Twenty Years of Coordination Technologies: COORDINATION contribution to the State of Art[☆]

Giovanni Ciatto[a], Giovanna Di Marzo Serugendo[b], Maxime Louvel[c], Stefano Mariani[d,*], Andrea Omicini[a], Franco Zambonelli[d]

[a]Alma Mater Studiorum–*Università di Bologna, Italy*
[b]*University of Geneva, Switzerland*
[c]*Bag-Era, France*
[d]*Università di Modena e Reggio Emilia, Italy*

Abstract

Complexity of intra- and inter-systems interactions is steadily increasing in modern application scenarios such as the Internet of Things, therefore coordination technologies are required to take a crucial step forward towards full maturity. In this paper we look back at the history of the COORDINATION conference series with the goal of shedding light on the current status of the coordination technologies there proposed throughout the years, also in comparison with other venues and industrial proposals, in an attempt to emphasise success stories as well as limitations, and possibly reveal a gap between actual technologies, theoretical models, and novel application needs.

Keywords: coordination technologies, middleware, survey

[☆] This paper is an extended version of paper *"Twenty Years of Coordination Technologies: State-of-the-Art and Perspectives"*, firstly appeared at COORDINATION 2018 (doi:10.1007/978-3-319-92408-3_3). Section 4 and Section 5 are brand new, Section 2 has been expanded w.r.t. the Logic Fragments model and technology, conclusions have been expanded to include discussion of open challenges.

[*] Corresponding author

Email addresses: `giovanni.ciatto@unibo.it` (Giovanni Ciatto), `Giovanna.DiMarzo@unige.ch` (Giovanna Di Marzo Serugendo), `maxime.louvel@bag-era.fr` (Maxime Louvel), `stefano.mariani@unimore.it` (Stefano Mariani), `andrea.omicini@unibo.it` (Andrea Omicini), `franco.zambonelli@unimore.it` (Franco Zambonelli)

Preprint submitted to Journal on Algebraic Methods in Programming December 18, 2019

1. Scope, Goal, and Method

Complexity of computational systems, as well as their impact on our everyday life, is constantly growing along with the increasing complexity of *interaction*—intra- and inter-systems. Accordingly, the role of *coordination models* should expectedly grow, too, along with the relevance of *coordination technologies* within ICT systems: instead, this is apparently not happening—*yet*.

Then, it is probably the right time (now, after twenty years of the COORDINATION conference series) to take a step back and reflect on what happened to coordination models, languages, and (above all) *technologies* in the last two decades. That is why in this paper we survey all the technologies that have been presented and discussed at the COORDINATION conference during the years, examine their stories and their current status, and try to provide an overall view of the state of art of coordination technologies as emerging from twenty years of work by the COORDINATION community. Also, to give a more meaningful and complete context to the survey, and to position it w.r.t. "the outside world", we include conferences closely related to COORDINATION, as well as related technologies proposed in the industry. The main goal is to provide a sound basis to answer questions such as: are coordination technologies ready for the industry? If not, what is currently missing? Which archetypal models lie behind them? Which are the research areas most/least explored? And what about the target application scenarios?

Although we aim at maximum *neutrality* by presenting the results of our survey, we hope that the data and insights here presented may serve as food for thought, and a fertile ground for further research in coordination technologies.

1.1. Structure & Contribution of the Paper

Section 2 provides at first an overview of the data about papers published in the conference throughout the years (Subsection 2.1), as collected from the official SpringerLink website and its companion BookMetrix service[1], with the

---

[1] `http://www.bookmetrix.com/`

aim of emphasising trends concerning *(i)* the number of *papers* published in each volume, *(ii)* the number of *citations* generated by each volume, *(iii)* the number of *downloads* generated by each volume, *(iv)* the *most cited* paper of each volume, and *(v)* the *most downloaded* paper of each volume.

Then, the scope of our analysis narrows down to those papers bringing a technological contribution, in the sense of describing a *software artefact* offering an API exploitable by other software to coordinate its components. Accordingly, Subsection 2.2 provides an overview of all the technologies presented within the COORDINATION conference series. For each one, the reference model implemented, and the web URL where to retrieve the software, if any, are given.

Then, a brief description of all the software for which no working implementation could be found is reported for the sake of completeness, whereas technologies still available are thoroughly described in Subsection 2.3. There, each one was downloaded and tested to clearly depict its *health status*: *(i)* date of last update to the source code (or project web page, if the former is not available), *(ii)* whether the software appears to be actively developed, in maintenance mode, or discontinued, *(iii)* availability of suitable documentation, *(iv)* availability of the source code publicly, *(v)* whether the build process of the software artefact is reproducible, and *(vi)* whether the software artefact, once built, executes with no errors. For the latter two items, in case of failures, an explanation of the problem and, if needed, the steps undertaken in the attempt to overcome it, are provided too. In particular, the latter test is not meant to measure performance, or, to provide a benchmark for comparisons: its purpose is to assess whether the technology is *usable*, that is, executable on nowadays software platforms and by nowadays programming languages. For instance, an artefact requiring an obsolete third-party library that hinders smooth deployment is considered not usable. Accordingly, each technology is tested either by running provided example code, or by developing a minimal working example of usage of the API.

Section 3 discusses the data collected so as to deliver insights about: *(i)* the *evolution* of technologies as they are stemming from a few archetypal models

(Figure 5), *(ii)* the *relationships* between the selected technologies, as a *comparison* of their features (Figure 6), and *(iii)* the *main goal* and *reference scenario* of each technology (Figure 7). Also, a general discussion is provided, reporting about success stories, peculiarities, and opportunities. Then, Section 4 and Section 5 relate the survey to, respectively, *(i)* other reference conferences often attended by researchers within the COORDINATION community, and *(ii)* industrial practice, so as to deliver insights about the relevance of COORDINATION results w.r.t. "the outside world".

Finally, Section 6 concludes the paper by summarising the results of the survey and providing some perspectives for future research activities concerned about coordination technologies.

1.2. Method

The scope of this survey is indeed mostly the COORDINATION conference series. We focus on coordination *technologies* intended as software implementing a given coordination model, language, mechanism, or approach with the goal of providing coordination *services* to other software applications. In other words, our focus is on technologies implementing some form of *coordination middleware* or *API*—analysed in Subsection 2.2. We nevertheless include in our overview other technologies presented within COORDINATION (Subsection 2.1), such as *simulation* frameworks, *model-checking* tools, and proof-of-concept implementations of *process algebras*—which are only described in short, for the sake of completeness.

Starting from the COORDINATION conference proceedings available online from SpringerLink[2], the survey proceeds as follows:

1. for each conference year, papers describing a coordination-related technology were gathered manually into a Google Spreadsheet
2. for each collected paper, we checked whether the paper was actually proposing some software package—papers failing the test are omitted

---

[2] `http://link.springer.com/conference/coordination`

3. for each paper passing the test, we verified the health *status* of the technology—as described in Subsection 1.1
4. then, for each paper featuring at least a *usable* distribution (downloadable and runnable) the corresponding software was downloaded and tested—i.e., installation & basic usage

The same process is applied to the technologies gathered from the other venues considered beyond COORDINATION, as listed in Section 4.

2. The Survey

Although the focus of this paper are coordination technologies, an overview of the whole conference proceedings is due to give context to the survey itself. Accordingly, Subsection 2.1 summarises and analyses all the data officially available from Springer—concerning, for instance, citations and downloads of each volume and paper. Then, Subsection 2.2 accounts for all the coordination technologies mentioned in COORDINATION papers, regardless of their actual availability, while Subsection 2.3 reports about the core of this survey: the status of the coordination technologies nowadays publicly available.

2.1. Overview

The COORDINATION conference series has been held 20 times since its first edition in 1996 in Cesena (Italy) until last year surveyed (2018[3], in Madrid, Spain) and generated as many conference proceeding volumes, all available online[2]. Data about the number of published papers, the number of *citations* and *downloads* per year of each volume, as well as the *most cited* and *most download* paper have been collected from SpringerLink and its companion service BookMetrix—and are reported in Table 1 on page 6 (last checked August 23rd, 2019). Highest values for each column are emphasised in bold.

---

[3] The 20th edition (2018), at which this survey appeared first. The 2019 edition has no data available from Springer, yet, hence has been left out of the survey.

Edition	No. of papers	Citations/Year	Downloads/Year	MCP	MDP
1996	34	12.17	139.13	**54**	132
1997	31	10	132.73	39	149
1999	32	8.20	236.00	25	188
2000	27	3.68	217.37	7	177
2002	**35**	9.18	338.24	14	207
2004	23	13.20	220.67	53	182
2005	19	7.43	311.43	17	261
2006	18	14	362.31	52	353
2007	17	19	367.50	48	367
2008	21	**20.55**	417.27	32	261
2009	15	14.90	397.00	28	306
2010	12	6.89	562.22	11	726
2011	14	8.38	567.50	14	743
2012	18	11.14	1 144.29	15	625
2013	17	15.17	1 396.67	15	**763**
2014	12	19	794.00	18	380
2015	15	9	1 337.50	12	411
2016	16	16	2 120.00	10	473
2017	14	12	2 260.00	7	362
2018	12	19	**2 910.00**	7	326
Avg.	20.10	12.44	811.59	23.90	369.60
Std. Dev.	7.62	4.71	801.71	16.59	200.74

Table 1: Overall data directly available online from Springer regarding the COORDINATION conference series. To compute citations (downloads) per year, the number of citations (downloads) was divided by the number of years the publications is available since. MCP stands for "Most Cited Paper" whereas MDP stands for "Most Downloaded Paper".

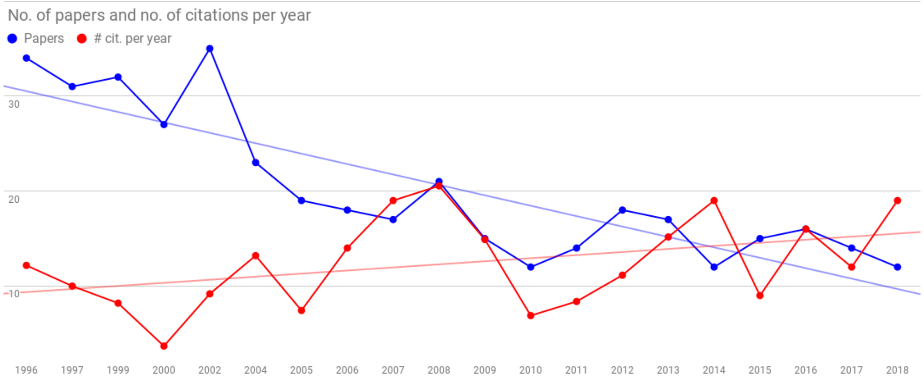

Figure 1: Number of papers in the volume and number of citations per year (computed as described in text) of the volume.

The trend over time of the number of papers, the citations of the volumes, and their downloads, are plotted in Figure 1 and Figure 2, respectively, along with their trend line. A few significant trends can be spotted in spite of the high variability between different editions of the conference. For the number of published papers, the trend is clearly *descending*: the first five editions featured an average of 32 papers, whereas the latest five an average of 14. As far as the number of citations per year generated by each volume of the proceedings is concerned, a few oscillations can be observed:

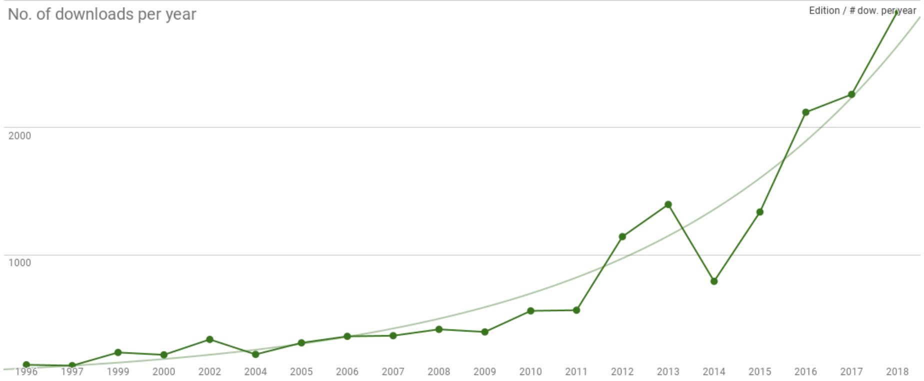

Figure 2: Number of downloads per year (computed as described in text) of the volume.

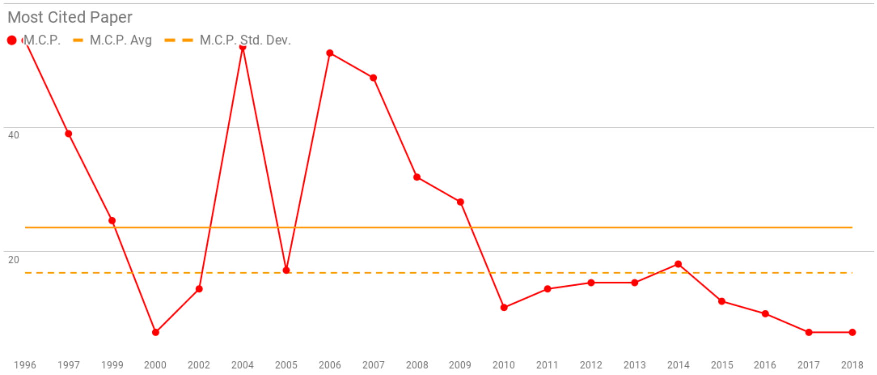

Figure 3: Most cited paper per year with average values & standard deviation.

- a first phase (from the 1st edition to the 4th) shows a *decreasing* number of citations, from 12.17 down to 3.68 (the *all-time-low*)
- then, in a second phase (from the 5th to the 10th edition) the number of citations *increases*, up to the *all-time-high* of 20.55 in 2008
- a third phase where the number of citations per year kept steadily increasing up to 2014 (19) started after a brief fall in 2009 and 2010
- finally, the last four editions show no clear trend as they alternate below and above average figures (average being 12.44)

For the number of downloads per year, two phases can be devised out in Figure 2:

- in the first period, from the 1st edition to the 13th (2011), the trend is quite *stable*, oscillating between 139.13 and 567.5
- in the second one instead, from 2012 up to latest edition, there is a *sharp increase* up to the *all-time-high* of 2 910 in 2018

Finally, Figure 3 and Figure 4 show the most cited paper and the most downloaded paper per year, respectively. For the former two main phases can be devised out: the first one starts with the first edition in 1996 and concludes with the 12th in 2010, durning which high and low figures (way above and below average) alternate quite frequently, whereas a second one exhibits a more regular trend where citations are rather low (namely, below average, even when considering the standard deviation)[4]. For the latter, instead, three epochs may be defined: a first one with a slowly increasing number of downloads per year, from the 1st to the 11th, a second one featuring a sharp increase (from 306 to 726 in just one year) holding still for a few editions (until 2013), finally a third one following a sharp decrease in 2014 that stabilises the figures around the average[4].

Besides these raw numbers, it is interesting w.r.t. the technological focus of this survey to check how many of such papers are related to technology. Overall, in the 20 editions of COORDINATION held, the most cited / downloaded paper is about technology – in the broadest acceptation of the term – in *slightly less then a half* of them: 7 papers amongst the most cited ones, and 8 amongst the most downloaded ones. By extending the analysis to all the papers published in the proceedings, instead, out of all the 402 papers published, only 49 (just 12.19%) convey a technological contribution—based on authors' inspection of the papers. And, such an estimate is somehow optimistic, since we counted papers just for merely *mentioning* a technology, with no means to access it—see Table 2, starting right below. This suggests that although technologies are seldom proposed at COORDINATION, they are quite impactful nevertheless.

---

[4] Keep in mind that the most recent the edition, the more time is needed to generate impact.

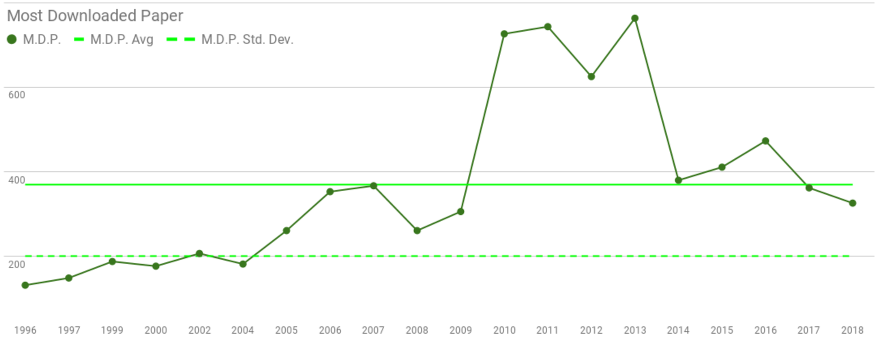

Figure 4: Most downloaded paper per year with average values & standard deviation.

Table 2: Overview of the coordination technologies presented at COORDINATION. “Name” denotes the technology, whereas “Model” makes explicit the model taken as reference for the implementation. The last column points to the web page where the software is available, if any, and provides for additional notes.

Name	Year	Model	(Closest) Web page & Notes
Manifold [1]	1996	IWIM [1]	`http://projects.cwi.nl/manifold` *no link to implementation*
Sonia [2]	1996	Linda + access control	*no implementation found*
Laura [3]	1996	service-oriented Linda	*no implementation found*
MultiBinProlog [4]	1996	$\mu^2 Log$ [4]	`http://cseweb.ucsd.edu/~goguen/courses/230/pl/art.html` *dead links*
MESSENGERS [5]	1996	Navigational Programming [5]	`http://www.ics.uci.edu/~bic/messengers` *dead links*
$\mathcal{ACLT}$ [6]	1996	Linda + programmable tuple spaces	*evolved into* TuCSoN
Blossom [7]	1997	Linda + coordination patterns	*no implementation found*
Bonita [8]	1997	asynch Linda	*no implementation found*
Berlinda [9]	1997	Linda	*no implementation found*
SecOS [10]	1999	Linda	*no implementation found*
Messengers [11]	1999	CmPS + mobility [12]	http://osl.cs.illinois.edu/software/ *no mention of “Messengers”*
MJada [13]	1999	OO Linda	`http://www.cs.unibo.it/cianca/wwwpages/macondo/` *no reference to MJada*
STL++ [14]	1999	ECM [14]	*no implementation found*
Clam [15]	1999	IWIM [1]	*no implementation found*
TuCSoN [16]	1999	*novel* (many extensions to Linda)	`http://tucson.unibo.it/`

Name	Year	Model	(Closest) Web page & Notes
Truce [17]	1999	*novel* (protocols + roles)	*no implementation found*
CoLaS [18]	1999	*novel* (protocols + roles)	*no implementation found*
OpenSpaces [19]	2000	OO Linda	*no implementation found*
Piccola [20]	2000	*novel*	`http://scg.unibe.ch/research/piccola`
Moses [21]	2000	LGI [21]	`http://www.moses.rutgers.edu`
Scope [22]	2000	Linda + mobility + space federation	*no implementation found*
$P\epsilon\omega$ [23]	2002	IWIM [1]	`http://reo.project.cwi.nl/reo` *evolved into Reo*
SpaceTub [24]	2002	Linda	*no implementation found*
O'Klaim [25]	2004	Klaim [26]	`http://music.dsi.unifi.it/xklaim` `https://github.com/LorenzoBettini/xKlaim`
Limone [27]	2004	Linda + mobility + spaces federation	`http://mobilab.cse.wustl.edu/projects/limone`
CRIME [28]	2007	Lime [29]	`http://soft.vub.ac.be/amop/crime/introduction`
TripCom [30]	2007	Triple Space Computing [31]	`http://sourceforge.net/projects/tripcom`
CiAN [32]	2008	*novel*	`http://mobilab.cse.wustl.edu/Projects/CiAN/Home/Home.shtml`
Smrl [33]	2008	Pepa [34]	`http://groups.inf.ed.ac.uk/srmc/download.html`
CaSPiS [35]	2008	IMC [36]	`http://sourceforge.net/projects/imc-fi`
LeanProlog [37]	2008	*novel*	`http://www.cse.unt.edu/~tarau/research/LeanProlog`
JErlang [38]	2010	Join-Calculus [39]	`https://tinyurl.com/yyggw4wx` (through Wayback Machine)
Session Java [40]	2011	Session Types [41]	`http://www.doc.ic.ac.uk/~rhu/sessionj.html`
WikiRecPlay /InFeed [42]	2012	BPM	*no implementation found*
Statelets [43]	2012	*novel*	`http://sourceforge.net/projects/statelets`
IIC [44]	2012	Reo [23]	`http://github.com/joseproenca/ip-constraints`
LINC [45]	2015	Linda [46]	*implementation not available for commercial reasons* *see http://bag-era.fr/index_en.html#about*

Name	Year	Model	(Closest) Web page & Notes
RepliKlaim [47]	2015	Klaim [26]	`http://sysma.imtlucca.it/wp-content/uploads/2015/03`
Logic Fragments [48]	2014	Sapere [49]	`https://www.unige.ch/cui/cas/publications/projects-output/`

2.2. Technologies at a glance

Table 2 provides an overview of the coordination technologies presented within the COORDINATION conference series throughout the years. Only those technologies passing test §2 in Section 1.2 are included, that is, those technologies actually delivering some form of *coordination services* to applications—i.e. in the form of a *software library* with suitable API. For each technology, the original paper is referenced, the *model* taken as reference for implementation indicated, if any, and the URL to the web page hosting the software given—if any is still reachable. Technologies whose corresponding software is still available – that is, those passing test §3 in Section 1.2 – are further discussed in Subsection 2.3; those with no working software found are briefly described in the following, for the sake of completeness.

The early days. The first few years of COORDINATION (1996–2000) saw a *flourishing* of successful technologies: some of the ideas introduced back then are still alive and healthy. For instance, $\mathcal{ACLT}$ [6] adopted *first-order logic terms* as Linda tuples, an intuition shared by the μ^2*Log* model and its language, MultiBinProlog [4]. Also, $\mathcal{ACLT}$ allowed agents to *dynamically program* tuple spaces via a specification language, enabling definition of computations to be executed in response to some *events* generated by interacting processes. Both features influenced the TuCSoN model and infrastructure [16], one of the few technologies to be still maintained nowadays.

Similarly, the IWIM coordination model and its corresponding language, Manifold [1], were introduced back in 1996 and survived until present days by evolving into Reo [23]. IWIM came by recognising a dichotomy between *exogenous* and *endogenous* coordination, and exploiting *channel composition* as a means to build increasingly complex coordination patterns by incrementally composing simpler ones.

Finally, Moses [21] was presented to the COORDINATION community as an infrastructure reifying the *Law Governed Interaction* (LGI) model. The technology is still alive and inspectable from its homepage, even if apparently no longer maintained. Analogously, the Piccola composition language presented in [20] clearly relies on a coordination technology which reached stability and robustness, even if it seems to be no longer maintained, too.

Besides these success stories, many other papers at that time proposed a technology, but either they only mentioned the technology without actually providing a reference to a publicly available software, or such a reference is no longer reachable (i.e. the link is dead and no reference to the software has been found on the web). For instance:

Sonia [2] — a LINDA-like approach supporting *human workflows*, therefore stressing aspects such as understandability of the tuple and template languages, time-awareness and timeouts, and security by means of access control

Laura [3] — a language attempting to steer LINDA towards *service-orientation*, where tuples can represent (formal descriptions of) service requests, offers, or results, thus enabling loosely coupled agents to cooperate by means of Linda-like primitives

MESSENGERS [5] — following the *Navigational Programming* methodology [5], where strongly-mobile agents (a.k.a. *Messengers*) can migrate between nodes. Here, coordination is seen as "invocation [of distributed computations] and exchange of data" and it "is managed by groups of Messengers propagating autonomously through the computational network"

Blossom [7] — a LINDA variant focusing on safety, which is provided by supporting a *type system* for tuples and templates, and a taxonomy of access patterns to tuple spaces, aimed at supporting a sort of "least privilege" principle w.r.t. access rights of client processes

Bonita [8] — another LINDA-like technology (as its successor WCL [50]) focusing on *asynchronous* primitives and distribution of tuple spaces, which can also migrate closer to their users

Berlinda [9] — providing a *meta-model*, along with a Java implementation, for instantiating different LINDA-like models

SecOS [10] — a LINDA variant focusing on *security* and exploring the exploitation of (a)symmetric key encryption

Messengers [11] — not to be confused with [5] despite its name, this focusses on *message exchange* by means of migrating actors

MJada [13] — an extension of the Jada language [51], focusing on coordinating concurrent (possibly distributed) Java agents by means of LINDA-like tuple spaces with an extended primitive set and *object-oriented tuples*

Clam [15] — a coordination language based on the IWIM model [1]

Truce [17] — a scripting language aimed at describing *protocols* to which agents must comply by enacting one or more *roles*

CoLaS [18] — a model and its corresponding language providing a framework where a number of *participants* can join interaction *groups* and play one or more *roles* within the scope of some coordination *protocol*. In particular, CoLaS focuses on the enforcement of coordination rules by validating and constraining participants behaviour

Much of the efforts are thus devoted at expanding LINDA along different dimensions, especially security.

The millenials. After year 2000, technologies are *less* present amongst COORDINATION papers, but not necessarily less important. For instance, Reo made its first appearance in 2002 [23], its name written in Greek ($P\epsilon\omega$). Reo provides an *exogenous* way of governing interactions between processes in a concurrent and possibly distributed system. Its strength is due to its *sound semantics*, enabling researchers to formally verify system evolution, as well as to the availability of *software tools*. The technology is indeed still alive and actively developed.

Recent implementations are also more easily available on the web. Out of 22 coordination technologies, only 5 were not found on the web during the survey:

OpenSpaces [19] — focussing on the harmonisation of the LINDA model with the OOP paradigm and, in particular, with the inheritance mechanism

Scope [22] — analogously to Lime, it provides multiple distributed tuple spaces cooperating by means of local interactions when some process attempts to access a tuple, thus providing a sort of federated view on the tuple space

SpaceTub [24] — successor of Berlinda, it aims at providing a meta-framework where other LINDA-like frameworks can be reproduced

WikiRecPlay / InFeed [42] — a pair of tools (browser extensions, no longer available) aimed at extracting and manipulating information from web applications to record them and later *replay*, enabling the definition of sequences of activities that can be *synchronised* with each other. The goal here is to augment *social software* with coordination capabilities

LINC [45] — a coordination environment implementing the basic LINDA primitives (`out`, `in`, `rd`) in a setting in which each tuple space (called bag) could implement the primitives differently (still preserving semantics), a convenient opportunity when dealing with *physical devices* (i.e. in the case of deployment to IoT scenarios) or *legacy systems* (i.e. databases). It provides *transactions* to alleviate to developers the burden of rolling back actions in the case of failures, and a chemical-reaction model inspired to Gamma [52] for enacting *reaction rules*. Several tools [53] are provided to help developers debug the rules, and to generate rules from high level specifications. The LINC software is nevertheless not publicly available because it is exploited by the Bag-Era company. Accordingly, it is not further analysed in Subsection 2.3, but it is included in Section 3 as an example of industrial exploitation

All other technologies are still publicly available, thus further analysed in next section.

For instance, the O'Klaim language presented in [25] is a linguistic extension of Klaim [26] with object-oriented features. Despite the reference paper describing Klaim has been published on the IEEE Transactions on Software Engineering in 1998, we were able to trace back a preliminary work on which appeared in the COORDINATION conference in [54]. Interestingly, the Klaim language soon evolved in X-Klaim [55], whose technology is still available[5]. Furthermore, the X-Klaim technology has been recently renewed by means of the Xtext language toolkit[6], and the project reboot is available on GitHub[7] [56].

Similar considerations can be made for Limone [27] and CRIME [28], which both stem from the idea of *opportunistic federation* of transient tuple spaces introduced by LIME [57], and improve it with additional features such as lightweightness and orientation to ambient-programming.

Analogously, the CiAN [32] model and middleware, targeting the coordination of distributed workflows over *M*obile *A*d-hoc *Net*works (MANETs), comes with a mature implementation, although no longer maintained. An extension to Session Java [58] is proposed in [40] to explicitly tackle synchronisation issues such as freedom from deadlock via multi-channel session primitives. Whereas the implementation was discontinued in 2011[8], the source code is still available from GoogleCode archive. JErlang [38], an implementation of Erlang extended with constructs borrowed from the JOIN-CALCULUS [39], appears to be no longer maintained too although a couple of implementations are still available and (partially) working.

---

[5] `http://music.dsi.unifi.it/xklaim`

[6] `https://www.eclipse.org/Xtext/`

[7] `https://github.com/LorenzoBettini/xKlaim`

[8] Year of latest commit: `https://code.google.com/archive/p/sessionj`.

Also **RepliKlaim** [47] – another variant of KLAIM [26] aimed at optimising performance and reliability through *replication* of tuples and tuple spaces – received updates until 2015 as far as we know, thus appears to be discontinued. Likewise, 2015 is the year when both Statelets [43] and IIC [44] received their last known update: the former is a programming model and language aimed at integrating *social context*, *social networks analysis*, and *semantic* relationships amongst shared artefacts into a single and coherent coordination model, while the latter proposes *Interactive Interaction Constraints* (IIC) as a novel framework to ground *channel-based* interaction (*à la* Reo) upon *constraints satisfaction*, interpreting the process of coordinating components as the execution of a constraints solver.

Next section further describes those technologies that can be actually installed and used nowadays—step §4 in Section 1.2.

2.3. Analysis of selected technologies

Table 3 overviews the *working technologies* we were able to somewhat successfully test, that is, only those technologies listed in Table 2 which successfully surpassed test §4 described in Section 1.2—a software artefact exists and is still working.

It is worth noting that, w.r.t. Table 2, a few technologies are not included in this section despite the corresponding software is available from the reference web page therein referenced. The reason is:

- Smrl requires ancient software to run—that is, an old version of Eclipse requiring in turn an ancient version of the Java runtime (1.4)
- CaSPiS [35] (or better, JCaSPiS, namely the Java-based implementation of CaSPiS) was not found anywhere—neither in the author personal pages, nor in their account profiles on Github, nor in the web pages of the SENSORIA project mentioned in the paper. Nevertheless, the IMC model and framework allegedly grounding its implementation is still accessible[9].

---

[9] `https://sourceforge.net/projects/imc-fi/`

Then we proceeded to download it looking for the CaSPiS code, without success. It is worth to be mentioned, anyway, that the IMC framework code appears to be broken, since compilation fails unless a restricted/deprecated Java API is used[10], and even in the case of instructing the compiler to allow for it[11] the attempt to run any part of the software failed without informative error messages—just generic Java exceptions.

- LeanProlog is not usable as a coordination technology as defined in Section 1.2: it is a Prolog engine with low-level mechanisms for handling multi-threading, and provides no API for general purpose coordination
- Session Java, as explicitly stated in its home page, requires an ancient version of the Java runtime to run, that is, 1.4
- Statelets is explicitly tagged as being in "pre-alpha" development stage, and, upon inspection, revealed to be only partially developed

TuCSoN. Although TuCSoN [16] appeared at COORDINATION in 1999, its roots date back to the first edition of the conference, as the $\mathcal{ACLT}$ model [6].

TuCSoN is a coordination model adopting LINDA as its core but extending it in several ways, such as by adopting nested tuples (expressed as first-order logic terms), adding primitives (i.e. *bulk* [59] and *uniform* [60]), and replacing tuple spaces with *tuple centres* [61] programmable in the ReSpecT language [62]. As such, the main driving concepts behind the TuCSoN model and technology are *(i)* first-order logic tuples and ReSpecT reactions to enable declarative expression of coordination policies, *(ii)* asynchronous communication and coordination primitives by default (however, synchronous versions are available, too) to enable full decoupling, *(iii)* programmable tuple spaces to enable full control over the coordination policies to be followed by the system at hand.

---

[10] A class uses a deprecated API, and another one requires breaking access restrictions.

[11] See `https://goo.gl/pdWCsx`.

Name	Last update	Health	Documentation	Source code	Build	Deployment
TuCSoN	2017	Actively developed	Available	Available	Successful	Successful
Moses	2017	Actively developed/maintained	Available	Unavailable	—	Successful
JErlang	2017	Discontinued	Poor	Available	Failed	—
IIC	2015	Discontinued	Poor	Available	Failed	Successful
Reo	2013	Actively developed	Available	Available	Successful	Partially successful
TripCom	2009	Discontinued	Partially available	Available	Successful	Successful
CiAN	2008	Discontinued	Available	Available	Successful	Successful
Piccola	2006	Discontinued	Available	Java only No Smalltalk	Successful	Successful
CRIME	2006	Discontinued	Unavailable	Unavailable	—	Successful
Klava	2004	Discontinued	Poor	Available	Successful	Successful
X-Klaim	2019	Actively developed	Available	Available	Successful	Successful
Limone	2004	Discontinued	Unavailable	Available	Failed	—
RepliKlaim	—[a]	—[a]	Unavailable	Available	Successful	Successful
Logic Fragments	2017	Actively developed	Available	Available	Successful	Successful

[a] There is no publicly available code repository, thus no information about latest commits.

Table 3: Overview of the working coordination technologies presented at COORDINATION. Column "Health" denotes the status of the software, for instance whether it is still actively developed, only in mainteinance mode, or actually discontinued, column "Build" is filled whenever source code is available and denotes whether build steps (i.e. compilation into binaries and dependencies resolution) were successful, column "Deployment" indicates whether the software has been successfully executed. It is worth to emphasise that LINC has been left out since it is part of commercial solutions sold by the Bag-Era company, thus no further inspection of the software was possible.

TuCSoN comes with a Java-based implementation providing *coordination as a service* [63] in the form of a Java library delivering an API and a middleware runtime, especially targeting distributed Java processes but open to rational agents implemented in tuProlog [64]. The TuCSoN middleware is *publicly available* from its home page[12], which provides both the binaries (a ready-to-use Java jar file) and a link to the *source code* repository. From there, also *documentation* pages are available, in the form of a usage guide and a few tutorials providing insights into specific features. Finally, a few related sub-projects are therein described too, such as TuCSoN4JADE [65] and TuCSoN4*Jason* [66], which are both Java libraries aimed at integrating TuCSoN with JADE [67] and *Jason* [68] agent runtimes, respectively, by wrapping TuCSoN services into a more convenient form which best suites those developers accustomed to programming in those platforms.

TuCSoN is still *actively* developed, as witnessed by the recently published extension to the ReSpecT language and toolchain [69]. Also, it is actively *exploited* as the infrastructural backbone for other projects (e.g., the smart home logic-based platform Home Manager [70]) and industrial applications (e.g., the Electronic Health Record solution described in [71]). Nevertheless, TuCSoN is the result of many years of active development by many different people with many different goals. Thus, despite some success stories, TuCSoN would require some substantial refactoring and refinement before it can become a truly commercially-viable product. The TuSoW project recently presented in [72] can be considered a notable effort in this direction, as it represent a rebooting attempt focusing on supporting modern mainstream technologies and platforms.

Moses. Moses [21] is the technology implementing the *Law Governed Interaction* (LGI) coordination model [73], which aims at controlling the interaction of agents interoperating on the Internet. In LGI, each agent interacts with the system by means of a *controller*, that is, a component exposing a fixed

---

[12] `http://tucson.unibo.it`

set of primitives allowing agents to exchange messages with other agents. The controller is in charge of intercepting invocations of primitives by interacting agents to check if they are allowed according to the *law* currently adopted by that controller.

Laws are shared declarative specifications dictating how the controller should react when it intercepts events of interest. Laws are expressed either in a Prolog-like language or as Java classes. Each controller has its own state which can be altered by reactions to events and can influence the effect of future reactions. Non-allowed activities are technically prohibited by the controller which takes care of aborting the forbidden operation—for instance, by not forwarding a message to the intended receiver if some conditions are met.

The main concepts around which LGI revolves therefore are *(i)* message passing for communication, *(ii)* reactive, declarative control laws for coordination, *(iii)* dedicated controllers to enact the coordination policies implemented.

The project home page[13] is well-organised and provides a number of resources focussed on Moses/LGI such as reference papers, manuals, tutorials, JavaDoc, examples. The page also provides an archive with the compiled versions of the *Moses middleware* suggesting that the project is *actively maintained and/or developed*, and representing another success story born within the COORDINATION series. We were able to *successfully* execute the executable: however, no source code is provided, and some portion of the web page, such as the JavaDoc, are not updated w.r.t. the current Moses implementation. Finally, Moses still bounds to *deprecated technologies* such as Java Applets, which may hinder its adoption.

JErlang. JErlang [38] is an extension of the Erlang language for concurrent and distributed programming featuring *joins* as the basic synchronisation construct—as borrowed from the JOIN-CALCULUS [39]. The web page mentioned in the pa-

---

[13] `http://www.moses.rutgers.edu/index.html`

per[14] is only accessible through the Wayback Machine[15]; by searching JErlang and the authors' names on the web, a GitHub repository with the same broken reference popped up[16], apparently tracking the development history of the JErlang technology. There, however, JErlang is described as an implementation of Erlang/OTP on the JVM. Also, another apparently very similar technology is therein referenced: Erjang. Later contact with one of the authors revealed that those projects are unrelated.

Anyway, JErlang installation and usage instructions are nowhere to be found, and, when trying to build the project through the provided Maven pom.xml file, the build fails due to many errors related to obsolete dependencies—which we were not able to fix. We feel then justified to declare the implementation as discontinued.

IIC. Interactive Interaction Constraints (IIC) [44] is a sort of "spin-off" of Reo introduced in 2013 [44]. The original approach of implementing Reo connectors as interaction constraints is extended to allow interaction to take place also *between rounds* of constraints satisfaction. This extends the expressive reach of IIC beyond Reo, and makes the whole process of constraints satisfaction *transactional* w.r.t. observable behaviour.

The IIC software is distributed as a Scala library providing an handy syntax which eases definition of Reo-like connectors. The Scala library *source code* is distributed by means of a GitHub repository[17] where the latest commit dates back to 2015. The library ships with a SBT configuration, allegedly supporting automatic building. Nevertheless, we were not able to reproduce the compilation process since the provided SBT configuration depends on an *ancient SBT version*. Therefore, we consider IIC a *no longer maintained* but *still usable* full-fledged coordination technology.

---

[14] `https://www.doc.ic.ac.uk/~susan/jerlang/`

[15] The web archive engine, working URL is: `https://web.archive.org/web/20160405003024/http://www.doc.ic.ac.uk:80/~susan/jerlang/`

[16] Second link in "See also" section at `https://github.com/jerlang/jerlang`

[17] `http://github.com/joseproenca/ip-constraints`

Reo. Reo was firstly introduced to the COORDINATION community in [23], its name in Greek letters ($P\epsilon\omega$). Similarly to the IWIM model, Reo adopts the paradigm of exogenous coordination of concurrent and possibly distributed software components. According to the Reo model, *components* are the entities to be coordinated, representing the computations to be performed, while *connectors* are the abstraction reifying coordination rules. The only assumption Reo makes about components is that they have a unique name and a well-defined interface in the form of a set of input ports and output ports. Conversely, connectors are composed by *nodes* and *channels*, or other connectors. A number of coordination schemes can be achieved by combining the different sorts of nodes and channels accordingly. This allows to formally specify *how*, *when*, and upon which conditions data may flow from the input to the output ports of components.

Reo is a fundamentally different model w.r.t. the tuple-based ones, as it fosters an *exogenous* form of coordination where the policies regulating interaction (hence coordination, too) are extracted from the interacting components and put into connectors. Its foundational abstractions are hence *(i)* connectors, composed by nodes and channels connecting I/O ports, and *(ii)* their compositionality, that is, the ability to preserve intended semantics when connectors are composed together to create more complex coordination policies.

Diverse research activities originated from Reo throughout the years, mostly aimed at *(i)* analysing the formal properties of both Reo connectors and the computational models behind Reo semantics (such as *constraints automata* [74]); and *(ii)* supporting web services orchestration [75], composition, and verification [76] by means of code generation and verification tools.

Several technologies are available from the Reo tools homepage[18], collectively branded as the Extensible Coordination Tools (ECT). They consist of various Eclipse IDE plugins, such as a graphical designer for Reo connectors, and a code generator which automatically converts the graphical description into

---

[18] `http://reo.project.cwi.nl/reo/wiki/Tools`

Java sources in which developers may inject applicative logic. Nevertheless, the generated code comes with no explicit support for distribution.

According to the home page, ECT are allegedly compatible with any Eclipse version starting from 3.6; while we were not able to reproduce its installation in that version (due to a dependency requiring an higher version of Eclipse), we succeeded in installing it on Eclipse version 2019-06 (the latest available to date), but the code generator appears *buggy and unstable*, hindering further testing, because of several non-informative error messages continuously appearing when trying to use the Reo model designer—which is a required step for code generation.

The ECT source code is available from a Google Code repository[19]—last commit dating back to 2013. In [77] a novel implementation is proposed, named Dreams, implemented in Scala and aimed at closing the gap between Reo and distributed systems. Nevertheless, its binary distribution seems *unavailable* and no documentation is provided describing how to compile or use it, thus we were not able to further test this novel Dreams framework.

TripCom. TripCom [30] is essentially a departure from the LINDA model where the tuple space abstraction is brought towards the Semantic Web vision [78] and web-based semantic interoperability in general. The former is achieved by employing the Resource Description Framework (RDF) – that is, a representation of semantic information as a triplet "subject-predicate-object" – as the tuple representation language, and by considering tuple spaces as RDF triplets containers. Also, LINDA primitives have been consequently re-thought under a semantics-oriented perspective—that is, by adopting an ad-hoc templating language enabling expression of semantic relationships. The latter is achieved by making triple spaces accessible on the web as SOAP-based web-services.

The implementation is hosted on a SourceForge repository[20] and it is apparently *discontinued*, provided that the last commit dates back to 2009, and the

---

[19] `https://code.google.com/archive/p/extensible-coordination-tools/source`

[20] `https://sourceforge.net/projects/tripcom`

home page lacks any sort of presentation or reference to publications or documentation. Nevertheless, the available source code appears well engineered and is *well documented*. It can be easily compiled into a `.war` file and then deployed on a Web Server (i.e. Apache Tomcat).

Once deployed, the web service is accessible via HTTP, making it virtually interoperable with any programming language and platform, and can be tested by means of a common web browser. Additionally, the service exposes a WSDL description of the API needed to use it, which implies that a client library (aka stub) may be automatically generated using standard tools for service-oriented architectures. Nevertheless, this WSDL description is the only form of documentation when it comes to actually interact with the web-service.

CiAN. Collaboration in Ad hoc Networks (CiAN) [32] is a Workflow Management System (WfMS) enabling users to schedule and execute their custom workflow over MANETs. It comes with a reference architecture and a middleware. The middleware keeps track of the workflow state in a distributed way, and takes into account routing of tasks' input/output data, on top of a dynamic network topology where nodes communication is likely to be opportunistic.

Workflows in CiAN are modelled as directed graphs whose vertices represent *tasks*, and edges represent the data-flow from a task to its successors: when a task is completed, a result value is transferred through its outgoing edges. Conditions may be specified within task definitions stating, for instance, whether a task should wait for all its inputs or just for one of them.

Users can encode their workflow descriptions via a XML-based language to be sent to an *initiator* singleton node, distributing the workflow to a number of *coordinator* nodes in charge of allocating tasks to the available *worker* nodes.

While the middleware is implemented in Java, tasks logic can be implemented virtually by means of any language since CiAN only assumes the application logic to interact with the middleware by means of the SOAP protocol, which provides great interoperability. Both the middleware's source code and its compiled version are distributed through CiAN website[21], together with detailed documentation and some runnable examples. The source code can be easily compiled, and both the obtained binaries and those publicly available can be run smoothly. The code is well documented and engineered. Nevertheless, the source code and documentation both date back to 2008: we therefore consider the project to be mature and usable, but no longer maintained.

Piccola. Piccola [20] is in its essence a *composition language.* It provides simple yet powerful abstractions: *forms* as immutable, prototype-like, key-value objects; *services* as functional forms which can be invoked and executed; *agents* as concurrent services; and *channels* as inter-agent communication facilities. Virtually *any* interaction mechanism can be built by properly composing these abstractions, such as shared variables, push and pull streams, message-passing, publish-subscribe, and so on.

Nevertheless, a limitation is due to the fact that not solely the coordination mechanisms are to be programmed with the Piccola language, but *also* the coordinated entities. There is thus no possibility of integration with mainstream programming languages, which is a severe limitation for adoption. Additionally, even if Piccola comes with networking capabilities *virtually* enabling deployment to a distributed setting, there is no middleware facility available and no opportunity for integration with others is given, which is another factor likely to hinder Piccola adoption within the scope of distributed programming and coordination.

Piccola home page[22] is still available and collects a number of useful resources such as documentation pages and implementation. This comes in two flavours: JPiccola, based on Java, which reached version 3.7, and SPiccola, based on Smalltalk, which reached version 0.7. Source code is provided for the Java implementation only, which *correctly compiles and executes.*

Nevertheless, the project appears to be *discontinued*, given that the last commit on the source repository dates back to 2006.

---

[21] `http://mobilab.cse.wustl.edu/Projects/CiAN/Software/Software.shtml`

[22] `http://scg.unibe.ch/research/piccola`

CRIME. CRIME adheres to the *Fact Spaces* model, a variant of LINDA which absorbs transient federation of tuple spaces from Lime [57] for implementing mobile *fact spaces*: tuple spaces where tuples are logic facts and each tuple space is indeed a logic theory. Federated fact spaces are therefore seen as distributed knowledge bases.

In this sense, CRIME has some similarities with TuCSoN, which exploits first-order logic tuples both as the communication items and as the coordination laws. In this context, LINDA `out` and `in` primitives collapse into logic facts assertions and retractions, respectively.

Suspensive semantics is not regarded as being essential within the scope of the *Fact Spaces* model, since the focus is about programming fact spaces to react to information insertion/removal (or appearance/disappearance in case of transient federation). Accordingly, users can register arbitrary logic rules by means of a Prolog-like syntax. The head of such rules represent propositions which may be proved true (activated) or unknown (deactivated) given the current knowledge base by evaluating the body of the rule. Users can then plug arbitrary application logic reacting to (de)activation of these rules.

Implementation of CRIME is available on the project home page[23] and consists of an archive shipping pre-compiled Java classes with no attached source code. The software is apparently *no longer maintained*: the web page has been updated last in 2010, and the archive dates back to 2006. Nevertheless, the archive provides a number of example applications which have been tested and are still *correctly working.* No support is provided to application deployment and *no documentation* has been found describing how to deploy CRIME to an actual production environment.

KLAIM-$\star$. With notation KLAIM-$\star$ we refer to the family of models and technologies stemming from KLAIM [26] – such as O'Klaim [25] and MoMi [79] – which nowadays evolved into the X-Klaim/Klava framework [80, 81]. X-Klaim

---

[23] `http://soft.vub.ac.be/amop/crime/introduction`

consists of a domain-specific language and its compiler, which produces Java code by accepting X-Klaim sources as input. The produced code exploits the Klava library in turn, that is, the Java library implementing the middleware corresponding to the KLAIM model.

The overall framework explicitly targets code mobility, thus allowing both processes and data to migrate across a network. To do so, X-Klaim and Klava provide a first-class abstraction known as *locality*. Localities are of two sorts: either *physical*, such as network *nodes* identifiers, or *logical*, such as symbolic references to network nodes having a local semantics. Each locality hosts its own tuple space, and the processes therein interacting. The LINDA primitives supported by Klava are always explicitly or implicitly related to the tuple space hosted on a specific locality. Furthermore, processes are provided with primitives enabling them to migrate from a locality to another in a *strong* manner, that is, along with their execution state.

Summing up, KLAIM is a tuple-based coordination model extending the expressive reach of LINDA-like models by explictly considering mobile environments. As such, its most peculiar concept is that of locality and the associated machinery to handle process-location association in presence of mobility.

Both X-Klaim and Klava are distributed by means of the KLAIM Project home page[24], providing well detailed *documentation*. For what concerns X-Klaim, its C++ *source code* (dating back to 2004, date of the last edit, visible right below the title) is *publicly available* along with a self-configuring script meant to ease compilation. Nevertheless, we were not able to reproduce the compilation process on modern Linux distributions, seemingly due to some missing (and undocumented) dependency. No clues about how to fix the self-configuration process when it fails is provided, neither we were able to find some sort of documentation explicitly enumerating the compilation dependencies. However, for the sake of completeness, it is worth to be mentioned that

---

[24] `http://music.dsi.unifi.it/klaim.html`

X-Klaim has been recently rebooted[25] as an Xtext/Ecplise-based technology, which is currently actively maintained and successfully deployable. There, the X-Klaim compiler has been actually replaced by a code generation utility leveraging on the Xtext toolkit[26], which also brings a number of Eclipse-related utilities.

Conversely, the Klava library – actually implementing the coordination middleware – is distributed as a single `.jar` file containing both Java sources and the binaries. The `.jar` file dates back to 2004 likewise for X-Klaim, so it is apparently no longer developed, but further testing showed how the Klava library is still *functioning*, since it is self-contained and targets Java versions 1.4+.

Limone. Limone [27] is a model and middleware meant to improve scalability and security in Lime [57] through access control, and explicitly targeting distributed mobile systems and, in particular, agents roaming across ad-hoc networks built on top of opportunistically interconnected mobile devices.

Once two or more devices enter within their respective communication range and thus establish a connection, the agents running on top of them are (potentially) enabled to interact by means of transient sharing of their own tuple spaces. But, for some agents to be actually able to communicate, Limone states they should specify their *engagement* policies. An agent A's engagement policy determines which agents are allowed to interact with it and to which extent, that is, which primitives are allowed to be invoked. Agents satisfying the policy are registered within A's *acquaintance* list. So, each agent only has to care about its acquaintance list, thus reducing the bandwidth requirements for the middleware.

A reactive programming mechanism completes the picture, enabling agents to inform their peer about their interest in tuples matching a given template, in order to be informed when such tuples becomes available.

---

[25] `https://github.com/LorenzoBettini/xKlaim`

[26] `https://www.eclipse.org/Xtext/`

The Limone technology is distributed by means of the project web page[27] in the form of a compressed archive containing the Java source code (dated back in 2004) and a `Makefile` for automatic build. Nevertheless, the code strictly requires to be compiled against a Java version *prior to 1.5*, and modern Java compilers do not support such an ancient version[28] . For these reasons, we could not proceed to further test the technology and we consider it to be no longer maintained *nor actually usable*.

RepliKlaim. RepliKlaim [47] is a variant of Klaim [26] introducing first-class abstractions and mechanisms to deal with *data locality* and *consistency*, so as to give programmers the ability to explicitly account for and tackle these aspects when developing parallel computing applications. Specifically, the idea is to let the programmer specify and *coordinate replication of data*, and operate on replicas with a configurable level of consistency. This enables the programmer to adapt data distribution and locality to the needs of the application at hand, especially with the goal of improving *performance* in terms of concurrency level and data access speed—in spite of latencies due to distribution.

Most of the abstractions and mechanisms, as well as syntax elements and semantics, of RepliKlaim are exactly as in Klaim, such as data repositories, processes, locations, and many actions. When due, actions are extended to explicitly deal with replication aspects, such as in the case of an `out` primitive putting multiple copies of the same tuple in multiple localities, or an `in` primitive removing all replicas from all locations at once. Also, various *degrees of consistency* among replicas in the same or different locations are achieved depending on whether primitives are synchronous (namely, atomically executed) or asynchronous.

There exists a *prototype* implementation of RepliKlaim on top of Klava, the Java implementation of Klaim, available for direct download from a URL[29] given

---

[27] `http://mobilab.cse.wustl.edu/projects/limone`

[28] As stated here: `https://docs.oracle.com/javase/9/tools/javac.htm#JSWOR627`

[29] `http://sysma.imtlucca.it/wp-content/uploads/2015/03/RepliKlaim-test-examples.`

in its companion paper [47]. From there, a .rar archive is provided, containing a version of Klava and the *source* files implementing RepliKlaim, which can be easily compiled and run successfully.

Nevertheless, as stated in the paper describing RepliKlaim, its implementation currently relies on encoding its model in the standard Klaim model, thus, on the practical side the code provided only features examples about how to translate RepliKlaim primitives into Klava. *No higher-level API* directly providing to developers the replica-oriented operations of RepliKlaim is provided. In other words, there exists no RepliKlaim Java library which can be imported to other Java projects in order to exploit its provided coordination services.

Logic Fragments. Logic Fragments, also called Logic-Based Chemical Coordination Model (LFCM) [82] is a *chemical-based* and programmable coordination model inspired to Sapere [49, 83], itself a coordination model for multi-agent pervasive systems inspired to natural chemical reactions. Sapere is based on four main concepts: *Live Semantic Annotations* (LSAs), *LSA Tuple Space*, *agents* and *eco-laws*. LSAs are tuples of types $(name, value)$ used to store applications data. LSAs belonging to a computing node are stored in a shared container named LSA Tuple Space. Each LSA is associated with an agent, such as sensors, services, or general applications that want to interact with the LSA space—e.g. injecting or retrieving LSAs from the LSA space. Inside the shared container, tuples react in a virtual chemical way by using a predefined set of coordination rules named *eco-laws*, which can *(i)* instantiate relationships among LSAs (*Bonding* eco-law), *(ii)* aggregate them (*Aggregate* eco-law), *(iii)* delete them (*Decay* eco-law), and *(iv)* spread them across remote LSA Tuples Spaces (*Spreading* eco-law). When a tuple is modified by an *eco-law*, its relative agent is notified. The implementation of the Sapere middleware allowed developing several kinds of real distributed self-adaptive and self-organising applications [49].

`rar`

Logic Fragments extends SAPERE and defines a coordination model based on *logic inference* [84]. Logic Fragments are combinations of logic programs defining interactions among agents distributed over the nodes of the system. Logic Fragments allows agents to inject logic fragments, a new type of LSA, into the shared space. An additional eco-law (the *Logic fragment* eco-law) interprets those fragments based on the current tuples in the tuple space (including neighbouring ones). Those fragments actually define on-the-fly ad-hoc chemical reactions that apply on matching data tuples present in the system, removing tuples and producing new tuples, possibly producing also new logic fragments. The model is defined independently of any specific logic, an actual instantiation and implementation of the model can use its own logic(s). The corresponding middleware for two-valued logic is publicly available as open source project[30].

Logic Fragments supports various types of logics, ranging from classical up to many-valued paraconsistent ones. The logical formalisation makes it possible to express coordination in a rigorous and predictable way, both at design-time and run-time, as well as injection of new eco-laws under the form of logic formulae.

Extensions of both SAPERE and Logic Fragments as prototyping platforms for large-scale experiments are available. TheOne-SAPERE is a prototyping tool [85] that integrates the SAPERE middleware within The Opportunistic Network Environment (The One) simulator [86], allowing to prototype and validate applications with realistic scenarios before deploying them. Indeed, it allows on the one hand to simulate a large number of computational nodes movements and their communications, placing them in various configurations allowing stochastic evaluation of parameters. On the other hand, each node is equipped with the actual SAPERE middleware (actual code), allowing to execute from within the simulation actual spatial system services (gradient, spreading, evaporation, etc.), thus providing actual results relating to spatial system services behaviour. Following the above idea, TheOne-LFCM [84] is a prototyping platform where the actual Logic Fragment middleware runs in each simulated node.

---

[30] `https://bitbucket.org/houssembenmahfoudh/theonesapere/src`

Codes for various SAPERE variants, Logic Fragments and the two prototyping platforms can be retrieved from open source repositories all reachable from `https://www.unige.ch/cui/cas/publications/projects-output/`. They all have some documentation available, either in the form of publications, readme files, or "hands-on" tutorials, and, Logic Fragments specifically, does successfully compile and run.

3. Insights

In this section we aim at providing further insights about the technologies described in Subsection 2.3, especially to understand *(i)* whether they stem from a common archetypal coordination framework (Figure 5), *(ii)* their relationships in terms of the features they provide (Figure 6), and *(iii)* which goal mostly motivated their development and which application scenario they mostly target (Figure 7).

A family tree. Figure 5 depicts a sort of "family tree" of the selected coordination technologies, emphasising how they stem from a few archetypal coordination models/languages, and how they are built on top of each other. It

Klaim reference (maybe report Klaim instead of Klava into the picture?)

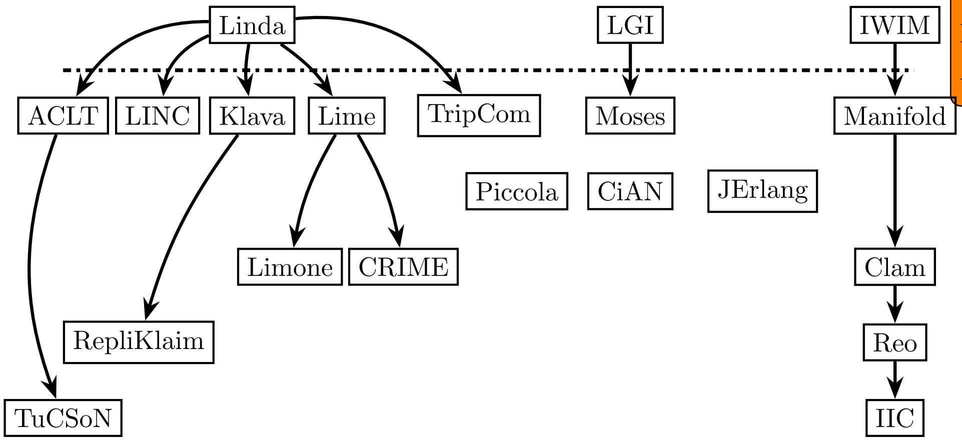

Figure 5: Lines of evolution of selected technologies (below the dashed line), as stemming from a few archetypal coordination model (above the dashed line).

makes thus apparent how most of the technologies still available stem from two archetypal models: LINDA [46] and IWIM [1]. Nevertheless, whereas in the case of LINDA many heterogeneous extensions have been proposed throughout the years, focussing on different features and thus diverging from LINDA in many diverse ways, the evolution of the IWIM model appears much more homogeneous, featuring descendants which "linearly" extend their ancestors' features. Summing up, from LINDA stem the TuCSoN family, the Klaim [26] family (including Klava and RepliKlaim), the LIME [87] family (with Limone and CRIME), besides the lone runners LINC and TripCom, whereas from the IWIM root stems the Reo family—completed by Manifold, Clam, and the latest extension IIC.

Apart from these two big family trees, we have the LGI model, along with its implementation, Moses, and a small group of "lone runners" with unique features: Piccola, CiAN, and JErlang. While the former inspired some features of technologies stemming from other models – for instance, its *programmable laws* inspired essentially any other technology or model having *reactive rules* of some sort, such as LINC –, the latter remained mostly confined to itself.

It is interesting to notice how "the IWIM family" and "the LINDA family" remained well-isolated one from each other over all these years. Whereas this can be easily attributed to the fundamental difference in the approach to coordination they have (data-driven vs. control-driven, as also emphasised in Figure 6 on page 35) it seems odd that nobody tried to somewhat integrate these two extremely successful coordination models, in an attempt to improve the state of art by cherry-picking a few features from both to create a novel, *hybrid* coordination model [88], with "the best of two worlds". To some extent, the TuCSoN model, along with its coordination language, ReSpecT, pursues this path: ReSpecT in fact can be regarded as a data-driven model because coordination is based on availability of tuples, as in LINDA, but, at the same time, coordination policies are enforced by declarative specifications which *control* the way in which the coordination medium behaves, thus, ultimately, how the coordinated components interact—as typical for control-driven models like IWIM.

The path toward integration could be the key in further perfecting and

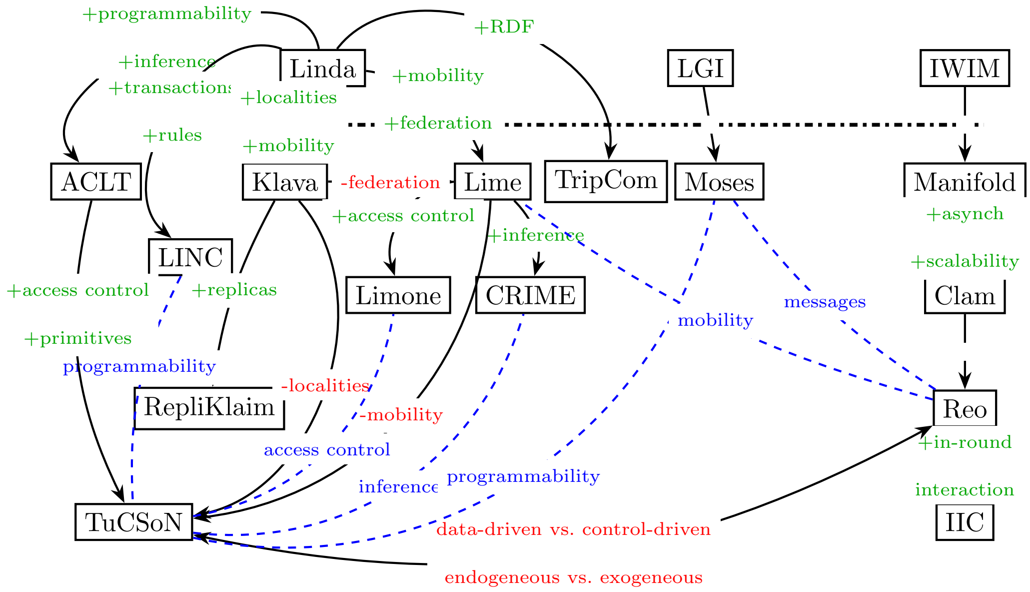

Figure 6: Main differences (plus and minus signs) and similarities (dashed lines) amongst selected technologies. Arrows indicate what it takes (plus, add something; minus, remove something) to go from one technology (the source) to another (the destination).

improving coordination models and languages, by complementing data-driven models elegance and flexibility with control-driven models fine-grained control and predictability.

Families marriage. Figure 6 enriches the family tree just described with relationships indicating *differences* (plus and minus signs on labels) and *similarities* (dashed lines) in features provided—notice that w.r.t. Figure 5 Piccola, CiAN, and JErlang have been removed because they are so unique that no clear relationship may be found with other technologies. As already mentioned for Figure 5, Linda has been taken as the common ground for many technologies which are instead very heterogeneous in the aim pursued: if *ACLT*, TuCSoN, and LINC have a Linda core enriched with many other features (such as programmability, transactionality, and novel primitives), the Klaim family and the

LIME one diverge more, by changing the way in which primitives behave (as in the case of localities in Klaim), or the way in which the interacting processes see each others' tuple spaces (as for LIME transient federation).

Nevertheless, technologies which may appear as being far apart from each other have interesting similarities, as in the case of the interaction rules of LGI, thus Moses, which strongly resemble $\mathcal{ACLT}$ and TuCSoN reactions, or the fact that both the Reo family and Moses are based on message passing. Or, the fact that both CRIME and TuCSoN rely on logic tuples so as to leverage on the inference capabilities of interacting agents, while Reo and both LIME and Klaim take into account mobility of processes and coordination abstractions (tuple spaces vs. channels) as a first-class citizen.

It is worth emphasising here that Figure 6 highlights the features to which more attention has been devoted throughout the years: programmability, access control, and mobility. These features, possibly extended with scalability and inference capabilities, are crucial for widening applicability of coordination technologies to real-world scenarios. For instance, the Internet of Things (IoT) [89] – along with its variants Web of Things [90] and Internet of Intelligent Things [91] – is a very good fit for testing coordination technologies, and requires precisely the aforementioned features.

Goals & preferred scenarios. Finally, Figure 7 relates the selected technologies with the main aim pursued which motivates their extension in a particular direction, along with the applications scenario they best target.

From the description of the selected technologies we gathered, two are the main goals motivating their evolution: *(i)* providing *flexibility* so as to deal with the majority of heterogeneous application scenarios possible, and *(ii)* focussing on first-class abstractions for better supporting *space-awareness* of both the coordination abstractions and the interacting processes.

In fact, TuCSoN / $\mathcal{ACLT}$, LINC, and Moses all provide means to somewhat *program* the coordinative behaviour of the coordination medium, thus aim at making it configurable, adaptable, malleable, even at run-time, and/or provide additional coordination primitives to expand the expressive reach of the coordination technology. The Klaim family, the Reo family, and the Lime family instead, are geared toward some forms of *space-awareness*, be it by promoting mobility or by providing location-sensitive primitives. Reo, for instance, has an explicit notion of location (as the logical or physical place where a component executes) that is also explicitly managed but language primitives, such as `_move` which enables relocation of channel ends to a different location.

Besides these, two more main goals can be devised, peculiar to specific technologies: *(iii)* supporting *humans-in-the-loop*, in the case of CiAN, and *(iv)* provide a *semantic* representation of data items, in the case of TripCom.

About the application scenarios explicitly declared as of particular interest for the technology, the most prominent one is *service composition*, which is especially interesting for Piccola, JErlang, the Reo family, the Klaim family, and TripCom—besides being naturally applicable to all other technologies too. Then, whereas technologies such as LINC and the Lime family are mainly tai-

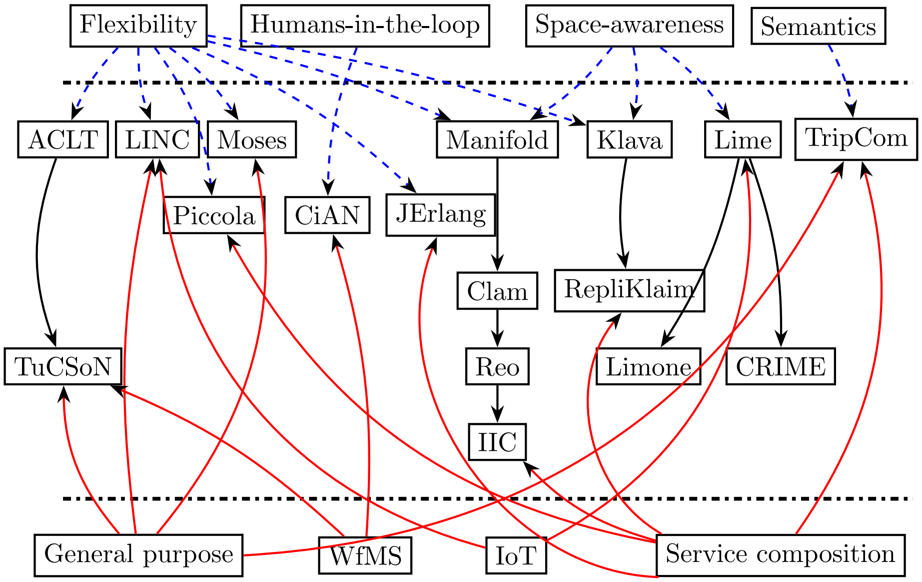

Figure 7: Selected technologies per main *goal* pursued (top, dashed arrows) and preferred *application scenario* (bottom, solid arrows).

lored to the *IoT landscape*, being meant to cope with the requirements posed by small, possibly portable, possibly embedded devices with low resources, *Workflow Management* (WfMS) is peculiar to CiAN, while also considered by TuCSoN [92]. Besides these application scenarios, there are many technologies without a specific focus, although they have been applied to many different ones, such as TuCSoN itself, LINC, Moses, and TripCom: these have been associated with the generic "General purpose" scenario.

The goals and application scenarios just highlighted strengthen our previous consideration that the IoT could be the "killer-app" for coordination technologies. In fact, flexibility (there including programmability and configurability), space-awareness (there including mobility and location-awareness), and semantics (there including interoperability of data representation formats) are all necessary ingredients for any non-trivial IoT deployment: the former helps in dealing with *uncertainty* and *unpredictability* typical of the IoT scenarios, the latter is required for building open IoT systems, and some form of space-awareness is a common feature of many IoT deployments, from retail to industry 4.0. Also, the fact that service composition has been already thoroughly explored within COORDINATION is a great advantage and the perfect starting point for tackling IoT challenges: both the IoT and the Web of Things vision foster a world where connected objects provide and consume services, which can be composed in increasingly high-level ones.

Along this line, many recent contributions started to recognise the need to adopt coordination models and languages as a means to effectively orchestrate the increasingly complex network of interactions amongst IoT components in distributed deployments—as encouraged by the movement from a CLoud-centric IoT to an Edge-based IoT: in [93] event-condition-action rules are used in a publish-subscribe setting to coordination services based on the events they generate; in [94] FIPA protocols are offered as ready to use coordination means, alongside with a topic-based blackboard mode (to achieve reference uncoupling) as well as event-drive coordination w.r.t. the cyberphysical part of the IoT system; in [95] the dataflow programming model is instead used the coordination model governing interactions between components in a Fog computing deployment.

4. Coordination technologies outside COORDINATION

In this section we want to position COORDINATION w.r.t. other *related* conferences while still retaining focus on technologies. Term "related" reflects the following inclusion criteria: we selected those conferences and workshops where the set of *most active authors* has a reasonable intersection with the most active authors of COORDINATION. Such sets have been identified thanks to the `dblp` portal[31]. As a result, four communities have been identified, thus considered: SAC, SASO, FOCLASA, and ISOLA. We then filtered out papers which do not explicitly contain word "coordination" either in the title or abstract, and finally manually inspected the remaining ones looking for technologies explicitly dealing with coordination.

The *Symposium on Applied Computing* (SAC), for instance, has a strong relationship with COORDINATION, as it hosted a specific track dedicated to "Coordination Models, Languages, and Applications" until its 30^{th} edition, in 2015. Then, it converged into the "Programming Languages" track. The international conference on *Self-Adaptive and Self-Organising Systems* (SASO) has often seen participation of several well known authors from the COORDINATION community, mostly because self-organisation is often built on top of handling interactions among components. Nevertheless, technological contributions are rare as many works in SASO are mostly concerned with simulation of emergent and adaptive behaviours resulting from rather simple, but numerous, interactions, rather than with designing coordination middleware. Finally, both the international workshop on *Foundations of Coordination Languages and Self-Adaptative Systems* (FOCLASA) and the *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation* (ISOLA)

---

[31] `https://dblp.uni-trier.de/`, search for "COORDINATION" than inspect the bar on the right, where authors and venues lie.

turned out to be strongly intertwined with the COORDINATION community, although the focus of both is much more on the theoretical and formal side of coordination theories, languages, and models, hence technological contribution are almost absent here.

In the following, we analyse each venue separately, briefly reporting on the technological contributions found, and emphasising relationships with COORDINATION whenever possible.

SAC. Many SAC papers have direct relationships with works presented at COORDINATION, hence already described in Section 2. For instance:

- the ReSpecT language for programming $\mathcal{ACLT}$ *tuple centres* is introduced and proved Turing-complete [96]. $\mathcal{ACLT}$ will later become TuCSoN, and ReSpecT would become its coordination language
- the TuCSoN model and technology is introduced [97], and its relationship with the tuple centre notion and ReSpecT are discussed
- *MARS-X* is presented as a *programmable* coordination architecture for Internet applications based on mobile agents [98]. *MARS-X* extends *MARS* [99] by letting agents coordinate through programmable XML dataspaces, accessed by agents in a Linda-like fashion. The programmable nature of MARS dataspaces and the focus on Internet applications are inspired by TuCSoN and ReSpecT, presented just one year before
- the DICE framework architecture is described and a report on its implementation is given [100]. DICE (Distributed Constraint Environment) is a framework for the construction of distributed constraint solvers from software components. The framework is implemented using the *Manifold* coordination language, and delivers coordination services to these components. The coordination services implement existing protocols for constraint propagation, termination detection, and splitting of constraint satisfaction problems

- in [101], the authors show how a variety of distributed tuple structures for *field-based coordination* can be easily programmed in the TOTA [102] middleware. Several examples clarify the approach, and performance measures are discussed to evaluate its effectiveness. TOTA inspired the whole research theme of field-based coordination, as well as the SAPERE approach, hence Logic Fragments in turn

- in [103], the authors discuss a framework for self-organising coordination: coordination media spread over the network are in charge of managing interactions with each other and with agents solely according to local criteria, making global properties of the resulting system appear by emergence. The authors strongly leverage on the TuCSoN/ReSpecT coordination infrastructure, used as a general purpose coordination platform for enacting self-organising coordination. The examples of *chemical-like coordination* here reported are precursors of the whole biochemical coordination research theme culminating in SAPERE

- in [104], the authors introduce a *semantic-oriented* extension of the tuple space model based on OWL[32] and Description Logics. An incarnation of this model is proposed using the TuCSoN/ReSpecT coordination model and infrastructure

- in [105], a logic based language for programming *coordination artefacts* is presented. The language is based on reactive rules to define coordination laws and policies. A prototype built on top of CArtAgO [106], and relying on the tuProlog Prolog engine [64] is also presented, where different coordination paradigms realised upon the language are shown. The work relates to the theme of chemical-like coordination above described, and is heavy influenced by the work on TuCSoN and ReSpecT by the same research group

---

[32] `https://www.w3.org/TR/owl-ref`

- in [107], the authors introduce the concept of a *pervasive ecosystem* (at the basis of the SAPERE approach, too), and present a coordination approach grounded upon it, which revolves around *(i)* the notion of a distributed and dynamic space of "live semantic annotations" (wrapping data, knowledge, and activities of humans, devices, and services) and *(ii)* a set of chemical-resembling coordination rules that are applied to such annotations semantically (both concepts closely resembling SAPERE). As an application example, a simulated scenario of crowd steering in an exhibition centre is presented, leveraging on the Alchemist simulator [108]. A number of contributions which have been proposed in the next years leverage on the Alchemist simulator as well – such as [109, 110] –, proving it as a solid solution for simulating coordination mechanisms
- the Dreams framework is introduced aimed at further integrating Reo with distributed systems [111]. In fact, in Reo, data is exchanged via synchronous atomic actions, whereas distributed systems are typically asynchronous and assume that messages can be delayed or get lost—as Dreams does
- in [112], the authors present a Peer to Peer (P2P) agent coordination framework for the exchange of Electronic Health Records between health organisations that comply with the existing interoperability standards as proposed by the Integrating Healthcare Enterprise. To model the interactions among communities, the framework uses a tuple centre and semantic web technologies, both implemented on an extension of the TuCSoN/ReSpecT infrastructure
- IMCREOtools is presented as a toolkit supporting Interactive Markov chains (IMC) [113], where IMC is a stochastic compositional model of concurrency which the authors argue may be effectively used to serve as a compositional semantic model for *Stochastic Reo* [114]

The above list already suffices in defining SAC as a premiere venue for research in coordination models and languages, second to COORDINATION itself only. As for COORDINATION, some technologies are either discontinued or no longer accessible, such as MARS, DICE, and TOTA, or became part of other technologies still available nowadays, such as for $\mathcal{ACLT}$, all the Reo different tools, and the whole chemical-inspired coordination research thread.

Besides the above technologies strictly related to COORDINATION products, the Coordination Models and Languages track of SAC generated many other technologies throughout the years. In [115], the *Tuple Channel* abstraction is injected in C, Haskell, and Smalltalk, chosen as examples of three different programming paradigms – imperative, declarative, and object oriented, respectively – to show the versatility of tuple-based coordination as well as its orthogonality w.r.t. the programming paradigm. In [116], a coordination model is presented aimed at deriving efficient implementations on top of *MPI*[33] for C using mixed task and data parallelism. The model provides a specification language in which the programmer defines the available degree of parallelism and a coordination language to define how the potential parallelism is exploited for a specific implementation. The transformation of a specification program into a coordination program is performed in well-defined steps, therefore can be automated, with the benefit of a correct output program by construction. In [117], the notion of *XML Space* is introduced as a tuple space where tuples are XML documents and templates are query languages addressing XML, such as XPath or XQL. The authors then survey three implementations supporting XML Spaces, namely, the aforementioned *MARS-X* [98], *WebSpaces* [118], and *XMIDDLE* [119]. In [120], a model enabling *multi-paradigm coordination* between distributed and mobile software agents is presented, along with a reference software architecture, ActiWare, for which a Java-based prototype implementation is also described. In [121], the authors present a coordination model which combines logic-based reasoning with a reliable semantic subscription mechanism. They discuss its practical applicability based on execution of

---

[33] `https://computing.llnl.gov/tutorials/mpi/`

performance benchmarks of a prototype implementation called SNES. In [122], the authors propose a framework aimed at supporting development of urban-wide applications, leveraging the AmbientTalk [34] language and toolkit and the TOTAM tuple space implementation [123]. In [124], the SmallSpaces tuple space implementation is described: it focuses on providing rights management to control access to tuples within the scope of applications where all the different flows of data need to be kept separate for confidentiality reasons.

Besides the relevance of coordination models in general, and especially the interest in the concept of tuple-based coordination, an aspect worth emphasising is the constant presence of proposals for coordination technologies throughout the years, along with the application to different business domains. This clearly indicates that research in coordination models and languages is considered a staple across application domains.

SASO. In the SASO series there is not a dedicated track on coordination models and languages, hence we can expect few contributions fostering new coordination models, languages, or technologies. Indeed, SASO is much more concerned with the two deeply related aspects of *self-organisation* (by emergence) and *adaptation*, hence many works are about simulation of systems or languages guaranteeing some global properties by construction, or again focus on the so-called *"local-to-global"* issue [125]. Nevertheless, being self-organising systems often architected as distributed systems in which a multitude of components interact, coordination is of paramount importance. This is well exemplified by the following contributions:

- an architecture and actual system for self-organising coordination of an ensemble of ground and air robots [126] is built upon the JADEX platform [127] for BDI agent development [128]. The proposed architecture features a *blackboard agent* which actually plays the role of a tuple space collecting task assignments and dispatching those assignments to either

---

[34] http://soft.vub.ac.be/amop/

an Individual Coordination Agent, in case the task does not require cooperation amongst agents, or a Swarm Coordination Agent (SCA) in case no agent is able to solve the task individually. The blackboard agent also coordinates cooperating agents when due, as instructed by the SCA. No explicit and dedicated coordination technology is used nor proposed here, but the blackboard agent clearly witness the need for one—and the naive attempt to provide it

- in [129] it is presented and evaluated an architecture and prototype for the coordination of multiple *autonomic managers* responsible for running the MAPE-K loop in charge of optimising Cloud resources usage. There, a *message broker* enables interaction and knowledge sharing amongst decentralised autonomic managers. A detailed event-based protocol is described, so as to make explicit the coordination actions corresponding to the admissible interactions. Again, no explicit coordination model nor technology is exploited, however, the unambiguous description of the protocol is itself a (implicit) coordination model dictating how to govern dependencies amongst the distributed autonomic managers

- in [130] the $\mathcal{M}$olecules of $\mathcal{K}$nowledge coordination model for the *self-organisation of information* items in a distributed network is presented in the form of a prototype implemented on top of TuCSoN and ReSpecT, as applied to the application domain of citizen journalism. The model was conceived within the same European project behind the SAPERE model, hence it shares many characteristics with the whole field of biochemical coordination, complemented with an original application of principles stemming from *observation-based coordination* (e.g. stigmergy) [131, 132]—in particular, from behavioural implicit communication theory [133]

A few other contributions, in particular [134, 135, 136], are all either preparatory to SAPERE or a byproduct of it, hence share the same distinguishing characteristics described while also describing, for instance, Logic Fragments (Subsection 2.3). Summing up, we can say that SASO, besides being a venue where new coordination models and languages are proposed, perhaps specifically geared toward self-organisation and adaptation, it is also a community which "stress-tests" existing coordination models and languages in highly decentralised and dynamic scenarios, as those fostering emergent phenomena typically here. It could be interesting, thus, to continue monitoring SASO production of coordination-related papers, especially technological ones, as a means to asses to which extent coordination impacts research on self-organisation and self-adaptation—until now, the impact has been pretty high, as exemplified by the exemplary papers overviewed above.

FOCLASA & ISOLA. The latest two conferences we found a reasonable overlap with in the pool of COORDINATION authors are FOCLASA and ISOLA. We group them together for two reasons: first, they both have a more theoretical focus, often emphasising aspects such as *minimality* and *expressiveness* reach of core calculi (for the former) and formal, automated *verifiability* of programs' correctness (for the latter); second, for such a motivation technological contributions in the sense of actual coordination middleware or libraries are rare—on the contrary, simulation and model checking frameworks do abound.

The few works worth describing as they preserve the spirit of our survey are:

- in [137] the authors present an extension to the jRESP Java-based runtime environment for running distributed programs written in the *SCEL language* [138], augmented with the notion of policy as stemming from the FACPL model for access control [139]. jRESP has a web page[35] and associated source code repository[36] still reachable although discontinued (last access in 2016). In jRESP, the means for sharing data amongst interacting agents is actually a tuple space, with addressing and discovery mechanisms similar to those employed in the Klaim family of models—SCEL was in fact largely developed by the same research group

---

[35] `http://jresp.sourceforge.net/`

[36] `https://sourceforge.net/projects/jresp/?source=navbar`

- in [140] an implementation in the Go language of the concept of *attribute-based interaction* is presented. The implementation is actually agnostic to the underlying mediation infrastructure, in fact, the authors evaluate their Go API with three different infrastructures for investigating the best efficiency trade-off. Regardless of the infrastructure, the kind of attribute-based interaction fostered in the paper is based on *message passing* where communications are dispatched in a sort of *publish-subscribe* paradigm where subscriptions change automatically and dynamically based on environmental properties and current context of interacting components. The authors also developed an Eclipse plugin for assisting programmers. All the software is available starting from the project webpage[37]

5. Coordination technologies in Industry

Based on the information gathered in the survey, there is only one coordination technology among those described in Section 2 which is actively used in industrial practice: *LINC*, as part of the Bag-Era company suite of solutions for orchestrating IoT services and handle consistency along data chains. Bag-Era is a young startup company (created mid-2016), founded by several researchers who used to work with coordination languages for some time (20 years for the eldest), that provides coordination solutions to improve industrial processes. Apart from this exception, the surveyed papers and the technologies web pages give no reason to believe some of them are actually used in industrial products.

Nevertheless, if we consider not the actual COORDINATION technologies (the software) but the goals, abstractions, and mechanisms behind them (such as ordering actions or orchestrating data flows, tuple spaces or message channels, suspensive semantics or reactive notification), then we find many more coordination technologies embedded in modern software products in the field of, for instance, *service-oriented computing*—mostly as enabler of service orchestra-

---

[37] `https://giulio-garbi.github.io/goat/`

tion. In particular, despite the heterogeneity of implementations, architectures, intended purpose and intended value added of the specific software product, a "coordination core" can be found in two categories of products:

- *in-memory data grids* (IMDG), that is, in-memory, usually distributed data storage layers enabling distributed applications to quickly, reliably, and consistently access shared data and communicate without the need to rely on direct message passing—ultimately enabling decoupling in space, time, and reference
- *Internet of Things* (IoT) platforms, ranging from full-fledged software suites providing basic interoperability and discovery services as well as application programming API, to more specific solutions targeting a single or a narrow spectrum of requirements and desiderata

In the following we mention a few technologies for each category, with the goal of clarifying the relationship with the concepts and mechanisms proposed in the various COORDINATION papers surveyed.

IMDG. Amongst in-memory data grids solutions, *GigaSpaces*[38] shines as it explicitly relies on an implementation of the *JavaSpaces specification* [141], one of the earliest implementation of the tuple space concept along with the LINDA model. GigaSpaces is actually a full-fledged application server which leverages a space-based architecture to enable low-latency and reliable communication between so-called Processing Units (a way to partition applications independent components, similarly to microservices). The core of the API is hence meant to provide access to the shared tuple space, upon which many high level middleware functionalities are realised, such a messaging, caching, parallel processing, reactive programming, publish-subscribe communication.

Another software explicitly mentioning tuples populating shared data spaces is TIBCO *ActiveSpaces*[39]: there, however, the notion of space is a bit different

---

[38] `https://www.gigaspaces.com/`

[39] `https://www.tibco.com/it/products/tibco-activespaces`

from a traditional tuple space, as spaces are dynamically composed of all the tuples of the same kind, like a sort of cache memory—which is configurable. Active spaces distribute and synchronise data across the network and proactively notifies applications of changes, thus can be used as a coordination mechanism for building distributed systems. Likewise GigaSpaces the core API provides actions to put, read, and withdraw tuples, as well as transaction-related operations and a way to subscribe to notifications of tuple changes.

Both GigaSpaces and ActiveSpaces borrow many concepts from tuple-based coordination, hence from the archetypal LINDA model. Then, enrich the basic model with many handy features critical for a mature, industry-ready product, such as transactions, access control, replication. It is worth emphasising that such features also appear in Figure 6, as they have been considered in the many technologies building on LINDA, such as LINC and TuCSoN.

IoT platforms. In the case of IoT platforms, we found no explicit mentioning of tuple spaces or shared data spaces in general, as was in the case of GigaSpaces and ActiveSpaces IMDG. However, many software products provide functionalities aligned with the purpose of coordination technologies, as tailored to the peculiarities of the IoT application domain. For instance, many IoT platforms deal with the issues of *data exchange* between heterogeneous, possibly mobile devices scattered across a network, and of triggering appropriate actions based on such data, in the right sequence, on the right device—essentially, a coordination problem.

All the big players in the market, such as Amazon, Google, Microsoft, and IBM provide cloud solutions and are currently striving to extend their reach towards the Edge of the network [142]. AWS IoT, Google Cloud IoT, MS Azure IoT, and IBM Watson IoT [40] all provide their own way of *(i)* configuring virtual representation of physical devices (e.g. AWS IoT “shadow” objects) to be

---

[40] `https://aws.amazon.com/it/iot/`, `https://cloud.google.com/solutions/iot/?hl=it`, `https://azure.microsoft.com/it-it/overview/iot/`, `https://www.ibm.com/it-it/internet-of-things`

managed by the platform, *(ii)* exploiting publish/subscribe blackboards for communication, *(iii)* exploiting event notification services for reactive computation, and *(iv)* program rules (e.g. Google Cloud IoT "functions") to connect events, data streams, and device status updates to various kinds of actions (either on physical devices or on other Cloud services), even in a graphical way requiring little programming background (as in the case of MS Azure IoT "telemetry rules").

Given the above, we can easily devise out a conceptual mapping where virtual devices are interacting agents or processes, blackboards are realised on top of tuple spaces or suitably composed channels, and rules are dataflow pipelines as in Reo or reactions as in TuCSoN and LINC. The concepts and the intended purpose are the same, albeit the implementation emphasises different aspects for obvious reasons, as stemming from the target audience intended for the technology—other researchers or industrial practitioners.

Indeed, the idea of tuples spaces becomes more and more relevant with the rise of novel computing paradigms and technologies, such as edge computing [142], the Internet of Things, local clouds, and so on. There, data and inter-process / system communication is becoming more and more relevant, and the focus is not only on enabling sharing of data with seamless interoperability while still maintaining loosely coupled components, but also on ensuring correctness of the overall system behaviour, which often critically depends on the correctness of component interactions. In this respect, technologies built out of well-defined coordination models can deliver a lot of value in terms of "correctness by design" and opportunity for formal verification.

Insights. In conclusion, coordination technologies as intended within the COORDINATION community are not in the industry, yet, even though they answer to several of the key challenges faced today, which will become even more relevant in the near future. We can only make informed guesses on the reasons behind the lack of adoption, and on the possible improvements to be pursued by the COORDINATION community to make an impact in the industry. Possibly the most apparent one is the gap in technological tools supporting development and deployment of coordination mechanisms, protocols, and policies: although the technologies are there, both the languages and the middleware, often there are no tools supporting integration with mainstream programming languages and platforms, as well as there are no tools for monitoring system operation or ease deployment to production.

Conversely, the industry needs to rely on actual tools to develop, validate, deploy, monitor, and update their systems, while minimising disruption on already operating deployments. Even if coordination languages are promising in terms of modelling and verification capacities they will not be used by industrial practitioneers without the required tools. Integrated Development Environments, for instance, are mandatory, as well as specific monitoring and debugging tools tailored to the peculiarities of coordination activities. Achieving better support in this facets would undoubtedly boost adoption of coordination technologies as “core” components of future commercial products dedicated to service orchestration, composition, as well as data exchange and sharing.

6. Conclusion

The main aim of this paper was to provide insights about the state of art of coordination technologies after twenty years of the COORDINATION conference series, and to stimulate informed discussion about future perspectives, as well as nurture a fertile ground for further research activity. Overall, apart from some notable success stories – i.e. the commercial success of LINC along with the active development of TuCSoN, Reo, X-Klaim, and Logic Fragments – most coordination technologies have gone through a rapid and effective development at the time they were presented, then lacked further improvements or even maintenance of their usability, thus never reached a wider audience—i.e. outside the COORDINATION community or in the industry.

Obviously, something also happens outside the COORDINATION boundaries, as overviewed in Section 4: for instance, coordination technologies are surveyed in [143], whereas [144] focuses on tuple-based technologies, however, a great deal of the technological developments reported in this survey happened after those papers were published, in 2001 [145]. Also, although the insights delivered in this paper are necessarily limited in scope as restricted to a sample of coordination-related conferences, they represent a great deal of what happened in the research area of coordination models, languages, and technologies, as those concepts and mechanisms presented elsewhere have often times been later presented at COORDINATION suitably expanded, generalised, or specialised to best match coordination problems and needs.

As regards the industry sector, it has shown some initial penetration of coordination concepts, and steadily increasing attention to the issue of, e.g., service orchestration, hence interaction between systems components. Nevertheless, actual usage of coordination technologies born within the academia is rather limited. This is mostly due to the inherent diversity in goals pursued: although there exist academic products which are rather complete and usable, they are rarely geared towards industrial deployment, for instance as concerns ease of deployment, interoperability, security and privacy, and streamlined development process.

Although we acknowledge that researchers are usually mostly concerned with providing scientifically-relevant models rather than production-ready software, we also believe that backing up models and languages with more then proof-of-concept software is crucial to promote wider adoption of both the technology itself and the models, which in turn may provide invaluable feedback to researchers for further developing and tuning models.

In summary, the COORDINATION conference is quite healthy and extremely relevant: although the number of published papers is decreasing, citations and downloads keep growing, contributions conveying technological advancements represent almost a half of all the contributions, and similar conferences seem to look favourably at its results. The next decade will probably tell us more about the actual role of coordination technologies in the development of forthcoming application scenarios: the IoT, for instance, was right at the start of the descending slope in the “peak of inflated expectations” according to Gartner’s hype cycle for 2018, and expected to reach the plateau in 2 to 5 years. This means the time is ripe for pushing forward the development of coordination technologies, so as to have them ready when the IoT will be mature enough to actually benefit from their added value.

Mobile phones, laptops, tablets, even autonomous cars locally connected to each other to form huge computing and storage infrastructures, although currently under-exploited, are the kind of infrastructures paving the way for a new category of services based on data propagation among devices, e.g. car traffic control services through vehicle-to-vehicle communication, information dissemination in a crowd to better steer the crowd towards points of interest or emergency exits, and alternative communication infrastructures in case of environmental disasters. Such services are time-related, as they may last just for a very short time for a specific purpose of exploiting current contextual data, as well as space-related, as they have a meaning because the data they rely on (or the data they spread) is spatially distributed over a geographic area. Coordination models and their correspondent technologies are particularly well suited for these kinds of IoT applications, supporting highly adaptive services able to cope with the dynamism implied by the underlying mobile and changing computing infrastructures, the spatiality of the considered data, and time-related issues [146].

Not too far from the IoT landscape, Digital Twins [147], for instance, is a recent trend aiming at “providing a digital replica of real-world devices, processes or even persons” [148]. A digital twin, provided of the specification of its original counterpart, evolves throughout the lifecycle of the latter, and is mainly used in industry for keeping track of current status or overview of devices or processes, for running simulations, or exploring scenarios (“what-if” analysis). Interest for digital twins is growing, and from initial industry applications the research activity is moving towards personalised medicine, transport infrastructure and maintenance, monitoring and prediction of cyber-physical systems, and managing data arising from IoT deployments. Globally, a digital twin can be considered as software agent with a model of its physical self, and its environment, plus additional data. Coordination technologies naturally work well with such a notion of agent, thus it is reasonable to expect that coordination technologies will further facilitate the development of dynamic, adaptive, collective, and AI-enhanced applications involving the use of digital twins.

References

[1] F. Arbab, The IWIM model for coordination of concurrent activities, in: Ciancarini and Hankin [155], pp. 34–56. `doi:10.1007/3-540-61052-9`.

[2] M. Banville, Sonia: An adaptation of Linda for coordination of activities in organizations, in: Ciancarini and Hankin [155], pp. 57–74. `doi:10.1007/3-540-61052-9`.

[3] R. Tolksdorf, Coordinating services in open distributed systems with Laura, in: Ciancarini and Hankin [155], pp. 386–402. `doi:10.1007/3-540-61052-9`.

[4] K. De Bosschere, J.-M. Jacquet, μ2Log: Towards remote coordination, in: Ciancarini and Hankin [155], pp. 142–159. `doi:10.1007/3-540-61052-9`.

[5] M. Fukuda, L. F. Bic, M. B. Dillencourt, F. Merchant, Intra- and inter-object coordination with MESSENGERS, in: Ciancarini and Hankin [155], pp. 179–196. `doi:10.1007/3-540-61052-9`.

[6] E. Denti, A. Natali, A. Omicini, M. Venuti, An extensible framework for the development of coordinated applications, in: Ciancarini and Hankin [155], pp. 305–320. `doi:10.1007/3-540-61052-9`.

[7] R. van der Goot, J. Schaeffer, G. V. Wilson, Safer tuple spaces, in: Garlan and Le Métayer [149], pp. 289–301. `doi:10.1007/3-540-63383-9`.

[8] A. I. T. Rowstron, Using asynchronous tuple-space access primitives (bonita primitives) for process co-ordination, in: Garlan and Le Métayer [149], pp. 426–429. `doi:10.1007/3-540-63383-9_98`.

[9] R. Tolksdorf, Berlinda: An object-oriented platform for implementing coordination languages in Java, in: Garlan and Le Métayer [149], pp. 430–433. `doi:10.1007/3-540-63383-9_99`.

[10] C. Bryce, M. Oriola, J. Vitck, A coordination model for agents based on secure spaces, in: Ciancarini and Wolf [156], pp. 4–20. `doi:10.1007/3-540-48919-3`.

[11] C. Varela, G. Agha, A hierarchical model for coordination of concurrent activities, in: Ciancarini and Wolf [156], pp. 166–182. `doi:10.1007/3-540-48919-3_13`.

[12] S. Jagannathan, Communication-passing style for coordination languages, in: Garlan and Le Métayer [149], pp. 131–149. `doi:10.1007/3-540-63383-9`.

[13] D. Rossi, F. Vitali, Internet-based coordination environments and document-based applications: a case study, in: Ciancarini and Wolf [156], pp. 259–274. `doi:10.1007/3-540-48919-3`.

[14] M. Schumacher, F. Chantemargue, B. Hirsbrunner, The STL++ coordination language: A base for implementing distributed multi-agent applications, in: Ciancarini and Wolf [156], pp. 399–414. `doi:10.1007/3-540-48919-3`.

[15] N. Sample, D. Beringer, L. Melloul, G. Wiederhold, CLAM: Composition language for autonomous megamodules, in: Ciancarini and Wolf [156], pp. 291–306. `doi:10.1007/3-540-48919-3`.

[16] M. Cremonini, A. Omicini, F. Zambonelli, Coordination in context: Authentication, authorisation and topology in mobile agent applications, in: Ciancarini and Wolf [156], pp. 416–416. `doi:10.1007/3-540-48919-3`.

[17] W. C. Jamison, D. Lea, TRUCE: Agent coordination through concurrent interpretation of role-based protocols, in: Ciancarini and Wolf [156], pp. 384–398. `doi:10.1007/3-540-48919-3`.

[18] J. C. Cruz, S. Ducasse, A group based approach for coordinating active objects, in: Ciancarini and Wolf [156], pp. 355–370. `doi:10.1007/3-540-48919-3`.

[19] S. Ducasse, T. Hofmann, O. Nierstrasz, Openspaces: An object-oriented framework for reconfigurable coordination spaces, in: Porto and Roman [157], pp. 1–18.

[20] F. Achermann, S. Kneubuehl, O. Nierstrasz, Scripting coordination styles, in: Porto and Roman [157], pp. 19–35.

[21] X. Ao, N. Minsky, T. D. Nguyen, V. Ungureanu, Law-Governed Internet communities, in: Porto and Roman [157], pp. 133–147.

[22] I. Merrick, A. Wood, Scoped coordination in open distributed systems, in: Porto and Roman [157], pp. 311–316.

[23] F. Arbab, F. Mavaddat, Coordination through channel composition, in: Arbab and Talcott [154], pp. 22–39. `doi:10.1007/3-540-46000-4`.

[24] R. Tolksdorf, G. Rojec-Goldmann, The SPACETUB models and framework, in: Arbab and Talcott [154], pp. 348–363. `doi:10.1007/3-540-46000-4_32`.

[25] L. Bettini, V. Bono, B. Venneri, O'Klaim: A coordination language with mobile mixins, in: De Nicola et al. [158], pp. 20–37. `doi:10.1007/b95570`.

[26] R. De Nicola, G. L. Ferrari, R. Pugliese, Klaim: a kernel language for agents interaction and mobility, IEEE Transactions on Software Engineering 24 (5) (1998) 315–330. `doi:10.1109/32.685256`.

[27] C.-L. Fok, G.-C. Roman, G. Hackmann, A lightweight coordination middleware for mobile computing, in: De Nicola et al. [158], pp. 135–151. `doi:10.1007/b95570`.

[28] S. Mostinckx, C. Scholliers, E. Philips, C. Herzeel, W. De Meuter, Fact Spaces: Coordination in the face of disconnection, in: Murphy and Vitek [159], pp. 268–285.

[29] J. Dedecker, T. Van Cutsem, S. Mostinckx, T. D'Hondt, W. De Meuter, Ambient-oriented programming, in: Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '05, ACM, New York, NY, USA, 2005, pp. 31–40. `doi:10.1145/1094855.1094867`.

[30] E. Simperl, R. Krummenacher, L. Nixon, A coordination model for triplespace computing, in: Murphy and Vitek [159], pp. 1–18.

[31] D. Fensel, Triple-space computing: Semantic web services based on persistent publication of information, in: F. A. Aagesen, C. Anutariya, V. Wuwongse (Eds.), Intelligence in Communication Systems, Vol. 3283 of LNCS, Springer, 2004, pp. 43–53.

[32] R. Sen, G.-C. Roman, C. Gill, CiAN: A workflow engine for MANETs, in: Lea and Zavattaro [152], pp. 280–295. `doi:10.1007/978-3-540-68265-3_18`.

[33] J. Abreu, J. L. Fiadeiro, A coordination model for service-oriented interactions, in: Lea and Zavattaro [152], pp. 1–16. `doi:10.1007/978-3-540-68265-3`.

[34] S. Gilmore, J. Hillston, The PEPA workbench: A tool to support a process algebra-based approach to performance modelling, in: G. Haring, G. Kotsis (Eds.), Computer Performance Evaluation Modelling Techniques and Tools, Vol. 794 of LNCS, Springer, 1994, pp. 353–368. `doi:10.1007/3-540-58021-2_20`.

[35] L. Bettini, R. De Nicola, M. Loreti, Implementing session centered calculi, in: Lea and Zavattaro [152], pp. 17–32. `doi:10.1007/978-3-540-68265-3`.

[36] L. Bettini, R. De Nicola, D. Falassi, M. Lacoste, L. Lopes, L. Oliveira, H. Paulino, V. T. Vasconcelos, A software framework for rapid prototyping of run-time systems for mobile calculi, in: C. Priami, P. Quaglia (Eds.), Global Computing, Springer, 2005, pp. 179–207.

[37] P. Tarau, Coordination and concurrency in multi-engine Prolog, in: De Meuter and Roman [150], pp. 157–171. `doi:10.1007/978-3-642-21464-6`.

[38] H. Plociniczak, S. Eisenbach, JErlang: Erlang with Joins, in: D. Clarke, G. Agha (Eds.), Coordination Models and Languages, Vol. 6116 of LNCS, Springer, 2010, pp. 61–75.

[39] C. Fournet, G. Gonthier, The reflexive CHAM and the Join-calculus, in: 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM, 1996, pp. 372–385. `doi:10.1145/237721.237805`.

[40] N. Ng, N. Yoshida, O. Pernet, R. Hu, Y. Kryftis, Safe parallel programming with Session Java, in: De Meuter and Roman [150], pp. 110–126. `doi:10.1007/978-3-642-21464-6`.

[41] K. Honda, V. T. Vasconcelos, M. Kubo, Language primitives and type discipline for structured communication-based programming, in: C. Hankin (Ed.), Programming Languages and Systems, Vol. 1381 of LNCS, Springer, 1998, pp. 122–138. `doi:10.1007/BFb0053567`.

[42] D. Rossi, A social software-based coordination platform, in: Sirjani [151], pp. 17–28. `doi:10.1007/978-3-642-30829-1_2`.

[43] V. Liptchinsky, R. Khazankin, H.-L. Truong, S. Dustdar, Statelets: Coordination of social collaboration processes, in: Sirjani [151], pp. 1–16. `doi:10.1007/978-3-642-30829-1_1`.

[44] J. Proença, D. Clarke, Interactive interaction constraints, in: R. De Nicola, C. Julien (Eds.), Coordination Models and Languages, Vol. 7890 of LNCS, Springer, 2013, pp. 211–225.

[45] M. Louvel, F. Pacull, LINC: A compact yet powerful coordination environment, in: E. Kühn, R. Pugliese (Eds.), Coordination Models and Languages, Vol. 8459 of LNCS, Springer, 2014, pp. 83–98.

[46] D. Gelernter, Generative communication in Linda, ACM Transactions on Programming Languages and Systems (TOPLAS) 7 (1) (1985) 80–112. `doi:10.1145/2363.2433`.

[47] M. Andrić, R. De Nicola, A. L. Lafuente, Replica-based high-performance tuple space computing, in: Holvoet and Viroli [153], pp. 3–18. `doi:10.1007/978-3-319-19282-6`.

[48] F. L. De Angelis, G. Di Marzo Serugendo, Logic Fragments: A coordination model based on logic inference, in: Holvoet and Viroli [153], pp. 35–48. `doi:10.1007/978-3-319-19282-6`.

[49] F. Zambonelli, A. Omicini, et al., Developing pervasive multi-agent systems with nature-inspired coordination, Pervasive and Mobile Computing 17 (2015) 236–252. `doi:10.1016/j.pmcj.2014.12.002`.

[50] A. I. T. Rowstron, WCL: A co-ordination language for geographically distributed agents, World Wide Web 1 (3) (1998) 167–179. `doi:10.1023/A:1019263731139`.

[51] P. Ciancarini, D. Rossi, Jada: Coordination and communication for Java agents, in: J. Vitek, C. Tschudin (Eds.), Mobile Object Systems Towards the Programmable Internet, Vol. 1222 of LNCS, Springer, 1997, pp. 213–226.

[52] J.-P. Banâtre, P. Fradet, D. Le Métayer, Gamma and the chemical reaction model: Fifteen years after, in: C. S. Calude, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing, Vol. 2235 of LNCS, Springer, 2001, pp. 17–44.

[53] M. Louvel, F. Pacull, E. Rutten, A. N. Sylla, Development tools for rule-based coordination programming in LINC, in: J.-M. Jacquet,

M. Massink (Eds.), Coordination Models and Languages, Vol. 10319 of LNCS, Springer, 2017, pp. 78–96. `doi:10.1007/978-3-319-59746-1`.

[54] R. De Nicola, G. Ferrari, R. Pugliese, Coordinating mobile agents via blackboards and access rights, in: D. Garlan, D. Le Métayer (Eds.), Coordination Languages and Models, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 220–237.

[55] L. Bettini, R. De Nicola, R. Pugliese, X-Klaim and Klava: Programming mobile code, Electronic Notes in Theoretical Computer Science 62 (2002) 24–37. `doi:10.1016/S1571-0661(04)00317-2`.

[56] L. Bettini, E. Merelli, F. Tiezzi, X-Klaim Is Back, Springer International Publishing, Cham, 2019, pp. 115–135. `doi:10.1007/978-3-030-21485-2_8`.
URL `https://doi.org/10.1007/978-3-030-21485-2_8`

[57] A. L. Murphy, G. P. Picco, G.-C. Roman, LIME: A coordination model and middleware supporting mobility of hosts and agents, ACM Transactions on Software Engineering and Methodology (TOSEM) 15 (3) (2006) 279–328. `doi:10.1145/1151695.1151698`.

[58] R. Hu, N. Yoshida, K. Honda, Session-based distributed programming in Java, in: J. Vitek (Ed.), ECOOP 2008 – Object-Oriented Programming, Vol. 5142 of LNCS, Springer, 2008, pp. 516–541. `doi:10.1007/978-3-540-70592-5_22`.

[59] A. I. T. Rowstron, Bulk primitives in Linda run-time systems, Ph.D. thesis, The University of York (1996).

[60] S. Mariani, A. Omicini, Coordination mechanisms for the modelling and simulation of stochastic systems: The case of uniform primitives, SCS M&S Magazine IV (3) (2014) 6–25.

[61] A. Omicini, E. Denti, From tuple spaces to tuple centres, Science of Computer Programming 41 (3) (2001) 277–294. `doi:10.1016/S0167-6423(01)00011-9`.

[62] A. Omicini, Formal `ReSpecT` in the A&A perspective, Electronic Notes in Theoretical Computer Science 175 (2) (2007) 97–117. `doi:10.1016/j.entcs.2007.03.006`.

[63] M. Viroli, A. Omicini, Coordination as a service, Fundamenta Informaticae 73 (4) (2006) 507–534.

[64] E. Denti, A. Omicini, A. Ricci, `tu`Prolog: A light-weight Prolog for Internet applications and infrastructures, in: I. Ramakrishnan (Ed.), Practical Aspects of Declarative Languages, Vol. 1990 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 184–198, 3rd International Symposium (PADL 2001), Las Vegas, NV, USA, 11–12 Mar. 2001. Proceedings. `doi:10.1007/3-540-45241-9_13`.
URL `http://link.springer.com/10.1007/3-540-45241-9_13`

[65] S. Mariani, A. Omicini, L. Sangiorgi, Models of autonomy and coordination: Integrating subjective & objective approaches in agent development frameworks, in: L. Braubach, D. Camacho, S. Venticinque, C. Bădică (Eds.), Intelligent Distributed Computing VIII, Vol. 570 of SCI, Springer International Publishing, 2015, pp. 69–79. `doi:10.1007/978-3-319-10422-5_9`.

[66] S. Mariani, A. Omicini, Multi-paradigm coordination for MAS: Integrating heterogeneous coordination approaches in MAS technologies, in: C. Santoro, F. Messina, M. De Benedetti (Eds.), WOA 2016 – 17th Workshop "From Objects to Agents", Vol. 1664 of CEUR-WS.org, Sun SITE Central Europe, 2016, pp. 91–99.

[67] F. L. Bellifemine, A. Poggi, G. Rimassa, JADE–a FIPA-compliant agent framework, in: 4th International Conference and Exhibition on the Practi-

cal Application of Intelligent Agents and Multi-Agent Technology (PAAM-99), 1999, pp. 97–108.

[68] R. H. Bordini, J. F. Hübner, M. J. Wooldridge, Programming Multi-Agent Systems in AgentSpeak using Jason, John Wiley & Sons, Ltd, 2007.

[69] G. Ciatto, S. Mariani, A. Omicini, ReSpecT𝕏: Programming interaction made easy, Computer Science and Information Systems 15 (3) (2018) 655–682, Special Section: Contemporary Topics in Intelligent Distributed Computing. `doi:10.2298/CSIS180111031C`.
URL `http://www.comsis.org/archive.php?show=ppridc-7418`

[70] R. Calegari, E. Denti, Building Smart Spaces on the Home Manager platform, ALP Newsletter.

[71] A. Dubovitskaya, V. Urovi, I. Barba, K. Aberer, M. I. Schumacher, A multiagent system for dynamic data aggregation in medical research, BioMed Research International 2016. `doi:10.1155/2016/9027457`.

[72] G. Ciatto, L. Rizzato, A. Omicini, S. Mariani, Tusow: Tuple spaces for edge computing, in: The 28th International Conference on Computer Communications and Networks (ICCCN 2019), Valencia, Spain, 2019.

[73] N. H. Minsky, J. Leichter, Law-Governed Linda as a coordination model, in: P. Ciancarini, O. Nierstrasz, A. Yonezawa (Eds.), Object-Based Models and Languages for Concurrent Systems, Vol. 924 of LNCS, Springer, 1994, pp. 125–146. `doi:10.1007/3-540-59450-7_8`.

[74] C. Baier, M. Sirjani, F. Arbab, J. Rutten, Modeling component connectors in reo by constraint automata, Science of Computer Programming 61 (2) (2006) 75–113. `doi:10.1016/j.scico.2005.10.008`.

[75] S.-S. T. Q. Jongmans, F. Santini, M. Sargolzaei, F. Arbab, H. Afsarmanesh, Orchestrating web services using Reo: from circuits and behaviors to automatically generated code, Service Oriented Computing and Applications 8 (4) (2014) 277–297. `doi:10.1007/s11761-013-0147-1`.

[76] N. Kokash, C. Krause, E. de Vink, Reo + mCRL2: A framework for model-checking dataflow in service compositions, Formal Aspects of Computing 24 (2) (2012) 187–216. `doi:10.1007/s00165-011-0191-6`.

[77] J. Proença, D. Clarke, E. de Vink, F. Arbab, Dreams: A framework for distributed synchronous coordination, in: 27th Annual ACM Symposium on Applied Computing (SAC 2012), ACM, New York, NY, USA, 2012, pp. 1510–1515. `doi:10.1145/2245276.2232017`.

[78] J. A. Hendler, Agents and the Semantic Web, IEEE Intelligent Systems 16 (2) (2001) 30–37. `doi:10.1109/5254.920597`.

[79] L. Bettini, V. Bono, B. Venneri, Coordinating mobile object-oriented code, in: Arbab and Talcott [154], pp. 56–71. `doi:10.1007/3-540-46000-4`.

[80] L. Bettini, R. de Nicola, R. Pugliese, G. L. Ferrari, Interactive mobile agents in x-klaim, in: Proceedings Seventh IEEE International Workshop on Enabling Technologies: Infrastucture for Collaborative Enterprises (WET ICE '98) (Cat. No.98TB100253), 1998, pp. 110–115. `doi:10.1109/ENABL.1998.725680`.

[81] L. Bettini, R. De Nicola, R. Pugliese, Klava: a java package for distributed and mobile applications, Software: Practice and Experience 32 (14) (2002) 1365–1394. `arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.486`, `doi:10.1002/spe.486`.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.486`

[82] F. L. De Angelis, G. Di Marzo Serugendo, Logic fragments: A coordination model based on logic inference, in: T. Holvoet, M. Viroli (Eds.), Coordination Models and Languages, Springer International Publishing, Cham, 2015, pp. 35–48.

[83] G. Castelli, M. Mamei, A. Rosi, F. Zambonelli, Pervasive middleware goes social: The sapere approach, in: Proceedings of the 2011 Fifth

IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW '11, 2010, pp. 9–14.

[84] F. De Angelis, A logic-based coordination middleware for self-organising systems: distributed reasoning based on many-valued logics, Ph.D. thesis, University of Geneva, School of Social Sciences - Information Systems (2017).

[85] J. L. Fernandez-Marquez, F. D. Angelis, G. D. M. Serugendo, G. Stevenson, G. Castelli, The one-sapere simulator: A prototyping tool for engineering self-organisation in pervasive environments., in: SASO, IEEE Computer Society, 2014, pp. 201–202.
URL `http://dblp.uni-trier.de/db/conf/saso/saso2014.html#Fernandez-MarquezASSC14`

[86] A. Keränen, J. Ott, T. Kärkkäinen, The ONE Simulator for DTN Protocol Evaluation, in: SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, ICST, New York, NY, USA, 2009.

[87] G. P. Picco, A. L. Murphy, G. C. Roman, LIME: Linda meets mobility, in: 1999 International Conference on Software Engineering (ICSE 1999), 1999, pp. 368–377. `doi:10.1145/302405.302659`.

[88] A. Omicini, Hybrid coordination models for handling information exchange among Internet agents, in: A. Bonarini, M. Colombetti, P. L. Lanzi (Eds.), Workshop "Agenti intelligenti e Internet: teorie, strumenti e applicazioni", 7th AI*IA Convention (AI*IA 2000), Milano, Italy, 2000, pp. 1–4.

[89] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Computer Networks 54 (15) (2010) 2787–2805. `doi:10.1016/j.comnet.2010.05.010`.

[90] J. Heuer, J. Hund, O. Pfaff, Toward the Web of Things: Applying Web technologies to the physical world, Computer 48 (5) (2015) 34–42. `doi:10.1109/MC.2015.152`.

[91] A. Arsénio, H. Serra, R. Francisco, F. Nabais, J. Andrade, E. Serrano, Internet of Intelligent Things: Bringing artificial intelligence into things and communication networks, in: F. Xhafa, N. Bessis (Eds.), Inter-cooperative Collective Intelligence: Techniques and Applications, Vol. 495 of SCI, Springer, 2014, pp. 1–37. `doi:10.1007/978-3-642-35016-0_1`.

[92] A. Ricci, A. Omicini, E. Denti, Virtual enterprises and workflow management as agent coordination issues, International Journal of Cooperative Information Systems 11 (3/4) (2002) 355–379. `doi:10.1142/S0218843002000637`.

[93] B. Cheng, D. Zhu, S. Zhao, J. Chen, Situation-aware iot service coordination using the event-driven soa paradigm, IEEE Transactions on Network and Service Management 13 (2) (2016) 349–361. `doi:10.1109/TNSM.2016.2541171`.

[94] G. Fortino, A. Guerrieri, W. Russo, C. Savaglio, Integration of agent-based and cloud computing for the smart objects-oriented iot, in: Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2014, pp. 493–498. `doi:10.1109/CSCWD.2014.6846894`.

[95] N. K. Giang, M. Blackstock, R. Lea, V. C. M. Leung, Developing iot applications in the fog: A distributed dataflow approach, in: 2015 5th International Conference on the Internet of Things (IOT), 2015, pp. 155–162. `doi:10.1109/IOT.2015.7356560`.

[96] E. Denti, A. Natali, A. Omicini, On the expressive power of a language for programming coordination media, in: Proceedings of the 1998 ACM Symposium on Applied Computing, SAC '98, ACM, New York, NY, USA,

1998, pp. 169–177. `doi:10.1145/330560.330665`.
URL `http://doi.acm.org/10.1145/330560.330665`

[97] A. Omicini, F. Zambonelli, Tuple centres for the coordination of internet agents, in: Proceedings of the 1999 ACM Symposium on Applied Computing, SAC '99, ACM, New York, NY, USA, 1999, pp. 183–190. `doi:10.1145/298151.298231`.
URL `http://doi.acm.org/10.1145/298151.298231`

[98] G. Cabri, L. Leonardi, F. Zambonelli, Xml dataspaces for mobile agent coordination, in: Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 1, SAC '00, ACM, New York, NY, USA, 2000, pp. 181–188. `doi:10.1145/335603.335738`.
URL `http://doi.acm.org/10.1145/335603.335738`

[99] G. Cabri, L. Leonardi, F. Zambonelli, Reactive tuple spaces for mobile agent coordination, in: K. Rothermel, F. Hohl (Eds.), Mobile Agents, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 237–248.

[100] P. Zoeteweij, Coordination-based distributed constraint solving in dice, in: Proceedings of the 2003 ACM Symposium on Applied Computing, SAC '03, ACM, New York, NY, USA, 2003, pp. 360–366. `doi:10.1145/952532.952605`.
URL `http://doi.acm.org/10.1145/952532.952605`

[101] M. Mamei, F. Zambonelli, Self-maintained distributed tuples for field-based coordination in dynamic networks, in: Proceedings of the 2004 ACM Symposium on Applied Computing, SAC '04, ACM, New York, NY, USA, 2004, pp. 479–486. `doi:10.1145/967900.968000`.
URL `http://doi.acm.org/10.1145/967900.968000`

[102] M. Mamei, F. Zambonelli, L. Leonardi, Tuples on the air: A middleware for context-aware computing in dynamic networks, in: 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings., IEEE, 2003, pp. 342–347.

[103] M. Viroli, M. Casadei, A. Omicini, A framework for modelling and implementing self-organising coordination, in: Proceedings of the 2009 ACM Symposium on Applied Computing, SAC '09, ACM, New York, NY, USA, 2009, pp. 1353–1360. `doi:10.1145/1529282.1529585`.
URL `http://doi.acm.org/10.1145/1529282.1529585`

[104] E. Nardini, M. Viroli, E. Panzavolta, Coordination in open and dynamic environments with tucson semantic tuple centres, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10, ACM, New York, NY, USA, 2010, pp. 2037–2044. `doi:10.1145/1774088.1774515`.
URL `http://doi.acm.org/10.1145/1774088.1774515`

[105] M. Sbaraglia, M. Casadei, M. Viroli, Programming coordination laws of artifacts in cartago, in: Proceedings of the 2011 ACM Symposium on Applied Computing, SAC '11, ACM, New York, NY, USA, 2011, pp. 885–886. `doi:10.1145/1982185.1982378`.
URL `http://doi.acm.org/10.1145/1982185.1982378`

[106] A. Ricci, M. Piunti, M. Viroli, A. Omicini, Environment programming in CArtAgO, in: R. P. Bordini, M. Dastani, J. Dix, A. El Fallah Seghrouchni (Eds.), Multi-Agent Programming II: Languages, Platforms and Applications, Multiagent Systems, Artificial Societies, and Simulated Organizations, Springer, 2009, Ch. 8, pp. 259–288. `doi:10.1007/978-0-387-89299-3_8`.
URL `http://link.springer.com/chapter/10.1007/978-0-387-89299-3_8`

[107] M. Viroli, D. Pianini, S. Montagna, G. Stevenson, Pervasive ecosystems: A coordination model based on semantic chemistry, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, ACM, New York, NY, USA, 2012, pp. 295–302. `doi:10.1145/2245276.2245336`.
URL `http://doi.acm.org/10.1145/2245276.2245336`

[108] D. Pianini, S. Montagna, M. Viroli, A chemical inspired simulation framework for pervasive services ecosystems, in: M. Ganzha, L. Maciaszek, M. Paprzycki (Eds.), Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS 2011), IEEE Computer Society Press, Szczecin, Poland, 2011, pp. 667–674.

[109] G. Stevenson, J. Ye, S. Dobson, D. Pianini, S. Montagna, M. Viroli, Combining self-organisation, context-awareness and semantic reasoning: The case of resource discovery in opportunistic networks, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, ACM, New York, NY, USA, 2013, pp. 1369–1376. `doi:10.1145/2480362.2480619`.
URL `http://doi.acm.org/10.1145/2480362.2480619`

[110] D. Pianini, M. Viroli, J. Beal, Protelis: Practical aggregate programming, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, ACM, New York, NY, USA, 2015, pp. 1846–1853. `doi:10.1145/2695664.2695913`.
URL `http://doi.acm.org/10.1145/2695664.2695913`

[111] J. Proença, D. Clarke, E. de Vink, F. Arbab, Dreams: A framework for distributed synchronous coordination, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, ACM, New York, NY, USA, 2012, pp. 1510–1515. `doi:10.1145/2245276.2232017`.
URL `http://doi.acm.org/10.1145/2245276.2232017`

[112] V. Urovi, A. C. Olivieri, S. Bromuri, N. Fornara, M. I. Schumacher, A peer to peer agent coordination framework for ihe based cross-community health record exchange, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, ACM, New York, NY, USA, 2013, pp. 1355–1362. `doi:10.1145/2480362.2480617`.
URL `http://doi.acm.org/10.1145/2480362.2480617`

[113] N. Oliveira, A. Silva, L. S. Barbosa, Quantitative analysis of reo-based

service coordination, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC '14, ACM, New York, NY, USA, 2014, pp. 1247–1254. `doi:10.1145/2554850.2555025`.
URL `http://doi.acm.org/10.1145/2554850.2555025`

[114] C. Baier, V. Wolf, Stochastic reasoning about channel-based component connectors, in: P. Ciancarini, H. Wiklicky (Eds.), Coordination Models and Languages, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–15.

[115] M. Díaz, B. Rubio, J. M. Troya, Multilingual and multiparadigm integration of a tuple channel-based coordination model, in: Proceedings of the 1998 ACM Symposium on Applied Computing, SAC '98, ACM, New York, NY, USA, 1998, pp. 194–196. `doi:10.1145/330560.330668`.
URL `http://doi.acm.org/10.1145/330560.330668`

[116] T. Rauber, G. Rünger, A coordination language for mixed task and and data parallel programs, in: Proceedings of the 1999 ACM Symposium on Applied Computing, SAC '99, ACM, New York, NY, USA, 1999, pp. 146–155. `doi:10.1145/298151.298224`.
URL `http://doi.acm.org/10.1145/298151.298224`

[117] P. Ciancarini, R. Tolksdorf, F. Zambonelli, Coordination middleware for xml-centric applications, in: Proceedings of the 2002 ACM Symposium on Applied Computing, SAC '02, ACM, New York, NY, USA, 2002, pp. 336–343. `doi:10.1145/508791.508857`.
URL `http://doi.acm.org/10.1145/508791.508857`

[118] R. Tolksdorf, Coordination technology for workflows on the web: Workspaces, in: Proceedings of the 4th International Conference on Coordination Languages and Models, COORDINATION '00, Springer-Verlag, Berlin, Heidelberg, 2000, pp. 36–50.
URL `http://dl.acm.org/citation.cfm?id=647016.713307`

[119] C. Mascolo, L. Capra, S. Zachariadis, W. Emmerich, Xmiddle: A data-sharing middleware for mobile computing, Wirel. Pers. Commun. 21 (1) (2002) 77–103. `doi:10.1023/A:1015584805733`.
URL `https://doi.org/10.1023/A:1015584805733`

[120] G. Fortino, W. Russo, Multi-coordination of mobile agents: A model and a component-based architecture, in: Proceedings of the 2005 ACM Symposium on Applied Computing, SAC '05, ACM, New York, NY, USA, 2005, pp. 443–450. `doi:10.1145/1066677.1066779`.
URL `http://doi.acm.org/10.1145/1066677.1066779`

[121] M. Murth, E. Kühn, Knowledge-based coordination with a reliable semantic subscription mechanism, in: Proceedings of the 2009 ACM Symposium on Applied Computing, SAC '09, ACM, New York, NY, USA, 2009, pp. 1374–1380. `doi:10.1145/1529282.1529588`.
URL `http://doi.acm.org/10.1145/1529282.1529588`

[122] D. Harnie, T. D'Hondt, E. G. Boix, W. De Meuter, Programming urban-area applications, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, ACM, New York, NY, USA, 2012, pp. 1516–1521. `doi:10.1145/2245276.2232018`.
URL `http://doi.acm.org/10.1145/2245276.2232018`

[123] C. Scholliers, E. G. Boix, W. De Meuter, Totam: Scoped tuples for the ambient, Vol. 19, 2009.

[124] A. Fongen, Data-centric authorization and integrity control in a linda tuplespace, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, ACM, New York, NY, USA, 2015, pp. 1827–1833. `doi:10.1145/2695664.2695681`.
URL `http://doi.acm.org/10.1145/2695664.2695681`

[125] On the "Local-to-Global" Issue in Self-Organisation: Chemical Reactions with Custom Kinetic Rates, Eighth IEEE International Conference

on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2014, IEEE CS, London, UK, 2014, best student paper award. `doi:10.1109/SASOW.2014.14`.
URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7056354`

[126] O. Kosak, C. Wanninger, A. Angerer, A. Hoffmann, A. Schierl, H. Seebach, Decentralized coordination of heterogeneous ensembles using jadex, in: 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W), 2016, pp. 271–272. `doi:10.1109/FAS-W.2016.65`.

[127] A. Pokahr, L. Braubach, W. Lamersdorf, Jadex: A BDI Reasoning Engine, Springer US, Boston, MA, 2005, pp. 149–174. `doi:10.1007/0-387-26350-0_6`.
URL `https://doi.org/10.1007/0-387-26350-0_6`

[128] M. Georgeff, B. Pell, M. Pollack, M. Tambe, M. Wooldridge, The belief-desire-intention model of agency, in: J. P. Müller, A. S. Rao, M. P. Singh (Eds.), Intelligent Agents V: Agents Theories, Architectures, and Languages, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 1–10.

[129] F. A. d. Oliveira, T. Ledoux, R. Sharrock, A framework for the coordination of multiple autonomic managers in cloud environments, in: 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems, 2013, pp. 179–188. `doi:10.1109/SASO.2013.27`.

[130] S. Mariani, A. Omicini, Self-organising news management: the molecules of knowledge approach, in: 2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2012, pp. 235–240. `doi:10.1109/SASOW.2012.48`.

[131] S. Mariani, A. Omicini, Anticipatory coordination in socio-technical knowledge-intensive environments: Behavioural implicit communication

in MoK, in: M. Gavanelli, E. Lamma, F. Riguzzi (Eds.), AI*IA 2015, Advances in Artificial Intelligence, Vol. 9336 of Lecture Notes in Computer Science, Springer International Publishing, 2015, Ch. 8, pp. 102–115, xIVth International Conference of the Italian Association for Artificial Intelligence, Ferrara, Italy, September 23–25, 2015, Proceedings. `doi:10.1007/978-3-319-24309-2_8`.
URL `http://link.springer.com/10.1007/978-3-319-24309-2_8`

[132] S. Mariani, Coordination of Complex Sociotechnical Systems: Self-organisation of Knowledge in MoK, 1st Edition, Artificial Intelligence: Foundations, Theory, and Algorithms, Springer International Publishing, 2016. `doi:10.1007/978-3-319-47109-9`.
URL `http://link.springer.com/10.1007/978-3-319-47109-9`

[133] C. Castelfranchi, G. Pezzulo, L. Tummolini, Behavioral implicit communication (bic): Communicating with smart environments, International Journal of Ambient Computing and Intelligence (IJACI) 2 (1) (2010) 1–12.
URL `https://EconPapers.repec.org/RePEc:igg:jaci00:v:2:y:2010:i:1:p:1-12`

[134] M. Viroli, E. Nardini, G. Castelli, M. Mamei, F. Zambonelli, Towards a coordination approach to adaptive pervasive service ecosystems, in: 2011 IEEE Fifth International Conference on Self-Adaptive and Self-Organizing Systems, 2011, pp. 223–224. `doi:10.1109/SASO.2011.42`.

[135] M. Viroli, E. Nardini, G. Castelli, M. Mamei, F. Zambonelli, A coordination approach to adaptive pervasive service ecosystems, in: 2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2011, pp. 114–119. `doi:10.1109/SASOW.2011.19`.

[136] M. Viroli, M. Casadei, S. Montagna, F. Zambonelli, Spatial coordination of pervasive systems through chemical-inspired tuple spaces, in:

2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop, 2010, pp. 212–217. `doi:10.1109/SASOW.2010.75`.

[137] M. Loreti, A. Margheri, R. Pugliese, F. Tiezzi, On programming and policing autonomic computing systems, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 164–183.

[138] R. De Nicola, M. Loreti, R. Pugliese, F. Tiezzi, A formal approach to autonomic systems programming: The scel language, ACM Trans. Auton. Adapt. Syst. 9 (2) (2014) 7:1–7:29. `doi:10.1145/2619998`.
URL `http://doi.acm.org/10.1145/2619998`

[139] M. Masi, R. Pugliese, F. Tiezzi, Formalisation and implementation of the xacml access control mechanism, in: G. Barthe, B. Livshits, R. Scandariato (Eds.), Engineering Secure Software and Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 60–74.

[140] Y. Abd Alrahman, R. De Nicola, G. Garbi, Goat: Attribute-based interaction in google go, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems, Springer International Publishing, Cham, 2018, pp. 288–303.

[141] E. Freeman, S. Hupfer, K. Arnold, JavaSpaces principles, patterns, and practice, Addison-Wesley Professional, 1999.

[142] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet of Things Journal 3 (5) (2016) 637–646. `doi:10.1109/JIOT.2016.2579198`.

[143] G. A. Papadopoulos, Models and technologies for the coordination of Internet agents: A survey, in: Omicini et al. [145], Ch. 2, pp. 25–56.

[144] D. Rossi, G. Cabri, E. Denti, Tuple-based technologies for coordination, in: Omicini et al. [145], Ch. 4, pp. 83–109.

[145] A. Omicini, F. Zambonelli, M. Klusch, R. Tolksdorf (Eds.), Coordination of Internet Agents: Models, Technologies, and Applications, Springer, 2001.

[146] G. Di Marzo Serugendo, N. Abdennadher, H. Ben Mahfoudh, F. L. De Angelis, R. Tomaylla, Spatial edge services, Global IoT Summit.

[147] Digital twins - rise of the digital twin in industrial iot and industry 4.0. URL `https://www.i-scoop.eu/internet-of-things-guide/industrial-internet-things-iiot-saving-costs-innovation/digital-twins/`

[148] E. Weippl, B. Sanderse, Digital twins - introduction to the special theme, ERCIM News 115 (2018) 6–7.

[149] D. Garlan, D. Le Métayer (Eds.), Coordination Languages and Models. 2nd International Conference COORDINATION '97 Berlin, Germany, September 1–3, 1997 Proceedings, Vol. 1282 of LNCS, Springer, 1997. `doi:10.1007/3-540-63383-9`.

[150] W. De Meuter, G.-C. Roman (Eds.), Coordination Models and Languages. 13th International Conference, COORDINATION 2011, Reykjavik, Iceland, June 6-9, 2011. Proceedings, Vol. 6721 of LNCS, Springer, 2011. `doi:10.1007/978-3-642-21464-6`.

[151] M. Sirjani (Ed.), Coordination Models and Languages. 14th International Conference, COORDINATION 2012, Stockholm, Sweden, June 14-15, 2012. Proceedings, Vol. 7274 of LNCS, Springer, 2012. `doi:10.1007/978-3-642-30829-1`.

[152] D. Lea, G. Zavattaro (Eds.), Coordination Models and Languages. 10th International Conference, COORDINATION 2008, Oslo, Norway, June 4-

6, 2008. Proceedings, Vol. 5052 of LNCS, Springer, 2008. `doi:10.1007/978-3-540-68265-3`.

[153] T. Holvoet, M. Viroli (Eds.), Coordination Models and Languages. 17th International Conference, COORDINATION 2015, Grenoble, France, June 2-4, 2015, Proceedings, Vol. 9037 of LNCS, Springer, 2015. `doi:10.1007/978-3-319-19282-6`.

[154] F. Arbab, C. Talcott (Eds.), Coordination Models and Languages. 5th International Conference, COORDINATION 2002 York, UK, April 8–11, 2002 Proceedings, Vol. 2315 of LNCS, Springer, 2002. `doi:10.1007/3-540-46000-4`.

[155] P. Ciancarini, C. Hankin (Eds.), Coordination Languages and Models. 1st International Conference, COORDINATION '96 Cesena, Italy, April 15–17, 1996 Proceedings, Vol. 1061 of LNCS, Springer, 1996. `doi:10.1007/3-540-61052-9`.

[156] P. Ciancarini, A. L. Wolf (Eds.), Coordination Languages and Models. 3rd International Conference COORDINATION'99 Amsterdam, The Netherlands, April 26–28, 1999 Proceedings, Vol. 1594 of LNCS, Springer, 1999. `doi:10.1007/3-540-48919-3`.

[157] A. Porto, G.-C. Roman (Eds.), Coordination Languages and Models. 4th International Conference, COORDINATION 2000 Limassol, Cyprus, September 11–13, 2000 Proceedings, Vol. 1906 of LNCS, Springer, 2000.

[158] R. De Nicola, G.-L. Ferrari, G. Meredith (Eds.), Coordination Models and Languages. 6th International Conference, COORDINATION 2004 Pisa Italy, February 24-27, 2004 Proceedings, Vol. 2949 of LNCS, Springer, 2004. `doi:10.1007/b95570`.

[159] A. L. Murphy, J. Vitek (Eds.), Coordination Models and Languages. 9th International Conference, COORDINATION 2007, Paphos, Cyprus, June 6-8, 2007. Proceedings, Vol. 4467 of LNCS, Springer, 2007.