The Knowledge Engineering Review, Vol. 26:1, 53–59. © Cambridge University Press, 2011
doi:10.1017/S026988891000041X

Coordination models and languages: from parallel computing to self-organisation

ANDREA OMICINI and MIRKO VIROLI

DEIS–Dipartimento di Elettronica, Informatica e Sistemistica, Alma Mater Studiorum–Università di Bologna, via Venezia 52, 47521 Cesena, Italy;
e-mail: andrea.omicini@unibo.it, mirko.viroli@unibo.it

Abstract

Starting from the pioneering work on LINDA and Gamma, coordination models and languages have gone through an amazing evolution process over the years. From closed to open systems, from parallel computing to multi-agent systems and from database integration to knowledge-intensive environments, coordination abstractions and technologies have gained in relevance and power in those scenarios where complexity has become a key factor. In this paper, we outline and motivate 25 years of evolution of coordination models and languages, and discuss their potential perspectives in the future of artificial systems.

1 Introduction

Complex computational systems of today—intelligent, knowledge-intensive, pervasive, self-organising systems—could be seen as the dynamic ensemble of a large number of distributed components, heterogeneous in nature, structure and behaviour, put together *somehow* so as to build up a coherent overall system behaviour. Roughly speaking, that 'somehow' is typically the key issue in the research for abstractions, models, technologies and methodologies for the engineering of complex systems. In spite of their origin in the context of closed and parallel systems, *coordination models and languages* (Papadopoulos & Arbab, 1998; Busi *et al.*, 2001) are likely to play a key role there.

In this paper, we shortly recall the last 25 years of the literature on coordination models and languages, and show how they evolved to become the potential sources for the abstractions and the technologies around which complex computational systems will be designed and built.

2 Classical coordination models

Tuple-based models (Rossi *et al.*, 2001) represent the main class of space-based coordination models, where communication and coordination occur through a shared data space—as in the case of blackboard systems (Corkill, 1991). A shared communication space, whose life is independent of the interacting components, is the conceptual basis for *generative communication* (Gelernter, 1985): as such, it represents the essential environment abstraction for the support of openness in distributed systems. In fact, the very idea of a coordination abstraction persisting along with the messages exchanged is the essential prerequisite for a system where components may come and go at run-time, and provides for time uncoupling, which makes it possible to conceive and design patterns of interaction that could survive the potential erraticism of component behaviour.

Nevertheless, the ancestor of all tuple-based models—that is, LINDA (Gelernter, 1985), where components communicate and synchronise by exchanging tuples through a shared *tuple space*—was first conceived to support parallel computation in closed systems, at least with no apparent

concern for open systems. However, Gelernter and Carriero (1992) were soon aware of the conceptual consequences of the introduction of an environment abstraction devoted to the management of the agent interaction space: computation and *coordination*—there conceived as the governing of interaction (Wegner, 1997)—were to be considered as two orthogonal dimensions of computer-based systems, to be handled—that is, analysed, modelled, designed, programmed—in an independent way, by adopting suitable abstractions and mechanisms.

Although suspensive semantics of coordination primitives is an essential mechanism for LINDA and its derivatives (requests for tuples by components are served only when tuples matching the request template are actually available in the tuple space), other features of tuple-based models would pave the way for the coordination of complex systems. First of all, a tuple is an ordered collection of possibly heterogeneous knowledge chunks. Therefore, in short, synchronisation based on the availability of tuples essentially means synchronisation based on the availability of structured knowledge of some sort. As such, tuple-based coordination is first of all knowledge-based coordination, where tuple spaces are possibly interpreted as knowledge repositories. In addition, tuple spaces are accessed associatively: queries specify tuple templates that match tuples based on their structure and the data they contain. On the one hand, this provides for complete uncoupling in communication: information neither on the sender nor on the structure of the share space is required for a message to be received. On the other hand, the possibility to synchronise over a partial representation of knowledge (the tuple template) is a fundamental feature in all the contexts where information is often vague, inaccurate, incomplete or partially specified, as is typical in knowledge-intensive systems. Even more, in logic tuple-space models, such as Shared Prolog (Brogi & Ciancarini, 1991) and ReSpecT (Omicini & Denti, 2001), components coordinate through first-order tuples, and tuple spaces are first-order logic theories, thus pushing further the interpretation of the shared communication space as a knowledge repository used for component coordination.

Finally, two other features characterise tuple-based models as they descend from the original LINDA ancestor: distribution and expressiveness of the coordination abstractions—termed as 'reshaping the *coordination media*' and 'programming the *coordination rules*', respectively, by Busi *et al.* (2001). On the one hand, distribution is essential for any complex system: so, in the same way as components of a distributed system are spread all over the system topology, multiple tuple spaces fill the system environment, providing for distributed coordination abstractions—as in JavaSpaces (Freeman *et al.*, 1999) and TSpaces (Wyckoff *et al.*, 1998)—and paving the way towards pervasive coordination systems. The ability to express the environment topology in a distributed setting is then essential for supporting the coordination of local interaction as well as of mobile components, as in LIME (Murphy *et al.*, 2006) and KLAIM (De Nicola *et al.*, 1998).

On the other hand, expressiveness of coordination media often needs to be tailored to the complexity and peculiarity of the specific coordinated system. Therefore, a number of LINDA derivatives focus on the programmability of the tuple space, in order to make it possible to explicitly express the rules of coordination, and to embed them within the coordination abstraction—as in the case of Law-Governed Interaction (Minsky & Ungureanu, 2000), MARS (Cabri *et al.*, 2000) and ReSpecT (Omicini & Denti, 2001). There, arbitrarily complex coordination policies can in principle be associated with each of the coordination media, which can be individually programmed so as to embed either global or local coordination policies, as required by the specific coordinated systems. Finally, the ability to define arbitrarily complex coordination policies and to embed them within the coordination media should in principle be coupled with the ability to capture and react to arbitrary environment events—as in the case of Situated ReSpecT (Casadei & Omicini, 2009)—thus providing for the level of situatedness typically required by coordination in pervasive computational environments.

3 Nature-inspired coordination models

Among space-based models, one of the earliest examples is represented by Gamma (Banâtre *et al.*, 2001), originally presented in 1986. Even though not a derivative of LINDA, Gamma is a reminder

of the features of programmable tuple space models in that the shared space of coordination is ruled by chemical-like laws defined by the programmers. However, the main interest for Gamma here lies in its inspiration: as for the CHAM (chemical abstract machine) model (Berry, 1992), coordination in Gamma is conceived as the evolution of a space governed by chemical-like rules, globally working as a rewriting system.

More generally, a whole class of coordination models is inspired by the extraction of patterns from natural and social complex systems: *naturally inspired* coordination models are mostly driven by the idea that working complex systems exist in the real world, which we can observe so as to understand their basic principles and mechanisms, to abstract them, and to bring them within our artificial systems. In particular, understanding the principles and mechanisms of coordination within complex natural systems is seemingly an effective approach for the definition of coordination models and technologies for complex artificial systems. Therefore, for instance, *field-based coordination* models like TOTA (Mamei & Zambonelli, 2004) are inspired by the way masses and particles move and self-organise according to gravitational/electromagnetic fields (Mamei & Zambonelli, 2006). There, computational force fields in the form of distributed tuples, generated by the active components or by the pervasive coordination infrastructure, propagate across the environment, and drive the actions and motion of the component themselves, for example, allowing two mobile agents to find each other in a dynamic network.

Historically, nature-inspired models of coordination are grounded in studies on the behaviour of social insects like ants or termites. The key concept here is *stigmergy*, introduced by Grassé (1959) as an explanation for the coordination observed in termite societies, where '*The coordination of tasks and the regulation of constructions are not directly dependent from the workers, but from constructions themselves*'. Specifically, the notion of stigmergy generally refers to a set of coordination mechanisms mediated by the *environment*. For instance, in ant colonies, chemical substances, namely *pheromone*, act as environment markers for specific social activities and drive both the individual and social behaviour of ants—much like what happens, for example, in TOTA. These concepts have been generalised into *environment-based coordination*, which systematically adopts structured abstractions for shaping the environment of system components so as to govern their interactions. A notable example is that of *coordination artefacts* (Ricci *et al.*, 2005). Complex social behaviour like behavioural implicit communication can be built upon coordination artefacts (Omicini *et al.*, 2004), which work then as the general-purpose environment abstractions used to inject *social intelligence* within computational systems independently of the intelligence of the individual system components. According to Ricci *et al.* (2005), '*artifacts are environment abstractions that mediate agent interaction and enable emergent coordination: as such, they can be used to encapsulate and enact the stigmergic mechanisms and the shared knowledge upon which emergent coordination processes are based*'. Based on the use of artefacts as tools populating and structuring the agent working environment, *cognitive stigmergy* is introduced which supports multiple-level coordination between heterogeneous components: while ordinary components perceive environment markers as mere signals and react accordingly, intelligent components can read them as signs, and behave according to their symbolic interpretation.

4 Coordination in self-organising systems

Nature-inspired coordination models such as chemical, field-based and stigmergic ones share a fundamental feature: they come from the core of complex natural self-organising systems. As such, they are seemingly the most intuitive sources for abstractions and mechanisms around which self-organising artificial systems could be designed and built. Generally speaking, *self-organising coordination* can be defined as the management of system interactions featuring self-organising properties—namely, where interactions are local, and global desired effects of coordination appear by emergence (Viroli *et al.*, 2009). Although in most classical coordination models the environment is filled with coordination media enacting coordination laws that are typically reactive, (essentially) deterministic and global, in self-organising systems coordination patterns typically

appear at the global level by emergence from probabilistic, time-dependent coordination laws, based on local criteria.

Therefore, according to Viroli *et al.* (2009), the required features of coordination models for self-organising systems are topology and locality, online character, time-dependency and probabilistic behaviour. Topology and locality mostly affect the nature of the coordination middleware: the coordination media provided should be associated with distributed locations, and mostly govern interaction among local components; also, coordination media should not be merely reactive to interaction, but should instead be enacted as always-running services able to adapt their coordinative behaviour at run-time—as in the case of LIME, ReSpecT and TOTA, among others. As far as time-dependency and probabilistic behaviour are concerned, classical coordination models apparently address those issues in some way: for instance, tuple matching templates are returned in a non-deterministic way, while chemical laws are known to be probabilistic and time-dependent. Nevertheless, on the one hand, the non-determinism of a classical tuple-based model is just a 'do not know' non-determinism, whereas non-determinism in self-organising systems is typically stochastic. Accordingly, models like TOTA, SwarmLinda (Tolksdorf & Menezes, 2004) and STOKLAIM (Bravetti *et al.*, 2009) have introduced stochastic mechanisms within tuple-based coordination. On the other hand, classical chemical coordination models like Gamma and CHAM do not really reproduce chemical behaviours, since they can express neither stochastic behaviours nor time-dependent coordination rules. As a result, a *chemical tuple-space* model and infrastructure have been defined that embodies all the typical features of self-organisation in natural chemical systems (Viroli *et al.*, 2010). There, self-organisation could be achieved in two ways: either by means of the behaviour of an individual chemical tuple space (intra-space self-organisation) or by means of a suitable pattern of interaction among chemical tuple spaces (inter-space self-organisation).

Overall, suitably extended tuple-based models already provide quite a promising platform for the design and development of self-organising coordinated systems. Nonetheless, a daunting challenge for tuple-based models comes from knowledge-intensive application scenarios. Therefore, the aforementioned benefits of tuple-based coordination in terms of knowledge-based coordination fade in front of the problems it induces in terms of syntax—for example, two tuples containing the same data may not match due to differences in the tuple structure—and (mostly) of semantics—for example, two tuples representing the same information may not match based on a different syntax adopted. Till now, the problem has been faced according to two different perspectives: (i) by exploiting tuple-based coordination within a middleware for knowledge-intensive environments: for example, Tolksdorf *et al.* (2008) experiments with a tuple-based coordination within Semantic Web middleware, while Nixon *et al.* (2008) survey similar approaches; or, (ii) by enhancing the tuple space abstraction with a semantic interpretation: for example, Nardini *et al.* (2010) extend tuple spaces with a description logic framework so as to equip each tuple, template and operation over tuple spaces with a well-founded semantic interpretation.

The latter perspective leads to the definition of the notion of *self-organising semantic coordination* as a generalisation of the basic principles and mechanisms of coordination and self-organisation for application to knowledge-intensive environments. This can be defined as the management of interactions in knowledge-intensive systems, where interactions are local and involve sharing and processing of knowledge, and the global desired effects of coordination over distributed knowledge appear by emergence and through self-organisation. Coordination infrastructures—in particular, tuple-based ones—should then be adopted to support self-organising semantic coordination, as in the case of *eternally adaptive service ecosystems* for pervasive computing (Viroli & Zambonelli, 2010).

5 Perspectives and future directions

In this paper, we adopted the software engineering notion of the coordination model discussed in Omicini (2001). Accordingly, a coordination model works as the source of the abstractions and metaphors required to shape the space of component interactions in complex computational systems. Along this line, in this article, we briefly recapitulated the evolution of coordination

models in the last 25 years from closed, parallel systems towards open, intelligent, knowledge-intensive, pervasive and self-organising systems.

From the very beginning (Gelernter & Carriero, 1992), literature on coordination models promoted the separation between computation and coordination as two orthogonal dimensions in the modelling and engineering of complex computational systems (Wegner, 1997). According to this view, systems are built out of a (possibly huge) number of (possibly heterogeneous) computational components put together by means of environment abstractions—the coordination media—explicitly meant to govern the space of component interaction. The separation between computational and coordination components is at the core of the evolution of coordination models towards complex systems. Therefore, for instance, by exploiting the expressiveness of coordination abstractions, systems could be provided with social intelligence independently of the component intelligence, as in stigmergic coordination. However (possibly diverging), individual goals of autonomous components could be reconciled to achieve a global system goal without harming the autonomy of components—as in multi-agent systems. The tuple space abstraction constituted the basic brick for such evolution, promoting generative communication and uncoupled coordination. Then, its many developments towards distributedness, reactiveness, situatedness, time-dependence, probabilistic behaviour, etc. made coordination models rich enough to be possibly exploited as the sources for metaphors, patterns and mechanisms for complex systems such as knowledge-intensive, pervasive and self-organising systems.

Future directions of research are likely to further deepen the novel theoretical settings discussed here—like the one of self-organising semantic coordination—possibly devising original and even unexpected ones. In addition, the impact of coordination models on the engineering of complex systems is going to have profound implications on methodologies and software processes, and on related research as well.

Finally, a huge number of technical challenges are waiting to be faced by researchers engaged in the development of coordination middleware and infrastructures, by means of which the effectiveness of coordination-based approaches is going to be actually put to test against many complex, real-world application scenarios. Among the many challenges, we could mention the integration of organisational and security models in the coordination setting; the full development and testing of nature-inspired coordination models; the definition of knowledge-oriented coordination models and languages embodying international standards; the construction of light-weight coordination technologies for pervasive scenarios; and, finally, the design of rich coordination frameworks providing developers with a complete and expressive set of tools to harness the intricacy of engineering the space of interaction in complex computational systems.

Acknowledgements

This work has been supported by the EU-FP7-FET Proactive project SAPERE—Self-aware Pervasive Service Ecosystems, under contract no. 256874.

References

Banâtre, J.-P., Fradet, P. & Le Métayer, D. 2001. Gamma and the chemical reaction model: fifteen years after. In *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View*, Calude, C. S., Păun, G., Rozenberg, G. & Salomaa, A. (eds). Lecture Notes in Computer Science **2235**, 17–44. Springer.

Berry, G. 1992. The chemical abstract machine. *Theoretical Computer Science* **96**(1), 217–248.

Bravetti, M., Latella, D., Loreti, M., Massink, M. & Zavattaro, G. 2009. Combining timed coordination primitives and probabilistic tuple spaces. In *Trustworthy Global Computing*, Kaklamanis, C. & Nielson, F. (eds). Lecture Notes in Computer Science **5474**, 52–68. Springer.

Brogi, A. & Ciancarini, P. 1991. The concurrent language, Shared Prolog. *ACM Transactions on Programming Languages and Systems (TOPLAS)* **13**(1), 99–123.

Busi, N., Ciancarini, P., Gorrieri, R. & Zavattaro, G. 2001. Coordination models: a guided tour. In *Coordination of Internet Agents: Models, Technologies, and Applications*, Omicini, A., Zambonelli, F., Klusch, M. & Tolksdorf, R. (eds). Springer, 6–24.

Cabri, G., Leonardi, L. & Zambonelli, F. 2000. MARS: a programmable coordination architecture for mobile agents. *IEEE Internet Computing* **4**(4), 26–35.

Casadei, M. & Omicini, A. 2009. Situated tuple centres in ReSpecT. In *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, Shin, S. Y., Ossowski, S., Menezes, R. & Viroli, M. (eds). ACM, Honolulu, Hawai'i, USA.

Corkill, D. 1991. Blackboard systems. *Journal of AI Expert* **9**(6), 40–47.

De Nicola, R., Ferrari, G. & Pugliese, R. 1998. KLAIM: a kernel language for agent interaction and mobility. *IEEE Transaction on Software Engineering* **24**(5), 315–330.

Freeman, E., Hupfer, S. & Arnold, K. 1999. *JavaSpaces Principles, Patterns, and Practice: Principles, Patterns and Practices*. The Jini Technology Series, Addison-Wesley Longman.

Gelernter, D. 1985. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems* **7**(1), 80–112.

Gelernter, D. & Carriero, N. 1992. Coordination languages and their significance. *Communications of the ACM* **35**(2), 97–107.

Grassé, P.-P. 1959. La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux* **6**(1), 41–80.

Mamei, M. & Zambonelli, F. 2004. Programming pervasive and mobile computing applications with the TOTA middleware. In *Pervasive Computing and Communications. 2nd IEEE Annual Conference (PerCom 2004)*, Orlando, FL, USA, 263–273.

Mamei, M. & Zambonelli, F. 2006. *Field-based Coordination for Pervasive Multiagent Systems. Models, Technologies, and Applications*, Springer Series in Agent Technology, Springer.

Minsky, N. H. & Ungureanu, V. 2000. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)* **9**(3), 273–305.

Murphy, A. L., Picco, G. P. & Roman, G.-C. 2006. Lime: a coordination model and middleware supporting mobility of hosts and agents. *ACM Transactions on Software Engineering and Methodology* **15**(3), 279–328.

Nardini, E., Viroli, M. & Panzavolta, E. 2010. Coordination in open and dynamic environments with TuCSoN semantic tuple centres. In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, Shin, S. Y., Ossowski, S., Schumacher, M., Palakal, M., Hung, C.-C. & Shin, D. (eds). **III**, ACM, Sierre, Switzerland, 2037–2044.

Nixon, L., Simperl, E., Krummenacher, R. & Martin-recuerda, F. 2008. Tuplespace-based computing for the Semantic Web: a survey of the state-of-the-art. *Knowledge Engineering Review* **23**(2), 181–212.

Omicini, A. 2001. Coordination models and languages: state of the art. Introduction. Omicini, A. *et al.* (eds). Springer, 3–5.

Omicini, A. & Denti, E. 2001. From tuple spaces to tuple centres. *Science of Computer Programming* **41**(3), 277–294.

Omicini, A., Zambonelli, F., Klusch, M. & Tolksdorf, R. (eds). 2001. *Coordination of Internet Agents: Models, Technologies, and Applications*. Springer-Verlag.

Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C. & Tummolini, L. 2004. Coordination artifacts: Environment-based coordination for intelligent agents. In *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, Jennings, N. R., Sierra, C., Sonenberg, L. & Tambe, M. (eds). **1**, ACM, New York, USA, 286–293.

Papadopoulos, G. A. & Arbab, F. 1998. Coordination models and languages. In *The Engineering of Large Systems, Vol. 46 of Advances in Computers*, Zelkowitz, M. V. (ed.). Academic Press, 329–400.

Ricci, A., Viroli, M. & Omicini, A. 2005. Environment-based coordination through coordination artifacts. In *Environments for Multi-Agent Systems*, Weyns, D., Parunak, H. V. D. & Michel, F. (eds). Lecture Notes in Artificial Intelligence **3374**, 190–214. Springer.

Rossi, D., Cabri, G. & Denti, E. 2001. Tuple-based technologies for coordination. In *Coordination of Internet Agents: Models, Technologies, and Applications*, Omicini, A., Zambonelli, F., Klusch, M. & Tolksdorf, R. (eds). Springer, 83–109.

Tolksdorf, R. & Menezes, R. 2004. Using swarm intelligence in Linda systems. In *Engineering Societies in the Agents World IV*, Omicini, A., Petta, P. & Pitt, J. (eds). Lecture Notes in Computer Science **3071**, 49–65. Springer.

Tolksdorf, R., Nixon, L. & Simperl, E. 2008. Towards a tuplespace-based middleware for the Semantic Web. *Web Intelligence and Agent Systems* **6**(3), 235–251.

Viroli, M. & Zambonelli, F. 2010. A biochemical approach to adaptive service ecosystems. *Information Sciences* **180**(10), 1876–1892.

Viroli, M., Casadei, M. & Omicini, A. 2009. A framework for modelling and implementing self-organising coordination. In *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, Shin, S. Y., Ossowski, S., Menezes, R. & Viroli, M. (eds). **III**, ACM, Honolulu, Hawai'i, USA, 1353–1360.

Viroli, M., Casadei, M., Nardini, E. & Omicini, A. 2010, Towards a chemical-inspired infrastructure for self-* pervasive applications. In *Self-Organizing Architectures*, Weyns, D., Malek, S., de Lemos, R. & Andersson, J. (eds). Lecture Notes in Computer Science **6090**, chapter 8, 152–176. Springer.

Wegner, P. 1997. Why interaction is more powerful than algorithms. *Communications of the ACM* **40**(5), 80–91.

Wyckoff, P., McLaughry, S. W., Lehman, T. J. & Ford, D. A. 1998. T Spaces. *IBM Journal of Research and Development* **37**(3 – Java Techonology), 454–474.