Novel Opportunities for Tuple-based Coordination: XPath, the Blockchain, and Stream Processing

Stefano Mariani*, Andrea Omicini†, Giovanni Ciatto†

*Department of Sciences and Methods for Engineering
Università degli Studi di Modena e Reggio Emilia
Reggio Emilia, Italy
Email: stefano.mariani@unimore.it
†Department of Computer Science and Engineering
Università di Bologna
Cesena, Italy
Email: andrea.omicini@unibo.it, giovanni.ciatto@unibo.it

***Abstract*—The increasing maturity of some well-established technologies – such as XPath – along with the sharp rise of brand-new ones – i.e. the blockchain – presents new opportunities to researchers in the field of multi-agent coordination. In this position paper we briefly discuss a few technologies which, once suitably interpreted and integrated, have the potential to impact the very roots of tuple-based coordination as it stems from the archetypal LINDA model.**

I. INTRODUCTION

In spite of their age – the original LINDA model dates back to more than thirty years ago [1] – coordination models have still to reach their full maturity and diffusion in real-world applications. This is mostly due to their early emergence in the literature: open distributed systems, where coordination is a key issue, have become common practice in the software engineering area only in the last decade [2]. Nevertheless, *coordination models and technologies* are to become essential in forthcoming scenarios such as the *Internet of Things* (IoT) [3], as well as in applications where the need for *pervasive intelligence* [4] can be addressed effectively by multi-agent systems (MAS) [5].

Thus, the next decade is likely to determine the extent to which coordination technologies may impact on real applications. Their success is going to depend, on the one hand, on the *expressiveness* of coordination models, since the complexity of application scenarios is growing steadily; on the other hand, on the ability of coordination middleware to *integrate* and exploit the novel *technologies* emerging from the newest application areas, since that would promote industry adoption.

Accordingly, in this position paper we speculate on the opportunities provided by a few technologies – some well-established, some brand-new – once they are integrated within the most expressive and developed coordination models, that is, *tuple-based* ones [6]. Thus, in the remainder of this paper we first (Section II) briefly introduce the enabling technologies – namely, *XPath*, the *blockchain*, and *stream processing* – then (Section III) we foresee their impact on tuple-based coordination models and technologies, finally we provide the reader with some conclusive remarks along with an outlook on future research (Section IV).

II. ENABLING TECHNOLOGIES

Our elaboration on the novel opportunities for tuple-based coordination models takes its move from three promising technologies: *XPath*, a well-established technology whose potential impact on *observability* of tuples and *pattern matching* (Subsection II-A) has been largely overlooked; the *blockchain*, which has recently got momentum amongst both industry practitioners and research teams in academia (Subsection II-B); and *stream processing*, which has been around for a while but is now more affordable and pervasive than ever before, especially if *in-stream* processing is concerned (Subsection II-C).

A. XPath

XPath [7] is a major element in the XSLT standard, aimed at enabling easy navigation through elements and attributes of an XML document.

XPath provides developers with *path expressions*, making it possible to select *nodes* (or sets of nodes) in an XML document: element, attribute, text, namespace, processing-instruction, comment, and document nodes, organised in a tree-like structure—whose root is, indeed, the XML document root element. Nodes have *relationships* – namely, parent, children, siblings, ancestors, and descendants – that XPath allows to be easily navigated by exploiting path expressions, which are combinations of the *name* of an XML node/attribute, *selection symbols* (i.e. `//` enabling to match nodes wherever they are in current XML document), *predicates* (i.e. `position()` enabling to precisely match nodes among siblings), and *wildcard symbols* (i.e. `@*` to match any attribute node). As an example, path expression

```
/bookstore/book[position()<3]
```

matches the first two `book` nodes children of the `bookstore` element, whereas

```
//title[@*]
```

matches all `title` elements which have at least one attribute (regardless of its kind).

XPath path expressions can be even more expressive by exploiting *XPath axes*, which define sets of nodes relative to the current one, such as `ancestor` (to match parents, grandparents, etc. of the current node), `descendant` (to match children, grandchildren, etc.), `following-sibling` to match all the siblings after the current element—and many others[1]. A complete path expression is thus made up of an axis, a node selection instruction, and optionally a predicate for further filtering. As an example, path expression

```
child::*/child::price
```

matches all `price` grandchildren of the current node.

By combining all the above XPath language constructs, many interesting and novel forms of *pattern matching* for tuple-based coordination can be conceived, thus providing a new dimension for *associative access*, a typical feature of such a models [6]. For instance, it is possible to select an XML element (tuple argument) regardless of its location within the XML document (tuple), that is, with no clue on the structure of the XML document itself (i.e. depth and breadth of the tree or a given sub-tree)—as further commented in Section III.

B. Blockchain

The blockchain is a *shared ledger* distributed across a network [8] where transactions are verified against programmed rules of the blockchain itself, and persistently tracked in append-only blocks within the ledger itself. *Permissionless* blockchains are open: any participant can view transactions, even anonymously—as in Bitcoin blockchain [9]. Instead, *permissioned* blockchains allow participants to inspect solely those transactions relevant to them—as in the Hyperledger Project [10].

Transactions are asset transfers onto or off of a ledger, whose blocks are *synchronised* with all other ledgers in the same network. *Consensus* among nodes (copies of the ledger) about transactions verification occurs through programmed rules called *smart contracts*, and ensures that ledgers are exact copies, lowering the risk of fraudulent operations—statistically, because tampering would have to involve many nodes simultaneously. *Smart contracts* are programs directly executing on the blockchain network, which encode the business rules that transactions must abide to so as that all participants can enforce verification consistently. Cryptographic hashes and asymmetric key encryption complete the picture by guaranteeing blocks' integrity and identification of participants, respectively, with no need for a secure communication channel.

Whether it is used within its most famous business domain – that is, supporting cryptocurrencies such as Bitcoin [9] – or in other areas such as supply chain management and medical records tracking [8], the blockchain is explicitly conceived to tackle all the problems of business transactions in open networks, at once. As a result, with the blockchain *(i)* establishing *trust* between parties is no longer necessary, since they only have to trust the technology—the blockchain; *(ii) transparency* comes for free, as the ledger is distributed and all peers involved in a transaction perceive the same state of the blockchain; *(iii) accountability*, too, is naturally supported as for transparency.

As it will be further elaborated in Section III, the blockchain technology could play a fundamental role in expanding the business domains suitable for tuple-based coordination while also improving infrastructural support. For instance, *(i)* nodes synchronisation and consensus may well support implementation of *distributed tuple spaces* – traditionally based on distributed hash tables techniques [11] –; *(ii)* smart contracts may be exploited as a means to consistently enforce global coordination rules [12]; *(iii)* transactions may play the role of tuple space primitives in *privacy/security demanding* application domains.

C. Stream Processing

In its very essence, a *stream processing* engine is just a sort of data processing engine that is designed to deal with *infinite* data sets [13]. Many other definitions exist, typically focussing on specific issues of stream processing, such as whether they process data in (near) real-time or in batch mode, whether they fetch and post data to databases or devices (in the former case they are often called *query engines*), etc. With the aim of being as much inclusive as possible w.r.t. actual streaming technologies – many of which are maintained by the Apache Foundation[2] – we prefer to stick with the more essential definition provided.

Stream processing engines are a *de facto* standard for implementing the *big data infrastructures* underlying many IoT applications [3], where they are often merely exploited as gateway components pushing data to centralised *cloud-based analytics* platforms—thus, they are mostly used in batch processing mode, on finite slices of data. Our focus here is instead on *in-stream* processing, that is, actually, on unbounded streams of data which are continuously processed in order to keep the rest of the system up-to-date with the most recent information.

In-stream processing engines are usually composed by a front-end of fault-tolerant data buffers for storing incoming streams, a back-end of stream filters and processors coordinating to split the computational load and re-assemble results, and a storage layer for both processing state and output data. The front-end takes care of some crucial services such as *stream replay*, that is, the possibility to rollback a stream to fetch older data, possibly already processed, in case of failures of any kind. The back-end of processing nodes is obviously mostly concerned with providing the functionalities regarding data

[1] See https://www.w3schools.com/xml/xpath_axes.asp

[2] To name the most famous ones, Apache Storm (http://storm.apache.org/), Apache Spark (https://spark.apache.org/), and Apache Kafka (https://kafka.apache.org/). A less-known and relatively recent but promising stream processing engine specifically conceived for on-board processing in IoT scenarios worth mentioning is Apache Edgent (http://edgent.apache.org/docs/overview).

aggregation required by the application at hand, but may also be in charge of providing ancillary services such as *lineage tracking*, that is, correlation of streaming and processing events stemming from the same lineage of descendants for monitoring and debugging purpose.

In Section III we discuss how stream processing techniques may fruitfully integrate with tuple-based coordination as well as with the other technologies mentioned in this section. For instance, if a data stream in mapped onto a (set of) XML-tuple(s), continuously updated and observed through XPath queries, a slew of novel opportunities arise. On the one hand, stream processing may bring to tuple-based coordination (near) real-time efficiency; on the other hand, it may benefit of tuple spaces as a form of buffering for stream replay, and of blockchain transactions and smart contracts to guarantee lineage tracking.

III. Impact on Tuple-based Coordination

Tuple-based coordination constitutes the most relevant and suitable class of models and technologies for managing the *interaction space* in complex software systems such as mas and pervasive intelligent applications [14]. In the remainder of this section, we foresee the potential impact of the technologies briefly presented in Section II on the coordination of complex software systems via tuple spaces.

A. Novel Application Domains

Privacy demanding application domains, for instance and most notably *healthcare*-related ones, may in principle be troublesome for tuple-based coordination, mostly because the shared space approach requires *extra-linguistic* – usually, infrastructural – means to address the issue of "*who* may get access to *what*" [15]—typically, the infrastructure is patched with some support to *role-based access control* [16].

Here, the kind of *partial* and *obscure* pattern matching enabled by XPath queries comes handy to prevent interacting agents to *discover* the whole structure of an information item (i.e. an XML tuple representing a medical record from a patient's personal health folder) from a portion of it. In fact, on the one hand the interacting agents are not required to know in advance the *whole* tuple structure in order to be able to perform queries – as in LINDA [1] inspired models typically is – while, on the other hand, the behaviour of the tuple space storing the information may be programmed to return only the *most specific* piece of information which satisfies the query—i.e. the list of the matching nodes without their position (aka path) within the XML tuple (aka document). In this way, privacy of sensitive information is straightforwardly preserved by linguistic means—possibly complemented by the usual extra-linguistic ones.

The blockchain technology further strengthens suitability of the novel tuple-based models here envisioned in the aforementioned business domains by adding features such as *accountability*, *traceability*, and *transparency* of interactions. In fact, asymmetric encryption (to authenticate interacting agents), transactions (as tuple space primitives) along with smart contracts (as coordination laws), and the shared ledger concept itself (as the shared tuple space) straightforwardly enable, respectively, to: *(i)* univocally and securely identify *who* is interacting with *whom* (accountability); *(ii) consistently* manage distributed interactions and enforce coordination laws while seamlessly supporting dynamic construction of *interaction traces* (traceability); *(iii)* greatly enhancing *observability* of the whole coordination process (transparency). For instance, a blockchain-based distributed tuple space may conveniently track medical records exchange and manipulation. In fact, it would be naturally capable of relating interaction events through transactions, providing for free a sort of book-keeping of the *history of interactions* occurred within the system – who accessed what, due to which previous events, leading to which state of affairs – which is an invaluable information for, i.e., accountability purpose.

Finally, stream processing could extend the reach of tuple-based coordination towards IoT-related scenarios, where, on the one hand, the *pace* at which data and events are generated demands indeed for stream processing techniques, while, on the other hand, the amount of *loosely-coupled interactions* among distributed components that need to be fruitfully governed according to a system goal requires decentralised coordination mechanisms. There, for instance, sensor and actuator devices may produce and consume, respectively, streams of data in the form of tuples, and the role of streaming buffers may be efficiently played by tuple spaces. Then, stream filtering and processing components may coordinate to self-organise in pipelines by relying on tuple-based coordination to split incoming data and aggregate partial results. Moreover, if also the blockchain enters the picture, lineage tracking would straightforwardly be supported by transactions and smart contracts, and fault tolerance would be greatly improved thanks to blockchain nodes' synchronisation.

B. Increased Expressiveness

Adopting XPath as the pattern matching language in a XML-tuples setting would straightforwardly enable novel forms of *partial* and *obscure* pattern matching: not only the actual information content may be not known to the agent looking for matching tuples through a suitable tuple template – as it typically stems from LINDA's associative access – but also the overall structure of the information chunks possibly matching is unknown – not the position within a tuple, nor its depth in case of nesting tuples – and even more it cannot be discovered as a side-effect of pattern matching itself—namely, not the whole tuple is returned, but only a suitable portion.

Such a novel form of matching mechanism would enable fine-grained tuning of *observability* of information – and interactions – directly at the language level. Furthermore, the blockchain could *complement* this feature by offering the required *infrastructural support*, also expanding observability beyond tuples, towards interactions and coordination rules.

In fact, the blockchain straightforwardly supports a form of *event correlation*, by ensuring that distributed transactions are *totally ordered* and consistent. Then, if transactions

represent interactions, event correlation actually translates to coordination of interactions. The policies according to which such a coordination process would occur are actually the smart contracts enforced by blockchain's nodes. Since they are programs, expressiveness would naturally lean towards the *Turing-equivalent* expressiveness of coordination languages such as ReSpecT [17].

Then, in-stream processing techniques bring along novel forms of (near) real-time event correlation: in particular, *lineage tracking* in a tuple-based coordination setting may enable to build not only interaction histories, but also *causal relations* between interaction events occurring in the same "interaction pipeline"—intended, for instance, as the sequence of interactions where different components get access to the same data, or perform the same operations on different data.

IV. Conclusion & Outlook

The future of software systems is dealing with complexity, and complexity in software systems typically comes from interaction [18]. Coordination models and technologies are naturally born to deal with that sort of complexity, by providing software engineers with the means for harnessing the space of interaction [19].

Novel application scenarios such as pervasive intelligence and the IoT mandate for new abstractions and tools for agent coordination, so as to properly adapt to the specific application requirements. In this position paper we explore the perspectives of integrating a few novel technologies within tuple-based coordination models – namely, XPath, the blockchain, and stream processing – and provide some insight of their potential impact on multi-agent coordination.

In the next step of our research agenda we will likely proceed either with the integration of the aforementioned technologies within an existing coordination framework, presumably TuCSoN [20], or with the conception and design of a brand-new model and lightweight coordination middleware featuring XML tuples, XPath-based coordination primitives, a blockchain-based backbone implementing a distributed tuple space, smart contracts and transactions as coordination rules, and stream-processing oriented interaction paradigm for the coordinating agents.

Acknowledgment

This work was partially supported by the CONNECARE (Personalised Connected Care for Complex Chronic Patients) project (EU H2020-RIA, Contract No. 689802).

References

[1] D. Gelernter, "Generative communication in Linda," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, 1985. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2433

[2] A. Omicini, "Nature-inspired coordination for complex distributed systems," in *Intelligent Distributed Computing VI*, ser. Studies in Computational Intelligence, G. Fortino, C. Bădică, M. Malgeri, and R. Unland, Eds., vol. 446. Springer, 2013, pp. 1–6, 6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, 24-26 Sep. 2012. Proceedings. Invited paper. [Online]. Available: http://link.springer.com/10.1007/978-3-642-32524-3_1

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[4] R. Calegari, E. Denti, S. Mariani, and A. Omicini, "Logic Programming as a Service (LPaaS): Intelligence for the IoT," in *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC 2017)*, G. Fortino, M. Zhou, Z. Lukszo, A. V. Vasilakos, F. Basile, C. Palau, A. Liotta, M. P. Fanti, A. Guerrieri, and A. Vinci, Eds. IEEE, May 2017.

[5] A. Omicini and S. Mariani, "Agents & multiagent systems: En route towards complex intelligent systems," *Intelligenza Artificiale*, vol. 7, no. 2, pp. 153—164, Nov. 2013, Special Issue Celebrating 25 years of the Italian Association for Artificial Intelligence. [Online]. Available: http://content.iospress.com/articles/intelligenza-artificiale/ia056

[6] D. Rossi, G. Cabri, and E. Denti, "Tuple-based technologies for coordination," in *Coordination of Internet Agents: Models, Technologies, and Applications*, A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, Eds. Springer, Jan. 2001, ch. 4, pp. 83–109. [Online]. Available: http://link.springer.com/10.1007/978-3-662-04401-8_4

[7] "XML Path Language (XPath) 3.1," W3C. [Online]. Available: http://www.w3.org/TR/xpath-3/

[8] S. Underwood, "Blockchain beyond Bitcoin," *Communications of the ACM*, vol. 59, no. 11, pp. 15–17, Nov. 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3013530.2994581

[9] "Bitcoin." [Online]. Available: http://bitcoin.org/

[10] "Hyperledger," The Linux Foundation. [Online]. Available: http://www.hyperledger.org/

[11] Y. Jiang, G. Xue, Z. Jia, and J. You, "DTuples: A distributed hash table based tuple space service for distributed coordination," in *5th International Conference on Grid and Cooperative Computing (GCC 2006)*, Oct. 2006, pp. 101–106. [Online]. Available: http://ieeexplore.ieee.org/document/4031440/

[12] P. Ciancarini, "Coordination models and languages as software integrators," *ACM Computing Surveys*, vol. 28, no. 2, pp. 300–302, Jun. 1996. [Online]. Available: http://dl.acm.org/citation.cfm?doid=234732

[13] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015. [Online]. Available: http://www.manning.com/books/big-data

[14] A. Omicini and M. Viroli, "Coordination models and languages: From parallel computing to self-organisation," *The Knowledge Engineering Review*, vol. 26, no. 1, pp. 53–59, Mar. 2011, Special Issue 01 (25th Anniversary Issue). [Online]. Available: http://journals.cambridge.org/abstract_S026988891000041X

[15] N. Busi, R. Gorrieri, R. Lucchi, , and G. Zavattaro, "Secspaces: a data-driven coordination model for environments open to untrusted agent," *Electronic Notes in Theoretical Computer Science*, vol. 68, no. 3, pp. 310 – 327, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066105803755

[16] M. Viroli, A. Omicini, and A. Ricci, "Infrastructure for RBAC-MAS: An approach based on Agent Coordination Contexts," *Applied Artificial Intelligence*, vol. 21, no. 4–5, pp. 443–467, Apr. 2007. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/08839510701253674

[17] A. Omicini, "Formal ReSpecT in the A&A perspective," *Electronic Notes in Theoretical Computer Science*, vol. 175, no. 2, pp. 97–117, Jun. 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066107003519

[18] P. Wegner, "Why interaction is more powerful than algorithms," *Communications of the ACM*, vol. 40, no. 5, pp. 80–91, May 1997. [Online]. Available: http://portal.acm.org/citation.cfm?id=253801

[19] ——, "Coordination as constrained interaction," in *Coordination Languages and Models. First International Conference, COORDINATION '96 Cesena, Italy, April 15–17, 1996. Proceedings*, ser. Lecture Notes in Computer Science, P. Ciancarini and C. Hankin, Eds. Springer Berlin Heidelberg, Apr. 1996, vol. 1061, pp. 28–33. [Online]. Available: http://link.springer.com/10.1007/3-540-61052-9_37

[20] A. Omicini and F. Zambonelli, "Coordination for Internet application development," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 3, pp. 251–269, Sep. 1999, Special Issue: Coordination Mechanisms for Web Agents. [Online]. Available: http://link.springer.com/10.1023/A:1010060322135