Chapter 1
Agreement Technologies: A Computing perspective

Sascha Ossowski, Carles Sierra and Vicente Botti

Abstract In this chapter we analyse the concept of agreement from a Computing perspective. In particular, we argue that the capability of software components to dynamically forge and execute agreements at run-time will become increasingly important, and identify key areas and challenges that need to be addressed in order to advance in this direction. Finally, we introduce the emerging field of Agreement Technologies for the construction of large-scale open distributed software systems, and identify technologies that are in the sandbox to define, specify and verify such systems.

1.1 Introduction

In the past, the concept of agreement was a domain of study mainly for philosophers and sociologists, and was only applicable to human societies. However, in recent years, the growth of disciplines such as social psychology, socio-biology, social neuroscience, together with the spectacular emergence of the information society technologies, have changed this situation. Presently, agreement and all the processes and mechanisms involved in reaching agreements between different kinds of agents are also a subject of research and analysis from technology-oriented perspectives.

In Computer Science, the recent trend towards large-scale open distributed software systems has triggered interest in computational approaches for modelling and enacting agreement and agreement processes. Today, most transactions and interac-

Sascha Ossowski
CETINIA, University Rey Juan Carlos, Madrid, Spain, e-mail: sascha.ossowski@urjc.es

Carles Sierra
IIIA - CSIC, Barcelona, e-mail: sierra@iiia.csic.es

Vicente Botti
Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Valencia, Spain, e-mail: vbotti@dsic.upv.es

tions at business level, but also at leisure level, are mediated by computers and computer networks. From email, over social networks, to virtual worlds, the way people work and enjoy their free time has changed dramatically in less than a generation. This change has meant that IT research and development focuses on aspects like new Human-Computer Interfaces or enhanced routing and network management tools. However, the biggest impact has been on the way applications are thought about and developed. These applications require components to which increasingly complex tasks can be delegated, components that show higher levels of intelligence, components that are capable of interacting in sophisticated ways, since they are massively distributed and sometimes embedded in all sorts of appliances and sensors. These components are often termed *software agents* to stress their capability to represent human interests, and to be autonomous and socially-aware. In order to allow for effective interactions in such systems that lead to efficient and mutually acceptable outcomes, the notion of *agreement* between computational agents is central.

Over the last few years, a number of research initiatives in Europe and the USA have addressed different challenges related to the development and deployment of large-scale open distributed systems. One of the most related to the Action's goals was the Global Computing initiative (GCI) launched in 2001 as part of the FP6 IST FET Programme. The vision of the call, also contained in the Global Computing II (GCII) initiative, was to focus research on large-scale open distributed systems: a timely vision given the exponential growth of the Internet and the turmoil generated in the media and scientific fora of some international initiatives like the Semantic Web, and the peak of Napster usage in 2001 with more than 25 million users. Most projects had a highly interdisciplinary nature, and a large number of groups from theoretical computer science, agents, networks and databases worked together in a fruitful way. The focus of GCI was on three main topics: analysis of systems and security, languages and programming environments, and foundations of networks and large distributed systems. Along these lines, GCI projects dealt with formal techniques, mobility, distribution, security, trust, algorithms, and dynamics. The focus was ambitious and foundational, with an abstract view of computation at global level, having as particular examples the Grid of computers or the telephone network. The focus on GCII shifted towards issues that would help in the actual deployment of such big applications, namely, security, resource management, scalability, and distribution transparency.

Other approaches for large distributed systems (although with a limited degree of openness) include Peer-to-Peer (P2P) systems, where nodes in a graph act both as clients and servers and share a common ontology that permits easy bootstrapping and scalability, or Grid applications where the nodes in a graph share and interchange resources for the completion of a complex task. The Semantic Web proposal that has received large funding in the European Union and the USA is generating standards for ontology definition and tools for automatic annotation of web resources with meta-data. The size of the Semantic Web is growing at a fast pace. Finally, the increasing availability of web services has enabled a modular approach to solve complex systems by combining already available web services. The annotation of those through standards like WSDL or BPEL permits the automatic

orchestration of solutions for complex tasks. Combinations of Semantic Web and Web services standards have been carried out (SAWSDL, SESA) by standardization bodies such as the W3C and OASIS. And finally a strong social approach to developing new collaborative web applications is at the heart of the Web 2.0 and 3.0 initiatives (Flickr, Digg).

There are diverging opinions regarding the similarities and differences between services, agents, peers, or nodes in distributed software systems. The terms usually imply different degrees of openness and autonomy of the system and its elements. Nevertheless, our stance is that the commonality is rooted in the interactions that can, in all cases, be abstracted to the establishment of *agreements for execution*, and a subsequent *execution of agreements*. In some cases these agreements are implicit, in others they are explicit, but in all cases we can understand the computing as a two-phase scenario where agreements are first generated and then executed.

This chapter is organised as follows. Section 1.2 analyses the concept of agreement from a Computing perspective, discussing the different types of agreements and agreement processes related to software. It also outlines challenges that need to be addressed as software components become increasingly adaptive and autonomic. Section 1.3 introduces the field of Agreement Technologies, where the complex patterns of interaction of software agents are mediated by the notion of agreement. It describes and relates the different technologies and application areas involved, and provides pointers to subsequent parts and chapters of this book. Finally, Section 1.4 provides an outlook and concludes.

1.2 Agreement from a Computing perspective

The Computing Curricula 2005 Overview Report[1] defines Computing quite broadly as

> [...] any goal-oriented activity requiring, benefiting from, or creating computers. Thus, computing includes designing and building hardware and software systems for a wide range of purposes; processing, structuring, and managing various kinds of information; doing scientific studies using computers; making computer systems behave intelligently; creating and using communications and entertainment media; finding and gathering information relevant to any particular purpose, and so on. The list is virtually endless, and the possibilities are vast.

However, the same report acknowledges that "Computing also has other meanings that are more specific, based on the context in which the term is used". In the end, the open distributed systems mentioned in the introduction are software systems as well, so in this section we take a foundational stance and look into the role of agreement and agreement processes in relation to programs and software development. We also identify challenges that will need to be addressed in an open world where software components become increasingly adaptive and autonomic.

[1] `http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf`

1.2.1 Agreement and Software

Software development has traditionally been based on implicit agreements between programmers and language designers. Thanks to the formal semantics of programming languages specified by their designers, programmers are aware of the meaning of the computational concepts used to write and execute a program. For instance, programmers who use some imperative programming languages share the meaning of the concept of variable, value or loop as well as the notion of program state and state transition. There is no need for direct interaction between programmers and language designers — the meaning of the programming constructs is *fixed* before a particular program is designed, and will certainly remain *unchanged* during the execution of the program.

As software systems become bigger and more complex, more dynamic and explicit agreements between project leaders and programmers are needed, in order to establish and document the relationships between the different software components. These agreements (e.g. in the form of specifications of interface and behaviour) become then the basis for the subsequent implementation and verification of the resulting software product. They are established *dynamically* at design-time, as project leaders and programmers need to interact and discuss the component specifications. However, once such agreements are reached for all elements of the software system, they necessarily remain *unchanged* at execution-time.

But, when software systems and their elements become open, adaptive, and autonomic, it turns out to be impossible to explicitly define such agreements at design-time. Some software elements may need to interact with others that their programmer was unaware of at design-time. In addition, they cannot rely on complete functional descriptions of the elements that they interact with, as certain behaviours may vary at run-time in response to changes in the environment. Therefore, agreements need to forged *dynamically* at run-time, and there must be mechanisms for re-assessing and *revising* them as execution progresses.

Such agreements will need to rely on an explicit description of the interoperation between two independent pieces of code. They will certainly be multi-faceted and refer to different issues: to the meaning of the exchanged input/output variables, to the protocol to follow during the interaction and its exceptions, to the constraints to be respected during the computation (e.g. time, accuracy), etc. They will need to be generated at run-time by the two pieces of code themselves, perhaps by means of some particular type of built-in interaction between the software entities. This view requires that the interaction between two components starts by the generation (or perhaps selection) of the interoperation agreement, followed by a subsequent phase in which the actual interoperation of the parties takes place. Agreements can then evolve in a long term interoperation by further interaction between the computational entities. Agreements should become the basic run-time structures that determine whether a certain interaction is correct, in a similar way as type-checking currently determines if the values in a call to a procedure are correct. In summary, software components need to be "interaction-aware" by explicitly representing and reasoning about agreements and their associated processes.

1.2.2 Challenges

Software components willing to participate in open systems need extra capabilities to explicitly represent and generate agreements on top of the simpler capacity to interoperate. To define and implement these new capabilities, a large number of unsolved questions must be tackled that require a significant research effort and in some cases a completely new and disruptive vision. Following Sierra *et al.* [18], in the sequel we briefly outline a few areas where new solutions for the establishment of agreements need to be developed. They are key to supporting the phase in which autonomous entities establish the agreements to interoperate.

1.2.2.1 Semantics

The openness in the development of agents, components, or services creates the need for semantic alignments between different ontologies. Every component may have an interface defined according to a (not necessarily shared) ontology. Although standards are in place for ontology representation (e.g. OWL) there is currently no scalable solution to establish agreements between software entities on the alignment of their semantics. The sheer dimension of some ontologies and the large number of them available on the web makes it impossible to solve the alignment problem entirely by hand, so robust computational mechanisms need to be designed. Techniques that might bring light into the problem include: data mining of background knowledge for the alignment algorithms, information flow methods to align concepts, or negotiation techniques to allow agents or services to autonomously negotiate agreements on the meaning of their interactions. Agreements on semantics are of a very fundamental nature and their establishment is key for the success of truly open software systems in the long run [9].

1.2.2.2 Norms

The entities that interact with each other may have a behaviour that changes along time, and there may be different contexts within which to reach agreements. A way this context is defined and constrained is through the definition of conventions and norms regulating the interaction. What set of norms to use in an interaction is a matter of agreement between the entities. These and other considerations require that the code of entities be highly adaptive to its environment so that agreements including a normative context can be correctly interpreted and executed. This is not the case in current software development. For instance, most current approaches to service programming assume a static environment, and the classical approaches to code verification still focus on static verification techniques. Adaptive code is a necessity for the design of open distributed applications. In particular, programming will need to face issues like norm adoption and behaviour learning. Agreements are explicit and declarative, and thus they open the door to using model checking

and logic based approaches, like BDI. These techniques may make open entities norm-aware and endow them with the capacity to build cognitive models of their environment. For recent discussions see [1].

1.2.2.3 Organisations

Many tasks require the recruiting of agents or services to form teams or compound services. These software entities bring in different capabilities that, when put together, may solve a complex task. How many entities have to be involved and what tasks have to be associated to each one of them are difficult questions. Traditional planning systems and balancing algorithms are not well adapted due to the large search space and the high dynamics and uncertainty of an open environment where software entities may join or leave or stop behaving cooperatively at any time. Thus, new techniques need to be developed to underpin agreements between open and possibly unreliable computational resources in order to forge stable co-operation for the time needed to solve a task.

Business process modelling systems and languages (e.g. BPEL or BPEL4WS) [10] have made the interaction between activities and entities the central concept in software design. A detailed workflow regulates the activities and the combination of roles in an organisation as well as their associated data flow. The interaction between entities is modelled as a precise choreography of message interchanges. However, current approaches assume that the orchestration and choreography is external to the entities and static. In an open world the way entities will interact and be combined has to be determined on-the-fly. And this choreography in an evolving world must necessarily be part of the agreement between entities. Agreeing on the workflow of activities implies reaching agreements on the role structure, the flow of roles among activities, and most importantly the normative system associated to the workflow. In a sense the signing of an agreement between two entities is the decision on what workflow to follow. Techniques from the field of coordination models and languages are particularly promising in this context [14].

1.2.2.4 Negotiation

Most programming languages and methodologies base their semantics on a compositional view of code. Knowing the behaviour of the components, and how they are combined, we can know the overall behaviour. This approach is to a large extent not applicable to open software systems where the behaviour of the components cannot be totally known or analysed, and can only be observed. They are black boxes. Even though the behaviour of an entity can be restricted by the normative context and the agreements signed, it is not completely determined at component definition time. Moreover, setting agreements does not give total guarantees on behaviour: autonomy and self-interest may mean agents refrain from honouring their commitments

if there is a potential gain in so doing. New and radically different approaches are required to deal with this problem.

The interaction between software components depends on two types of dynamics. First, open systems evolve, new entities appear and disappear, and thus new agreements have to be set. Second, the rules of the game that regulate the interaction between two entities might change due to the establishment of new agreements between them and due to agreements with third parties. This dynamics is a true challenge as many traditional research solutions are based on a static world view (e.g. game theory, classic planning). Given that the entities are autonomous and black boxes to each other the only way agreements can be reached is via negotiation of its terms and conditions. Negotiation is the key technique to achieving a composition of behaviours capable of dealing with the dynamics of open software systems [7, 11]. Some of the challenges are how to efficiently negotiate the terms of an agreement, how to explore large spaces of solutions, how to establish trade-offs between the dimensions of the agreements or how to use the experience of human negotiation in software development. Computational models from the field of argumentation will also be relevant in facilitating and speeding up the process of reaching such agreements.

1.2.2.5 Trust

There are two basic security dimensions over open networks. The first is how to guarantee identity, and this is to a large extent solved by cryptographic methods. The second is how to guarantee behaviour. Entities sign agreements and these agreements have to be honoured by the signatories. In the case where the entities' code is available for inspection, recent results on Proof Carrying Code techniques provide an answer [12, 6]. These techniques permit the mobile code itself to define properties of its behaviour and to carry a formal proof that it satisfies the behaviour. Source code and properties are input to compilers that generate executable code and certificates which permit to verify that the code has not been tampered with. However, when the code is not mobile, as in the area of web services (where the service is executed remotely), the possibility of fraud and malevolent behaviour creates a security threat to applications. No definitive solution has been found yet.

Trust models summarise the observations of the execution of agreements and allow entities to decide whether to sign agreements again with the same entity or which entity to prefer for particular tasks [8]. Reputation measures are needed to bootstrap the signing of agreements between entities. There are two challenges that need to be addressed to guarantee behaviour: on semantics of the agreements and on social relations between entities. Trust and reputation models need to take into account semantic aspects of the agreements to permit entities to understand the relationship between past experiences and new ones. Social network measures are needed to understand the intentions of the entities and therefore predict their behaviour (e.g. they may be cheating to favour a friend). In this sense, the relationships

built over time among entities and/or their principals may also provide guarantees of behaviour [17].

1.3 Agreements among Software Agents

In this section, we shift our attention to open distributed systems whose elements are *software agents*. There is still no consensus where to draw the border between programs or objects on the one hand and software agents on the other [4, 22]. Perhaps the most commonly accepted characterisation of the term was introduced by Wooldridge and Jennings [21], who put forward four key hallmarks of agenthood:

- *Autonomy*: agents should be able to perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their own actions and their own internal state.
- *Social ability*: agents should be able to interact, when they deem appropriate, with other software agents and humans in order to complete their own problem solving and to help others with their activities where appropriate.
- *Responsiveness*: agents should perceive their environment (which may be the physical world, a user, a collection of agents, the INTERNET, etc.) and respond in a timely fashion to changes which occur in it.
- *Proactiveness*: agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate.

For the purpose of this chapter (and this book in general), we remark that, in order to show the aforementioned properties, the interactions of the software agent with its environment (and with other agents) must be guided by a reasonably complex program, capable of rather sophisticated activities such as reasoning, learning, or planning.

Our vision is a new paradigm for next-generation open distributed systems, where interactions between software agents are based on the concept of agreement. It relies on two main ingredients: firstly, a normative model that defines the "rules of the game" that software agents and their interactions must comply with; and secondly, an interaction model where agreements are first established and then enacted. Agreement Technologies (AT) refer to a sandbox of methods, platforms, and tools to define, specify and verify such systems. This book compiles the state of the art of research activities in Europe, and worldwide, working towards the achievement of the aforementioned goal.

This part of the book analyses the notion of agreement and agreement processes from different viewpoints, in particular from a perspective of Philosophy and Sociology of Law (Chapter 2) and of Cognitive and Social Science (Chapter 3), thus contributing to putting AT on solid conceptual foundations. Parts II to VI describe in detail the different technologies involved, while Part VII provides examples of real-world applications of AT and their potential impact. In what follows, we give a

short overview of the different technologies involved in AT and relate them to each other. We also outline how, by gluing together these technologies, novel applications in a variety of domains can be constructed.

1.3.1 Agreement Technologies

The basic elements of the AT sandbox are related to the challenges outlined in Section 1.2.2, covering the fields of semantics, norms, organisations, argumentation & negotiation, as well as trust & reputation. However, we are dealing with open distributed systems made up of software agents, so more sophisticated and computationally expensive models and mechanisms than the ones mentioned in Section 1.2.2 can be applied.

The key research fields of AT can be conceived of in a tower structure, where each level provides functionality to the levels above, as depicted in Figure 1.1.

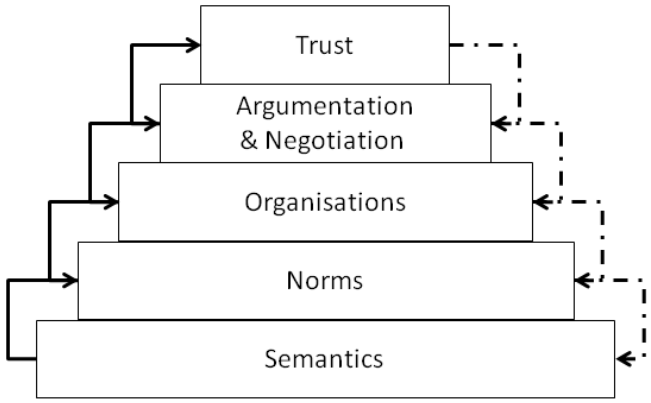

Fig. 1.1: AT Tower

Semantic technologies constitute the bottom layer, as semantic problems pervade all the others. Solutions to semantic mismatches and alignment of ontologies are essential, so agents can reach a common understanding on the elements of agreements. In this manner, a shared multi-faceted "space" of agreements can be conceived, providing essential information to the remaining layers. Part II of this book is dedicated to semantic technologies.

The next level is concerned with the definition of *norms* determining constraints that the agreements, and the processes leading to them, should satisfy. Thus, norms can be conceived of as a means to "shaping" the space of valid agreements. Norms may change over time, so support for the adaptation of the behaviour of software agents and of the normative system itself is to be provided. Part III of this book provides a survey of the field of norms.

Organisations further restrict the way agreements are reached by imposing organisational structures on the agents, defining the goals and capabilities of certain positions or roles that agents can play, as well as a set of relationships among them (e.g. power, authority). They thus provide a way to efficiently design and evolve the space of valid agreements, possibly based on normative concepts. Determining efficient workflows for teamwork that respect the organisational structures is also a concern at this level. Part IV of this book describes the state of the art in the field of organisations.

Then, the *argumentation and negotiation* layer provides methods for reaching agreements that respect the constraints that norms and organisations impose over the agents. This can be seen as choosing certain points in the space of valid agreements. Again, support for dynamicity is of foremost importance, so agreements can be adapted to changing circumstances. Part V of this book gives an overview of computational argumentation and negotiation models.

Finally, the *trust and reputation* layer provides methods to summarise the history of agreements and subsequent agreement executions in order to build long-term relationships between the agents. They keep track of as to how far the agreements reached respect the constraints put forward by norms and organisations. Trust and reputation are the technologies that complement traditional security mechanisms by relying on social mechanisms that interpret the behaviour of agents. Part VI of this book describes the state of the art this field.

Even though one can clearly see the main flow of information from the bottom towards the top layers, results of upper layers can also produce useful feedback that can be exploited at lower levels. For instance, as mentioned above, norms and trust an be conceived as a priori and a posteriori approaches, respectively, to security. Therefore, in an open and dynamic world it will certainly make sense for the results of trust models to have a certain impact on the evolution of norms.

In fact, such "direct relations" between layers are manifold, and different chapters of this book study them in detail. Chapter 15 is devoted to analysing the complex relationship between norms and trust. In Chapter 16 an overview of the existing work in the field of argumentation and norms is presented. Chapter 25 discusses how argumentation can be used in trust and reputation models and vice versa. Chapter 26 is devoted to relating ontologies, semantics and reputation, presenting several approaches to the problem of how agents can talk about trust and reputation among them. Finally, Chapter 28 analyses how reputation can influence different dimensions of an organization.

Some techniques and tools are orthogonal to the AT tower structure. The topics of environments and infrastructures [5], for instance, pervade all layers. In much the same way, coordination models and mechanisms [13, 15] are not just relevant to

the third layer of Figure 1.1, but cross-cut the other parts of the AT tower as well. Where appropriate, these tools and techniques are presented within the context of a particular application within Part VII of the book.

1.3.2 Key domains

There is no limitation *per se* regarding the domains where AT can be successfully applied. Still, in this section we have chosen three broad areas that we believe are particularly attractive in the context of this book: firstly, because the problems and challenges in these areas are of significant socio-economic relevance, so that applications based on AT can actually make a difference; and secondly because applications in these fields usually require the simultaneous use of several of the AT building blocks, thus illustrating the integration of the different layers outlined in the previous section.

E-Commerce is certainly a major application domain for AT, as the challenges for efficiently supporting business transactions neatly fit the building blocks of AT: negotiation and argumentation are often essential for agreeing on effective deals, transactions take place within a specific normative and organisational context, trust models are pervasive in electronic marketplaces, semantic matching and alignment is crucial to find and compare goods in a meaningful manner, etc. Part VII provides several examples of AT-based applications for e-Commerce. Chapters 32 and 36 describe applications that support Business-to-Business interactions in a dynamic and adaptive manner (to this respect, see also the comments on crowdsourcing in Chapter 2). Chapter 30 and 37 show how AT, and in particular argumentation models, can be used in Business-to-Consumer scenarios for customer support and product guidance, respectively.

Transportation Management is another candidate domain for applying AT-based solutions. Transportation is certainly a large-scale open distributed system, where the self-interested behaviour of agents (drivers, passengers, etc.) is organised by a set of norms (traffic rules), some transport modes are more reliable (trustworthy) than others, etc. In this context, Chapter 31 describes the on-the-fly generation and adaptive management of transport chains using AT. Chapter 35 illustrates how AT can be used to effectively manage fleets of ambulances. Another strand of work in the transportation field refers to next-generation intelligent road infrastructures. For instance, vehicle-to-vehicle and vehicle-to-infrastructure communication can be used for novel approaches to intersection management based on negotiation and auctions [19], and semantic technologies are essential for dynamically locating and selecting traffic information and management services [3].

AT are particularly well-suited for *E-Governance* applications as well, as it provides methods and tools for modelling, simulating, and evaluating processes and policies involving citizens and public administrations. Chapters 33 and 34 show how the electronic institution framework [2] (see also Chapter 18) can be used to

build a detailed model of real-world water rights markets, including negotiation and grievance procedures, as well as to simulate and evaluate their dynamics.

A plethora of other domains are candidates to become the playground for AT in the future. Among them, the field of *smart energy grids* is currently receiving much attention. In the energy grids of tomorrow, thousands or even millions of small-scale producers of renewable energy (solar, wind, etc.) will be distributed across the transmission as well as distribution networks and – taking into account certain norms and regulations – may decide to act together temporarily as *virtual power plants*. Through *demand side management* strategies (which may involve negotiation and trust models), neigbourhoods could coordinate (shift) their demands so as to adapt to the contingencies of intermittent renewable generation (see [16] for an overview). Some interesting overlap with the transportation domain becomes apparent when challenges such as coordinating the recharging process of large groups of electric vehicles, or employing the unused battery capacity of vehicles as a huge distributed storage facility come into play [20].

1.4 Outlook and Conclusion

In this chapter we have analysed the relevance of the notion of agreement and agreement processes from a Computing perspective. Open distributed systems are going to be the norm in software development, and the interoperation of software entities will need to rely on a declarative concept of agreement that is autonomously signed and executed by the entities themselves. We have presented a number of challenges for representation languages and programming techniques that need to be addressed in order to adequately support this type of software development. We then introduced Agreement Technologies as a sandbox of methods, platforms, and tools to define, specify and verify next-generation open distributed systems where interactions between software agents are based on the concept of agreement. The building blocks of AT comprise otherwise disparate research areas, such as semantics, norms, organisations, argumentation & negotiation, as well as trust & reputation. We show how they can be integrated into a natural tower structure, and provide examples of domains of socio-economic relevance, where new types of innovative applications were built by gluing together these technologies.

In this book we describe the state of the art of research and applications in the field of Agreement Technologies. Besides further progress in the development of methods and tools in the AT key areas, and an even tighter coupling of them, we would like to point to some complementary lines of work to be addressed in the future. First of all, a deeper integration of AT with the fields of programming and software engineering should be sought. This will require further formalisation and standardisation of AT, perhaps based on a graded notion of agreements and a set of alternative agreement processes of different complexity, so as to be able to balance expressiveness and computational complexity depending on the requirements of a

particular setting. The analysis of the relation of AT to Semantic Web standards put forward in Chapter 4 constitutes a step in this direction.

Another exciting enterprise refers to efforts for smoothening the boundary between the human space and the computational space, by extending the AT paradigm to include not just software agents but also *human agents*. For this purpose, the AT sandbox needs to be extended, so as to support systems in which humans work in partnership with highly inter-connected computational agents. Adaptation of the normative context, adjustable autonomy and recognition of user intentions are some of the characteristics that should be covered from a theoretical and practical perspective. Semantics, norms, argumentation, learning, behavioural modelling and human-agent collaboration are additional building blocks needed for the specification, enactment, and maintenance of such systems. The v-mWater application of Chapter 34, for instance, is specifically geared towards supporting the interaction between humans and software agents. Also, Chapter 21 on argumentation makes some steps in this direction: it not only describes methods and techniques for argumentation to aid machine reasoning but also shows how methods and techniques for argumentation can aid human reasoning.

Acknowledgements The term "Agreement Technologies" was introduced by Michael Wooldridge in conversations at the AAMAS conference in 2004. It was also used by Nicholas R. Jennings as title for a keynote talk given in 2005. Carles Sierra was among the first to give shape to the field by defining five key areas as technological building blocks for AT in 2007.

This work was partially supported by the Spanish Ministry of Science and Innovation through the project "Agreement Technologies" (CONSOLIDER CSD2007-0022, INGENIO 2010). The authors would like to thank Matteo Vasirani for inspiring discussions on the challenges of extending Agreement Technologies to mixed societies of human and software agents.

References

1. Andrighetto, G., Governatori, G., Noriega, P., van der Torre, L.: Dagstuhl Seminar Proceedings 12111: Normative Multi-Agent Systems. `http://www.dagstuhl.de/12111` (2012)
2. Arcos, J.L., Esteva, M., Noriega, P., Rodríguez, J.A., Sierra, C.: Engineering open environments with electronic institutions. Journal on Engineering Applications of Artificial Intelligence **18**(2), 191–204 (2005)
3. Fernández, A., Ossowski, S.: A Multiagent Approach to the Dynamic Enactment of Semantic Transportation Services. IEEE Transactions on Intelligent Transportation Systems **12**(2): 333–342 (2011)
4. Franklin, S., Graesser, A.: Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. Intelligent Agents III, Agent Theories, Architectures, and Languages. *Lecture Notes in Computer Science*, vol. 1193, pp. 21–35. Springer-Verlag: Heidelberg, Germany (1997)
5. García-Fornes, G., Hübner, J., Omicini, A., Rodríguez-Aguilar, J., Botti, V.: Infrastructures and tools for multiagent systems for the new generation of distributed systems. Engineering Applications of AI **24**(7): 1095–1097 (2011)
6. Hermenegildo, M., Albert, E., López-García, P., Puebla, G.: Abstraction carrying code and resource-awareness. In: Principle and Practice of Declarative Programming (PPDP-2005), pp. 1–11, ACM Press (2005)

7. Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Sierra, C., Wooldridge, M.: Automated negotiation: Prospects, methods and challenges. International Journal of Group Decision and Negotiation **10**(2), 199–215 (2001)
8. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems **43**(2), 618–644 (2007).
9. Kalfoglou, Y., Schorlemmer, M.: IF-Map – An ontology-mapping method based on information-flow theory. In: Spaccapietra, S., March, S., Aberer, K. (eds.) Journal on Data Semantics I, *Lecture Notes in Computer Science*, vol. 2800, pp. 98–127. Springer-Verlag: Heidelberg, Germany (2003)
10. Ko, R.K.L., Lee, S.S.G., Lee, E.W.: Business process management (bpm) standards: A survey. Business Process Management Journal **15**(5), 744–791 (2009)
11. Kraus, S.: Negotiation and cooperation in multi-agent environments. Artificial Intelligence **94**(1–2), 79–97 (1997)
12. Necula, G.C., Lee, P.: Proof-carrying code. In: 24th Symposium on Principles of Programming Languages (POPL-1997), pp. 106–119, ACM Press (1997)
13. Omicini, A., Ossowski, S., Ricci, A.: Coordination infrastructures in the engineering of multi-agent systems In: Methodologies and Software Engineering for Agent Systems – The Agent-Oriented Software Engineering Handbook, pp. 273–296. Kluwer (2004)
14. Ossowski, S, Menezes, R.: On coordination and its significance to distributed and multi-agent systems. Concurrency and Computation: Practice and Experience **18**(4): 359–370 (2006)
15. Ossowski, S.: Coordination in Multi-Agent Systems – Towards a Technology of Agreement. In: Multiagent System Technologies (MATES-2008), *Lecture Notes in Computer Science*, vol. 5244, pp. 2–12. Springer-Verlag: Heidelberg, Germany (2008)
16. Ramchurn, S., Vytelingum, P., Rogers, A. and Jennings, N.: Putting the "Smarts" into the Smart Grid: A Grand Challenge for Artificial Intelligence. Communications of the ACM, **55**(4), 86–97 (2012)
17. Sierra, C., Debenham, J.: Trust and honour in information-based agency. In: Proc. 5th Int. Conf. on Autonomous Agents and Multi Agent Systems, pp. 1225–1232. ACM Press (2006)
18. Sierra, C., Botti, V., Ossowski, S: Agreement Computing. Künstliche Intelligenz **25**(1): 57–61 (2011)
19. Vasirani, M., Ossowski, S.: A Market-Inspired Approach for Intersection Management in Urban Road Traffic Networks. Journal of Artificial Intelligence Research (JAIR) **43**: 621–659 (2012)
20. Vasirani, M., Ossowski, S.: A computational monetary market for plug-in electric vehicle charging In: Auctions, Market Mechanisms and Their Applications (AMMA 2011),*Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 80, Springer-Verlag: Heidelberg, Germany (2012)
21. Wooldridge, M., Jennings, N.: Intelligent Agents – Theory and Practice. Knowledge Engineering Review **10**(2): 115–152 (1995)
22. Wooldridge, M.: Agents as a Rorschach Test: A Response to Franklin and Graesser. Intelligent Agents III, Agent Theories, Architectures, and Languages. *Lecture Notes in Computer Science*, vol. 1193, pp. 47–48. Springer-Verlag: Heidelberg, Germany (1997)