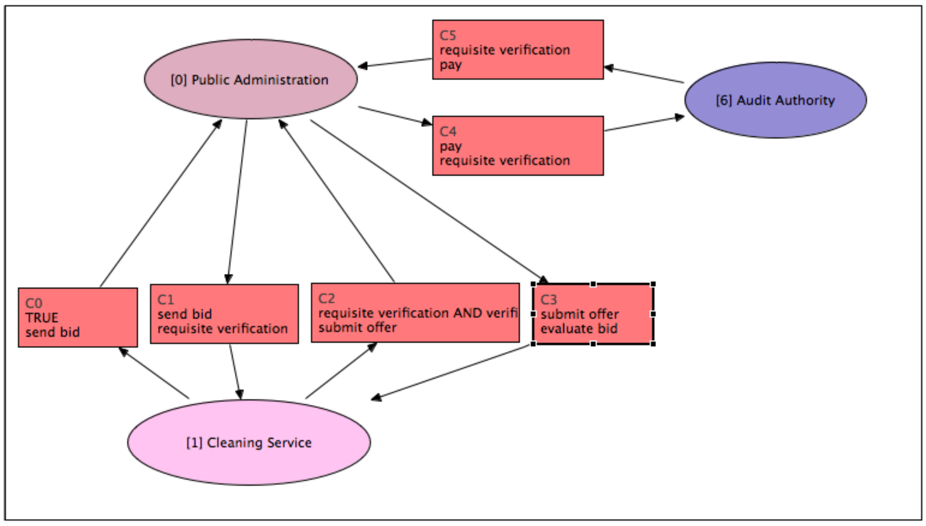

Fig. 2. Business model combining two patterns.

violated. In such cases, a software like CME can be very useful. The tool, in fact, allows specifying in a clear way the many business relationships that are built and that, if integrated with the information systems of the involved realities, would also allow the monitoring of responsibilities and of the satisfaction (or violation) of commitments. There would, thus, be a significant impact in terms of bureaucratic simplification and reduction of costs.

References

1. RosettaNet. http://www.rosettanet.org/.
2. Gazzetta ufficiale n. L 313, pp. 0003–0035, November 2009.
3. ITACA. Guida operativa per l'utilizzo del criterio di aggiudicazione dell'offerta economicamente più vantaggiosa negli appalti di lavori pubblici di sola esecuzione, 2013.
4. Munindar P. Singh. An Ontology for Commitments in Multiagent Systems. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
5. Pankaj R. Telang and Munindar P. Singh. Comma: a commitment-based business modeling methodology and its empirical evaluation. In *AAMAS*, pages 1073–1080. IFAAMAS, 2012.
6. Pankaj R. Telang and Munindar P. Singh. Specifying and Verifying Cross-Organizational Business Models: An Agent-Oriented Approach. *IEEE T. Services Computing*, 5(3):305–318, 2012.