JADE/JaCaMo+2COMM: Programming Agent Interactions

Matteo Baldoni[0000-0002-9294-0408], Cristina Baroglio[0000-0002-2070-0616], Roberto Micalizio[0000-0001-9336-0651], and Stefano Tedeschi[0000-0002-9861-390X] (✉)

Università degli Studi di Torino - Dipartimento di Informatica, Torino, Italy
`firstname.lastname@unito.it`

Abstract. We present 2COMM, a middleware in which social relationships, created during agent interactions, are represented as social commitments. These relationships are reified as resources in the agents' environment, and can be directly manipulated by the agents themselves via standard operations. We show that this perspective induces an agent programming schema that is independent of the actual agent platform. The uniformity of the approach is exemplified in two well-known agent platforms: JADE and JaCaMo.

Keywords: Social commitments · Agent Programming · Interaction.

1 Introduction

When the autonomy of components is a key requirement, software engineers can choose from a wide number of agent platforms (see, e.g., [5, 6]). However, while providing coordination and communication mechanisms, all of them fail in clearly separating the interaction logic from the agent dimension. The way in which agents interact is "hard-coded" into their implementations, with a negative impact on code decoupling and reuse. We claim that an explicit representation of the *social relationships* among agents is beneficial since it improves modularity and flexibility. In this work, we practically demonstrate these advantages when social relationships are modeled as *social commitments*. A commitment is a promise that an agent (debtor) makes to another (creditor) to bring about a condition (consequent) when a given context (antecedent) holds. Commitments have a standardized lifecycle, and can be manipulated through some standard operations. By properly selecting the terms of the agreement, an agent can create a commitment so as to entice the cooperation of another agent, while maintaining autonomy and low coupling. In particular, we present 2COMM, a framework to define *dedicated commitment artifacts*, seamlessly integrated as resources in the environment where agents are situated. These artifacts *reify* the set of social relationships created during an interaction. Such an approach paves the way for a general, platform-independent, agent programming schema: agents, by directly creating and manipulating commitments, can engage others to cooperate for achieving their goals. As a practical use case, we rely on a distributed logistics scenario.

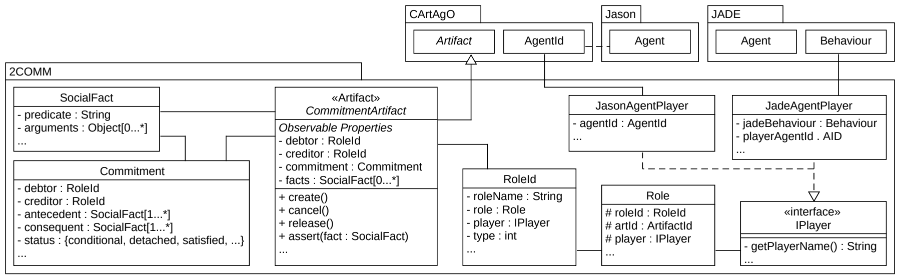

Fig. 1. Excerpt of the 2COMM architecture together with the connectors for JADE and JaCaMo.

2 Main purpose

This demo presents the main features of 2COMM, which implements the *conceptual architecture* originally presented in [4]. In 2COMM social relationships are first-class programming entities, and the demo shows the practical implications of using these entities from an agent programming perspective. Following the A&A meta-model [8], 2COMM is a middleware that makes social commitments available to agents by means of CArtAgO artifacts [9]. Notably, the approach is not bound to any agent programming platform: to use 2COMM within a specific platform, it is sufficient to realize a dedicated connector. So far, two connectors are available for JADE [5] and JaCaMo [6]. The demo exemplifies how the modeling of interaction in terms of commitments, and grounded on 2COMM, allows to program both JADE and JaCaMo agents in a uniform way.

2COMM is the result of more than four years of work of researchers and students at Department of Computer Science at the University of Torino. Main methodological and practical outcomes are presented in [1–3, 7]. The platform is freely available at `http://di.unito.it/2comm` together with some practical use cases encompassing both JADE and JaCaMo agents.

3 Demonstration

Figure 1 reports an excerpt of 2COMM main components. Each commitment artifact encapsulates a commitment and provides two *roles* that agents can enact: the debtor and creditor of the commitment itself. For instance, by adopting the debtor role, an agent will be able to perform the artifact *operations* for creating or canceling the corresponding commitment. Of course, roles are linked to agents of the specific platform through platform-dependent connector classes. Note that commitments are *observable properties* of artifacts; this means that agents focusing on commitment artifacts are notified whenever a commitment state change occurs. In particular, the infrastructure automatically handles commitment progression according to their standard lifecycle and to the events occurring in the environment. Events that are relevant for the progression of commitments are encoded as *social facts*, and hence maintained within the artifacts themselves. Agent programming with 2COMM amounts, then, to realizing a classical "sense-plan-act cycle", whose phases can be renamed "observe the environment", "activate behaviors according to the state of relevant commitments", and "schedule behavior execution."

Let us consider, to illustrate, a setting in which a *seller* agent sells its products online and ships them to a *customer* agent. Let us assume *seller* needs to rely on multiple couriers for the shipment from the original location A to destination D. The route is divided into three parts: 1) from A to B, covered by truck trk_1; 2) from B to C, covered by plane *pln*; and 3) from C to D, covered by truck trk_2. The relationships among these agents are captured by the following set of commitments. From a conceptual point of view, these commitments

$$\begin{aligned}
&\boldsymbol{c_1} : \mathsf{C}(seller,\ customer,\ pay(500, seller),\ at(goods, D))\\
&\boldsymbol{c_2} : \mathsf{C}(trk_1,\ seller,\ pay(50, trk_1) \wedge at(goods, A),\ at(goods, B))\\
&\boldsymbol{c_3} : \mathsf{C}(pln,\ seller,\ pay(200, pln) \wedge at(goods, B),\ at(goods, D))\\
&\boldsymbol{c_4} : \mathsf{C}(trk_2,\ pln,\ pay(50,\ trk_2) \wedge at(goods, C),\ at(goods, D))
\end{aligned}$$

have a precise meaning in terms of mutual expectations between the agents. For instance, c_1 represents the offer that *seller* proposes to *customer*: if *customer* pays 500 for some *goods*, *seller* will deliver them at *customer*'s place D. Since *seller* cannot directly deliver the goods, however, it will likely, and safely, create c_1 only after having established the commitments c_2 and c_3, representing the offers made by the couriers to move the goods along the route. With the former, the *seller* gets a means for moving the goods from A to B by using trk_1. With the latter, instead, the *seller* has an agreement with *pln* to ship goods from B to D. The last commitment, c_4, encodes an agreement between *pln* and trk_2 for the shipping of goods from C to D.

From a programming point of view, 2COMM allows a programmer to use directly these commitments to guide the implementation of both environment and agents. As concerns the environment, for each commitment, a corresponding commitment artifact is implemented in 2COMM. On the agent side, instead, a programmer is asked to implement the operations on the commitments that are of interest for the agent under development. For instance, while implementing *seller*, the programmer has to consider the creation of commitment c_1 and the reaction to the detachment of c_1 after the payment by the customer. In JADE, this process amounts to equipping each agent with proper behaviors to take the initiative of creating commitments, or to handle those events that, intercepted from 2COMM, are relevant for them (e.g., a commitment status change). The JaCaMo framework, in turn, adopts Jason as agent programming language. In this case, each agent has its own belief base, and a set of plans. Since JaCaMo already integrates CArtAgO, the 2COMM connector enables a direct mapping between the observable properties of commitment artifacts into beliefs of Jason agents. It follows that a change of state in a commitment can be used as a triggering event for agent plans. Similarly, operations on commitment artifacts are directly made available to agents, that can use them in their plans. Note how a programming pattern driven by social relationships emerges: the development of socially responsive JaCaMo agents amounts to equipping them with plans to properly manipulate the commitments in which they are involved.

4 Conclusions

2COMM supports programming heterogeneous interacting agents by following a uniform approach, which decouples the interaction and agent dimensions. Agent programming can leverage the conceptual architecture and general schema outlined in [4]. This feature fulfills the purpose of supporting the development of heterogeneous and open agent systems. Any agent can take part in an interaction with others by simply using properly defined commitment artifacts. The use of 2COMM with Jason and JADE proves that programming agents starting from their desired interaction can be a valuable starting point to build a general methodology useful for open and heterogeneous scenarios. The implementation of a scenario like the logistic one serves the purpose of showing the validity and suitability of the approach both from a conceptual and from a practical standpoint. Indeed, it provides a proof of concept to highlight the benefits coming from the development of socially-responsive agents in a realistic use case.

References

1. Baldoni, M., Baroglio, C., Capuzzimati, F.: A commitment-based infrastructure for programming socio-technical systems. ACM Transactions on Internet Technology **14**(4), 23:1–23:23 (2014)
2. Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Commitment-based Agent Interaction in JaCaMo+. Fundamenta Informaticae **159**(1-2), 1–33 (2018)
3. Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Type Checking for Protocol Role Enactments via Commitments. Journal of Autonomous Agents and Multi-Agent Systems **32**(3), 349–386 (2018)
4. Baldoni, M., Baroglio, C., Micalizio, R., Tedeschi, S.: Programming agents by their social relationships: A commitment-based approach. Algorithms **12**(4) (2019)
5. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE — A Java Agent Development Framework, Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 15, pp. 125–147. Springer (2005)
6. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Science of Computer Programming **78**(6), 747–761 (2013)
7. Capuzzimati, F.: A Commitment-based Infrastructure for Programming Socio-Technical Systems. Ph.D. thesis, Università degli Studi di Torino, Italy (2015)
8. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems **17**(3), 432–456 (2008)
9. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems **23**(2), 158–192 (2011)