

## AGREEMENT TECHNOLOGIES

### Towards a new programming paradigm for agent-oriented technologies

Juan A. Rodríguez-Aguilar, IIIA-CSIC, Spain

---

Nowadays, most current transactions and interactions at business level, but also at leisure level, are mediated by computers and computer networks. From email to virtual worlds, the way people work and enjoy their free time has changed dramatically in less than a generation time. However, the biggest impact of this pervasive use has been on the way applications are thought and developed. These applications require components to which more and more complex tasks can be delegated, components that show higher levels of intelligence, components that are capable of sophisticated ways of interacting, as they are massively distributed, sometimes embedded in all sort of appliances and sensors. This is precisely the scenario we believe agent-oriented technology can contribute to.

Therefore, there is a need for developing models, frameworks, methods and algorithms for constructing large-scale open distributed computer systems where *autonomy*, *interaction* and *mobility* are the key characteristics. We envision that such technologies can be structured around the concept of agreement among computational agents. We envisage a new programming paradigm that is based on two concepts: (1) a *normative context or agreement environment* [1], that determines the rules of the game, i.e. how the interactions between agents are going to happen, and (2) a *call-by agreement interaction method* that is based on a two step process: first the establishment of an agreement for action between the agents that respects the normative context, and second, the actual program call for the enactment of the action.

We believe that agent programming languages must go beyond the current state of the art, so far focused on the notions of agent architecture and protocol. Even beyond the ideas represented by WADE [2], which attempts at bringing the notion of process into agent development by providing support for the execution of tasks defined according to the workflow metaphor. There is a need for a new programming paradigm that considers the notions of *agreement environment* and *agreement* as first-class citizens. Thus, the new programming paradigm must allow programmers to create agreement environments, their access rules, their composition, and even the adaptation mechanisms that allow them to adapt under changing circumstances. Furthermore, the programming paradigm must allow the dynamic establishment of agreements, the verification of the fulfilment of agreements, and the management of agents that fail to honour their commitments even when agreements are signed. In some sense, we advocate for a new programming paradigm inspired on the way we humans act: we firstly set up constraining environments (via organisations or institutions) wherein social contracts among individuals or companies are dynamically established and honoured.

In order to found a new programming paradigm, we do not depart from scratch. Under the umbrella of agreement technologies we consider the techniques and tools that enable agents to reach and fulfil agreements on the mutual provision of services. For agreement technologies to succeed in building next generation open distributed systems there is a wide number of challenges at sight: the

semantic alignments between the different ontologies employed by agents; the need for negotiation to allow agents to reach agreements; the development of trust and reputation models that allow to cope with agents that fail to honour their commitments even when agreements are signed; formal models and tools for virtual organisations and institutions defining normative contexts, agreement environments, within which to reach agreements; the need for learning models to adapt agreement environments; the adaptability of agents to cope with different agreement environments that may even change over time; and a better understanding of agreement mechanisms by means of game and decision theoretic results.

In our view, agent-oriented software engineering has made (and is doing) so far significant contributions to industry. There is a wealth of agent-oriented methodologies (e.g. AUML), programming languages and agent-based platforms (e.g. JACK, Agentis, Whitestein's Living Systems), and actual-world applications (e.g. the wide range of business solutions developed by Whitestein). Nonetheless, we still believe that there is something missing. We envision that future methodologies, programming languages and tools for MAS-oriented development must grow around three fundamental notions, namely agreement environment, agreement, and agent. Thus, agents are situated in some agreement environment that constrains the dynamic enactment and fulfilment of agreements as reached by agents. Therefore, from a programmer's point of view we shall need to specify and enact environments, to specify and enact agents, and a machinery to process agreements as agents interact.

Notice that this approach is not far from the way we humans daily operate. Thus, we daily enact contracts in the framework of some regulatory body that shape our future interactions.

## References

- [1] Special issue on environment for multi-agent systems. *Autonomous Agents and Multi-agent Systems*. Volume 14, number 1. February 2007.
- [2] Workflows and Agents Development Environment. <http://jade.tilab.com/wade/>