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

Agreement Computing

Carles Sierra · Vicent Botti · Sascha Ossowski

Received: date / Accepted: date

Abstract In this paper we introduce the concept of Agreement Computing, motivate the central role that the concept of agreement plays in open software systems and discuss a number of research challenges that need to be addressed to make the agreement computing vision a reality.

Keywords Semantics · Norms · Organisations · Negotiation · Trust

PACS 07.05.Mh · 89.20.Ff

1 Introduction

Most current transactions and interactions at business and leisure levels 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. This change has made IT research and development focus on aspects like new Human-Computer Interfaces, enhanced routing, or network management tools. However, the biggest impact of this change has been on the way applications are thought and developed. Current applications require built-in 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, and quite often embedded in all sort of appliances and sensors. These autonomous components are usually termed agents to stress their capability of representing human interests, of being autonomous and socially-aware.

These trends in software development have motivated a number of research initiatives in Europe and USA in recent years. One of the most relevant to our vision is the Global Computing initiative launched in 2001 as part of the IST FET Programme. The aim of the call, also contained in the subsequent Global Computing II initiative, was to focus research on large-scale open distributed systems: a timely vision given the exponential growth of Internet and the turmoil generated on the media and scientific fora of some international initiatives like the Semantic Web, IBM's autonomic computing concept, 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 fructiferous 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 or the telephone network. Both functional and non-functional (e.g. QoS) properties were be studied. The focus on GCII was shifted towards issues that would help in the actual deployment of such big applications, namely, security, resource management, scalability, and distribution trans-

Carles Sierra
Artificial Intelligence Research Institute, IIIA-CSIC
E-mail: sierra@iiia.csic.es
Vicent Botti
Universitat Politècnica de València
E-mail: vbotti@dsic.upv.es
Sascha Ossowski
CETINIA, University Rey Juan Carlos
E-mail: sascha.ossowski@urjc.es

parency. Other initiatives for large distributed systems (although not necessarily *open* in our sense) include P2P systems, where nodes in a graph act as clients and servers and share a common ontology that permit 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 significant funding in the EU and the US 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 high pace. Finally, the availability of applications as web services has permitted an approach to solving complex systems by combining already available web services. The annotation of those through standards like WSDL or BPEL opens the door to the automatic orchestration of solutions for complex tasks. Combinations of Semantic Web and Web services standards are currently underway (SA-WSDL, SEE TC) within standardization bodies such as W3C and OASIS. The recent growth of cloud computing makes it even clearer that the distribution of services and resources is an unstoppable movement.

A lot of discussion has been made on what are the similarities and differences between services, agents, peers, or nodes in a grid. Our stance in this paper is that the commonality is on the interaction that can in all cases be abstracted to the establishment of an *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.

In this paper we introduce the vision of agreement and agreement computing in Section 2, and list some basic research challenges to realise the vision in Section 3. To keep the references within a reasonable limit, only survey-like and classical papers and books and some specially recent developments are included.

2 Agreements and agreement computing

Computing has been traditionally based on *implicit agreements* between programmers and language designers. In particular, agreements on the meaning of computational concepts used to design and execute a program. For instance, all programmers share the meaning of the concept of variable, value or loop as well as the notion of program state and state transition. Complex software development has been mostly based on *explicit agreements* between project leaders and programmers by means of which the relationships between the different software components are established and documented. These agreements (in the form of specifications of interface and behaviour) become then the basis for subsequent verification of the resulting software product.

But, when software systems and their elements becomes autonomic, adaptive, and open, these explicit agreements are simply non-existing. How can we build software that interoperates correctly and is thus reliable in those situations? How can we build complex solutions from simple components? We believe that current programming methodologies, languages, and tools need to evolve to incorporate an explicit notion of agreement between computational entities. Agreements are in our vision an explicit description of the interoperation between two independent pieces of code that is generated by the two pieces of code themselves. Agreements are to be generated by a particular type of built-in interaction between software entities. Software components willing to participate in open systems will therefore require to include extra capabilities to explicitly represent and generate these agreements, on top of the simpler capacity to interoperate, once the agreements are set. That is, 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. Agreement-checking is a run-time analysis of whether a particular interaction between two entities satisfies an agreement. Agreements are multi-faceted and include different sorts of agreements: on meaning of the exchanged variables, on constraints to be respected during the computation made by the entities, on properties of the values exchanged, on the particular protocol to follow, etc. This view requires that the interaction between two components starts by the generation (or perhaps selection) of the interoperation agreement and then a subsequent phase in which the actual interoperation of the parties involved. Agreements can evolve in a long term interoperation by further interaction between the computational entities. This capacity will make software components “interaction-aware” and can produce a significant step forward in software design and run-time verification if a number of scientific challenges are solved. In the next section we list the most significant ones and discuss possible venues of work leading to this vision.

3 Challenges

Although many efforts have been devoted in the projects and initiatives mentioned in the introduction, there is still a large number of unsolved questions that require a significant research effort and in some cases a completely new and disruptive vision. In this section we briefly outline a few areas where new technologies for the establishment of agreements need to be developed. These technologies are key to support the phase in which autonomous entities establish the agreements to interoperate.

3.1 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[7].

3.2 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 [3,12]. 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 need 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 use 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 [2].

3.3 Organisations

Many tasks require the recruiting of agents or services to form teams or compound services. These software entities bring in different capabilities that 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.

Techniques that may be promising in doing so are normative systems, virtual organisations and electronic institutions [14,10,1]. These techniques build "societies" that meet several requirements of open systems such as: distribution, constant evolution, flexibility to allow members to enter or exit the society, and multi-platform execution. Agreements include agreements on behaviour and the techniques mentioned above limit and constrain behaviour, thus they may help in reaching agreements by successive steps in which the organisation and dependencies are progressively established. How an organisation that regulates the co-operation and set the rules of the interaction is generated on-the-fly is an unsolved question.

Business process modelling systems and languages (e.g. BPEL or BPEL4WS) [8] 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 has to be necessarily part of the agreement between entities. Agreeing of the workflow of activities needs to contemplate on the one hand an agreement of the social 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. The area of electronic institutions has produced representation languages that can be the basis of this part of the agreements.

3.4 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 refrain agents 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 way agents and services interact 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 achieve a composition of behaviours capable of dealing with the dynamics of open software systems [5,9,15].

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. New negotiation and planning methods need to be proposed that take into account the changing structure and capabilities of the entities. Real-time planning is needed to account for the potentially high volatility of the entities in the system. Agreement planners have to be developed where the planning algorithm includes as part of the problem solving the establishment of agreements among entities. The embedding of these planners within virtual organisations and semantic alignment algorithms seems key to cover the different levels of agreements that might be required to solve a complex task.

3.5 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 case that the entities' code is available for inspection, recent results on Proof Carrying Code techniques provide an answer [11,4]. These techniques permit that mobile code brings within itself a property of behaviour and the proof that the code satisfies the behaviour. Code and properties are input to compilers that generate code and certificates that permit to verify that the code has not been tampered. 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 [6]. 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 along time among entities and/or their principals may also provide guarantees of behaviour [13].

4 Conclusion

In this paper we have presented a number of challenges that need to be addressed in order to realise the vision of Agreement Computing. Open distributed systems are going to be the norm in the software development industry of the future and the interoperation of the software entities will need to rely on a declarative concept of agreement that is autonomously signed and executed by the entities themselves. The notion of agreement allows to integrate otherwise disparate research areas. The generation of agreements between entities will need to integrate semantic, normative, organisation, negotiation and trust techniques. These five areas of research need to give answers to a large number of challenges sketched in this paper in order to provide the representation languages and the programming techniques necessary that will make Agreement Computing a reality.

Acknowledgements Research supported by the Agreement Technologies CONSOLIDER project under contract CSD2007-0022 and INGENIO 2010 and by the Agreement Technologies COST Action, IC0801.

References

1. 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), 191204 (2005)
2. Boella, G., Noriega, P., Pigozzi, G., Verhagen, H.: Dagstuhl seminar proceedings 09121: Normative Multi-Agent Systems (2009)
3. Henrik, G., Wright, V.: Norm and Action, A logical Enquiry. Routledge and Kegan Paul (1963)
4. Hermenegildo, M., Albert, E., López-García, P., Puebla, G.: Abstraction carrying code and resource-awareness. In: Principle and Practice of Declarative Programming. ACM Press (2005)
5. 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)
6. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. **43**(2), 618–644 (2007).
7. 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)
8. 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)
9. Kraus, S.: Negotiation and cooperation in multi-agent environments. Artificial Intelligence **94**(1–2), 79–97 (1997)
10. March, J.: Organizational Decision-Making, chap. A Preface to understanding how decisions happen in organizations. Cambridge University Press (1996)
11. Necula, G.C., Lee, P.: Proof-carrying code. Tech. rep. (1996)
12. Ross, A.: Directives and Norms. Humanities Press. (1968)
13. Sierra, C., Debenham, J.: Trust and honour in information-based agency. In: Proc. 5th Int. Conf. on Autonomous Agents and Multi Agent Systems, 1225–1232. ACM Press (2006)
14. Simon, H.A.: Administrative Behavior. Free Press. (1997)
15. Vasirani, M., Ossowski, S.: A market-inspired approach to reservation-based urban road traffic management. In: Proc. 8th Int. Conf. on Autonomous Agents and Multi Agent Systems, 617–624. IFAAMAS (2009)

Carles Sierra is Full Professor at the Institute of Research on Artificial Intelligence of the Spanish Council for Scientific Research (CSIC) and Adjunct Professor at the University of Technology, Sydney, Australia. He has participated in around fourty research projects funded by the European Commission and the Spanish Government, and has published more than two hundred and fifty papers in specialized scientific journals, conferences and workshops. His most recent work includes Electronic Institutions, negotiation, and Trust and Reputation models.

Vicent J. Botti Navarro is Full Professor at the Departamento de Sistemas Informáticos y Computación of Universidad Politécnica de Valencia (Spain) and head of the GTI-IA research group of this centre. He received his PhD in Computer Science from the same university in 1990. His research interests are multi-agent systems, agreement techonologies and artificial intelligence, where he has more than 200 refereed publications in international journals and conferences. He has been leader of several National and European research projects. Currently he is Vice-rector of the Universitat Politècnica de València.

Sascha Ossowski is Full Professor of Computer Science and Director of the Centre for Intelligent IT at University Rey Juan Carlos in Madrid. He obtained his MSc degree in Informatics from U Oldenburg in 1993, and received a PhD in Artificial Intelligence from TU Madrid in 1997. His research focuses on the application of distributed AI techniques to real world problems such as transportation management and smart grids. He is chair of the COST Action on Agreement Technologies, chairs the Board of Directors of the European Association for Multiagent Systems, and is a member of the steering committee of the ACM SAC conference series.