Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 999–1005

System Design and Implementation Decisions for ParaMoise Organisational Model

Mateusz Guzek
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
6, rue R. Coudenhove-Kalergi
Luxembourg, Luxembourg
Email: mateusz.guzek@uni.lu

Grégoire Danoy
Computer Science and Communications Research Unit
University of Luxembourg
6, rue R. Coudenhove-Kalergi
Luxembourg, Luxembourg
Email: gregoire.danoy@uni.lu

Pascal Bouvry
Computer Science and Communications Research Unit
University of Luxembourg
6, rue R. Coudenhove-Kalergi
Luxembourg, Luxembourg
Email: pascal.bouvry@uni.lu

***Abstract*—ParaMoise is a novel organisational model that permits to specify parallel and concurrent systems' organisation and reorganisation. Workflows, locks and multiple organisation managers are the entities that differentiate this model from it antecedent, the $\mathcal{M}$oise$^+$ framework. All these entities must be efficiently designed and implemented to ensure the practical usage of the theoretically formulated model. The main challenge here is the distributed synchronisation of workflows and locks, that will maximise the performance of the system. This paper presents and analyses different workflows and locks management approaches that can be used to achieve this goal: from basic centralised or middleware based solutions, towards truly decentralised coordination mechanisms.**

I. Introduction

ORGANISATIONAL models are used in Multi-Agent Systems (MAS) to facilitate the teamwork between agents. They define the coordination and cooperation mechanisms between agents, resulting in a model that can be reused in various systems and environments, without need to create a custom solution for each of them. Additionally, a model of organisation enables to explicitly represent the social aspects of a MAS, which can be useful for both agents and external observers of the MAS.

Multiple organisational models were introduced [1]–[3], each of them with its own properties and assumptions about MAS architecture. The cited frameworks are notable, as they are general purpose and can fit into multiple domains that benefit from the MAS paradigm.

In this study, we further extend the MOISE [1] organisational model, which relies on a three dimensional description of the Organisation Specification (OS). The OS consists of a Structural Specification (SS) that describes roles together with their hierarchy and possible interactions, groups, and links between roles and groups, a Functional Specification (FS) that describes the schemas, further divided into goals, which are in turn grouped into missions, and a Deontic Specification (DS) that binds the SS and FS by a set of deontic modalities, which enforce or allow agents playing specific roles to commit to missions.

Moreover, MOISE clearly divides the general description of the organisation (OS) from the instantiation of that description, called Organisational Entity (OE). An OE consists of an OS, a set of agents, and the elements that create a valid instance of the OS using this set of agents, e.g. functions that determine the current assignment of agents to roles, groups, and missions, the set of currently existing groups, the set of applied deontic modalities.

The next step in the development of organisational models is considering the benefit and the cost of running an explicit organisation infrastructure in a system. The rationale for running an organisation is to facilitate reaching the desired states by the system. In case of a dynamic environment, it is likely that the organisation may decrease its efficiency due to changes in its environment. As a result, the *reorganisation* may be necessary to adapt the system [4]. The urge for the reorganisation can be especially important for large scale distributed systems, that may trigger reorganisation not only because of external environmental changes, but also due to internal events in MAS. Additionally, the concept of *artifact*, a general and abstract representation of object that can be perceived and used by agents, may be applied to represent organisation [5].

The state-of-the-art development in the field of modelling is ParaMoise [6], that enhances its predecessors by introducing novel concepts coming from the distributed and parallel computing field: workflows, locks, alternative or redundant execution paths, transactions, and failure handling mechanisms, as well as multiple managers of organisation. In effect, the resulting model offers more possibilities to execute parallel and concurrently, without removing or diminishing any of the properties of the antecedent models. The final goal of this development is improving the distribution properties of a MAS, which shall result in an increased performance and reliability, which are essential for dynamic, large scale systems.

However, ParaMoise is a theoretical approach, which application and performance will depend on a proper design and implementation of the proposed mechanisms. In this work, we aim to address this issue by discussing possible alternative designs. In this context, our contributions are the proposals of various design and implementation possibilities for the ParaMoise model divided into two groups:

978-83-60810-53-8/$25.00

a) Centralised: classical tools such as databases that supports transactions, which can be possibly seen as artifacts by MAS.

b) Decentralised: artifacts distributed among agents. In this context we consider that the responsible Organisation Manager (OrgManager) could host the organisational artifacts, which can be also delegated to a new auxiliary Organisation Carrier (OrgCarrier) role that sole purpose is to host the organisational artifacts. We also present the possible solutions for preventing deadlocks that can occur in case of multiple locks, followed by some further refinements of organisation to minimise the resulting synchronisation overheads. Finally, we highlight the impact of using distributed algorithms, as they give agents the possibility to choose the organisational artifact type according to their needs.

The rest of the article is organised as follows. Section II provides a state-of-the-art on ParaMoise, artifacts and distributed synchronisation. Section III presents a centralised solution approach that fulfils the basic requirements for the implementation, while section IV discusses the organisation distributions possibilities. Finally, section V describes the advantages of distributed artifacts and section VI concludes the paper.

II. State of the Art

This section is divided into three parts. Section II-A describes in more details the ParaMoise model, then section II-B presents the artifact-based approaches that are important in the discussed design concepts, and finally section II-C describes the basic algorithms that can be used for the distributed concurrency control.

A. ParaMoise

This section describes the main concepts introduced in the ParaMoise [6] organisational model. ParaMoise is a novel organisational model based on the MOISE [1] and $\mathcal{M}$oise$^+$ [7] models. One of the assumptions of Moise models is the full *autonomy* of agents, i.e. the agents decide by themselves what to do and when, given their current deontic situation, which in turn defines possible rewards or penalties for some performed actions. As a result, the system does not need any central scheduler that will assign tasks to agents, contrary to other state-of-the-art solutions such as GPGP/STÆM [3].

The ParaMoise model is based on the state-of-the-art definitions of organisational models [7], [8], and defines an OE as a tuple [6]: $\langle OS, \mathcal{A}, \mathcal{GI}, \mathcal{SI}, \mathcal{O}, sg, ar, am\rangle$, where OS is the organisational specification; $\mathcal{A}$ is the set of agents; $\mathcal{GI}$ is the set of group instances; $\mathcal{SI}$ is the set of social schemes; $\mathcal{O}$ is the set of current deontic modalities; $sg : \mathcal{GI} \rightarrow \mathbb{P}(\mathcal{GI})$ maps each group to its subgroups; $ar : \mathcal{A} \nrightarrow \mathbb{P}(\mathcal{R} \times \mathcal{GI})$ maps agents to the roles they are playing in the groups; $am : \mathcal{A} \nrightarrow \mathbb{P}(\mathcal{M} \times \mathcal{SI})$ maps agents to the missions they are committed to in the social schemes.

The first major contribution of the ParaMoise model is the *Workflow Specification* (WFS), which is a way to present goals and dependencies between them as a workflow. WFS is defined as [6]: $\langle \mathcal{G}, \mathcal{E}, \mathcal{M}, mo, nm, alt, fh\rangle$, where $\mathcal{G}$ is the set of global goals; $\mathcal{E}$ is the set of precedence relations; $\mathcal{M}$ is the set of mission labels; $mo : \mathcal{M} \rightarrow \mathbb{P}(\mathcal{G})$ is the function that specifies the mission set of goals ; $nm : \mathcal{M} \nrightarrow \mathbb{N} \times \mathbb{N}$ specifies the boundaries (min,max) of number of agents committed to the mission in well formed WFS; $alt : \mathcal{E} \rightarrow \mathbb{P}(\mathcal{E})$ is the function specifies the precedence relations alternatives; $fh : \mathcal{G}_p \nrightarrow \mathbb{N}$ specifies the failure handling mechanism for a primitive goal in terms of maximum number of allowed repetitions. An example of WFS is presented in Figure 1.

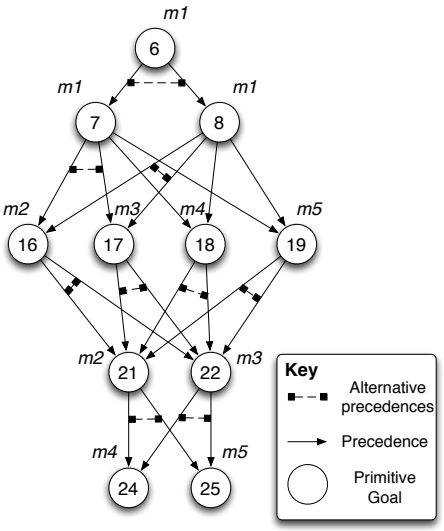

Fig. 1. An example of ParaMoise WFS.

An instantiation of a WFS by some agents is referred to as a *Workflow* (WF). The latter is defined as a tuple [6] $\langle WFS, es, gs, exe, gf\rangle$, where WFS is the workflow specification, $es : \mathcal{E} \rightarrow \{active, inactive, discarded\}$ is the function that maps edges to their activity status label; $gs : \mathcal{G}_p \rightarrow \{waiting, possible, executing, suspended, achieved, discarded\}$ is the function that specifies statuses of primitive goals; $exe : \mathcal{G}_p \nrightarrow \mathbb{P}(\mathcal{A})$ is the function that specifies the set of agents executing a goal; $gf : \mathcal{G}_p \rightarrow \mathbb{N}$ specifies numbers of repetitions of primitive goals. The status of the goals in the system changes according to the state transition diagram presented in Figure 2. The final example of a workflow usage is presented in Figure 3, which presents the capability of tracking the execution status.

The WFS and WF enable more parallelism, since they permit to represent an arbitrary structure of dependencies between goals. They are combined with *locks* to ensure mutual exclusion during reorganisation. The locks are defined as $\langle ROE, type\rangle$, where ROE is the reduced organisational entity (the elements of the OE on which the lock applies) and $type \in \{read, write\}$ specifies the type of the lock. Before a reorganisation, a lock must be created for all modified (write lock) or accessed (read lock) elements of the organisation. To minimise the scope of locks, they can be applied to a subset of

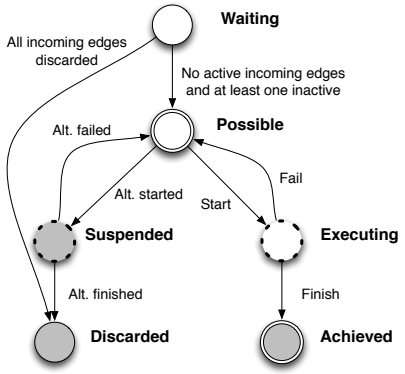

Fig. 2. State transition diagram of goals status in ParaMoise model [6].

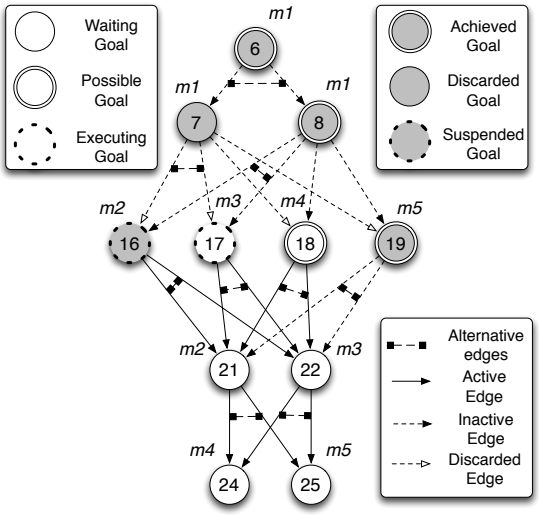

Fig. 3. An example of WF during execution

a set, or for a subdomain of a function. ParaMoise also introduces multiple managers of the organisation (OrgManagers), which fulfil the requirement for an effectively concurrent system.

An efficient access to the workflows and locks is crucial to achieve a high performance of concurrent and parallel execution and reorganisation in a system that applies ParaMoise. Therefore, it is important to find a design appropriate for a considered scenario. In this article we present two approaches: a basic centralised one, applicable for small scale systems, which is interesting as it underlines the basic requirements of the implementation of the model, and the decentralised approach with various possible design choices and their anticipated consequences.

The essential requirements for any implementation of the organisational model are:

1) the existence of all elements of the defined organisation,
2) the accessibility of all the existing elements by the agents with appropriate permissions,
3) the execution of workflow that ensures their correct state transitions,
4) the effective lock and reorganisation mechanisms.

The concept of artifacts can directly meet these requirements.

B. Artifact-Based Frameworks

ORA4MAS (Organisational Artifacts for Multi-Agent Systems) [5] is an approach that describes organisational entities as *artifacts*, based on the Agents and Artifacts (A&A) framework [9]. Artifacts are abstraction of interactive objects that can be perceived and used by agents. An artifact is defined by observable *properties* that represent its state, *operations* that determine its functionality, *links* that describe its relations with other artifacts, *events* that can be emitted in certain conditions, and corresponding *manual* that instructs the agents how to use artifacts. In this context we can see the artifacts as tools that can be used to achieve the goals of agents. ORA4MAS describes the theoretical foundations for using artifacts as the basis of reorganisation, but neglects some system designs aspects. It is a centralised solution based on the paradigms of the A&A framework.

Another drawback of ORA4MAS is the absence of reorganisation. It is partially covered by the JaCaMo [8] framework, which uses a central artifact to perform reorganisation by a single OrgManager that requires halting the whole organisation.

In this context, ParaMoise offers parallel and concurrent reorganisation at runtime, performed by multiple OrgManagers. In the same way ParaMoise enhances standard execution mechanisms, enabling arbitrary precedences between goals, novel possibilities of alternative goals, and a failure handling mechanism. However, the ParaMoise model does not propose any exact design and implementation, but only mentions the usage of artifacts as a perspective. This work proposes to answer to this need. In the remainder of the paper we discuss alternative scenarios, arguing that using well-known and established solutions results in an abundance of choices in which artifacts are one of the basic concepts, being an interface between agents and organisation support systems.

C. Decentralised Synchronisation

Decentralised synchronisation problems are crucial for distributed computing and distributed systems. In contrast to easier case of centralised systems synchronisation, they must be solved taking into account such properties as lack of the global knowledge or communication delays. As a result, a number of solutions to the problem were proposed to solve some main issues for ParaMoise: mutual exclusion (locks) and transactions [10].

1) Mutual Exclusion: Exemplary mutual exclusion algorithms are the Ricart and Agrawala algorithm [11] and token-based algorithms. We focus here on the basic algorithms, despite further refinements were proposed (e.g. Maekawa [12] or Sigma [13]) as they underline the common characteristics of this type of algorithms.

The Ricart and Agrawala algorithm requires communications between all agents, that have the possibility to access the critical section. As a result, the cost of synchronisation is $2(n-1)$, where n is the number of agents. Additionally, in the basic forms, the failure of any of the agents disturbs the proper work of the algorithm. These limitations prohibit usage of such algorithms in a large groups of agents. On the other hand, the algorithm is fair and in optimistic case leads to a fast resolution of the problem.

The token-based algorithms [10] ensure mutual exclusion by using a unique token for a critical section. These can be applicable for a set of distinct critical sections or resources. The token can be passed according to various strategies, e.g. continuously moving in the ring or using more sophisticated hierarchical structures.

2) Transactions: Another aspect of concurrency control is proper transaction handling, which is required for an effective WF management. Three main approaches can be distinguished [10].

Two phase locking (2PL) [14] is based on the standard lock mechanism. The locks are created in the two phases: in the first phase the locks are consecutively acquired according to the needs of a transaction and then in the following phase they are released.

The optimistic concurrency control [15] assumes that violations of mutual exclusion are rare, and it is possible to repair the potential damages done by such violation. As a consequence, there must exist an efficient repair mechanism and the collisions cannot be destructive. This is most effective in the case of relatively rare occurrence of conflicting write operations in a system [16].

The pessimistic timestamp ordering [10] is the last approach presented here. It associates the demands to access elements being part of a critical section with additional read and write timestamps. During the evaluation of the incoming transactions, the timestamps are used to check if the incoming transactions conflict with the ongoing ones. The approach is safer than the optimistic concurrency control, as it avoids the potential problems instead of resolving them.

III. Centralised Design

The centralised solution for synchronisation can be achieved with classical tools used for mutual exclusion and transactions. A central entity is responsible for keeping all the information about the state of the organisation. In case any agent needs to acquire knowledge about any part of the organisation, e.g. the agent's roles, obligations or known agents, it can query this entity.

The concurrent access control is performed centrally and in result does not pose a major challenge. A system of role-based access can effectively enforce that only the entitled agents can access specific elements of the organisation. The transaction mechanism can be straightforwardly applied to the execution of workflows, ensuring the correct state transitions of goals. Finally, lock creation and checking is done by a single entity that can prohibit any forbidden overlaps.

As a result, we can see this entity as a database which stores all elements of the organisation and grants access to them only to the roles that have the required permission. The workflows are stored inside the database and their status can be changed using the mechanism of transactions. The database routines ensure that the created lock does not overlap with other locks.

For an agent in the system, the database is seen as an artifact that stores the information about the organisation with well-defined interfaces to perform organisational actions. It can return information about the organisation or be exploited as a synchronisation tool used for efficient teamwork, as it holds the workflow state. Finally, the artifact has interfaces that enable OrgManagers to change the shape of the organisation by modifying the current state of the organisation in a safe way.

IV. Decentralised Design

This section presents the variety of possible design choices for the ParaMoise model and discusses their properties. Firstly, it describes the basic decentralisation capabilities and concepts in section IV-A, which introduces the decentralised artifacts described in section IV-B. Artifacts management problems and the corresponding organisational challenges are presented in section IV-C, while the solving of possible deadlock problems is discussed in section IV-D.

A. Decentralised Middleware

The most straightforward way to decentralise the system is to use an existing solution to distribute the centralised middleware, e.g. a database. This involves correct replication schema together with synchronisation of replicas. From the MAS design perspective these problems of distributed computing are out of the scope of this paper, as logically there is still one entity that is distributed, possibly with multiple equivalent interfaces. Therefore, the following paragraphs describes the applicability of decentralised synchronisation algorithms for the ParaMoise model. Following the structure of Section II-C, we describe two main issues: Mutual Exclusion and Transactions.

1) Mutual Exclusion: Solving the mutual exclusion problem is an essential design decision for ParaMoise, as it effectively determines the locks mechanism, its performance and properties. The Ricart and Agrawala algorithm is applicable for the ParaMoise model. The need for broadcast communication may rise scalability issues, however proper division schemas may result in more applicable solutions, which are discussed further in Section IV-D. The other discussed solution, token-based algorithms, seems to have limited applicability in the ParaMoise model. The lock in ParaMoise could have an arbitrary form, which makes the token impractical. There could be either a large number of tokens that could create significant overhead, or in the opposite case a small amount of token responsible for major organisational elements would decrease the possible number of concurrent reorganisations. Additionally, gathering multiple

tokens to perform reorganisation may lead to increased waiting time and deadlocks.

2) Transactions: The transactions mechanism is necessary to properly modify the status of goals during progress of WF, which is the core issue for the efficiency of the parallel and concurrent execution in a ParaMoise system. A WF is also used to express all the schemas in the organisation, including reorganisation processes. 2PL conceptually fits the ParaMoise locks and could be directly applied. In the case of using optimistic concurrency control for reorganisation, there is a need to ensure that the occurrence of a conflict will not result in breaking the consistency, by applying an effective rollback mechanism. The property of allowing the conflicts to happen is not acceptable in MAS organisations, as achieving some of the goals by agents can be impossible to undo, making the rollback impossible. Additionally, the changes of reorganisation shall be directly mapped into the behaviour of agents, which could lead to an inconsistent state in an organisation. Pessimistic timestamp ordering is applicable thanks to its more cautious nature. As it ensures that the organisation will be unique at any moment in time, it is applicable for usage in ParaMoise. The specifics of the solution require a mechanism for comparing the timestamps, which could be directly performed by an artifact created for each WF. We discuss more generally the usage of decentralised artifacts in the following section.

B. Decentralised Artifacts

The next step toward decentralised system is a logical distribution of artifacts among the distributed system. In this way, the agents are no longer using the monolithic organisational artifact, but they access a set of distinct artifacts. An example of such a division may be a system that stores the definition of each role as a separate artifact.

The logical division can lead to practical consequences: as the elements of the organisation are separated among artifacts, locks are distributed among the artifacts, therefore decreasing the complexity of checking for possible conflicts. Additionally, this approach eliminates performance bottleneck, and decreases the risk of single point of failure. On the other hand, larger reorganisations can require interactions with several artifacts, increasing therefore the complexity of the operation, leading to the possibility of deadlocks, and in case of lack of redundancy failure of any of the artifacts can lead to breaking down the whole organisation.

C. Role-Based Synchronisation

The concept of decentralised artifacts can be further advanced by merging it with the concept of roles. Agents of specific roles would be responsible for maintaing organisational artifacts. The responsibility may hold for the whole system, or for a specific group. A natural choice for the role responsible for artifacts is the OrgManager, however this solution would add another functionality to this role. As OrgManagers are already responsible for coordinating organisation and executing reorganisation, we propose a new role in the structural specification: Organisation Carrier (OrgCarrier).

The OrgCarrier responsibility is to maintain the organisational artifacts. To keep the control over parts of the organisation, OrgManagers have authority over OrgCarriers, i.e. the latter should follow the orders of the former. We present the novel structural specification in Figure 4. The authority link of the OrgManager pointing to the root role *soc* is transitively propagated to the OrgCarrier role. However, this is the only connection of OrgCarrier, since none of *Org* roles, except OrgManager, has knowledge about the OrgCarrier. This structure can fulfil its goals, being transparent for the rest of the system.

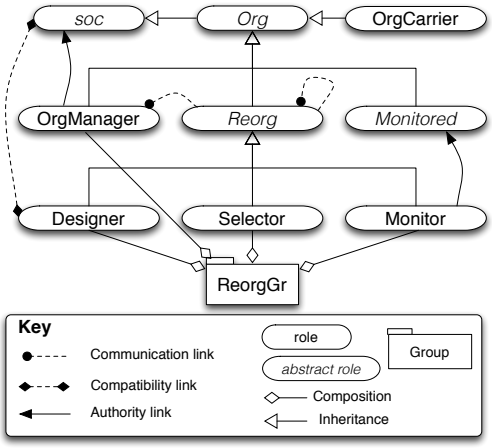

Fig. 4. Organization group structure with OrgCarrier role.

D. Resolving Deadlocks

As previously mentioned, introducing multiple locks distributed among artifacts leads to the possibility of deadlocks. An example leading to a deadlock is an attempt to create locks on two organisational elements by two OrgManagers. If they actions are synchronised, but they create locks in reverse order, a deadlock occurs. Such behaviour may be simply overcome by adding a timeout for each of the locks, however this may lead to poor system performance (waiting for timeout) or aborting lock for a valid operation that lasts longer than expected [16].

Another approach to solve the problem of deadlock could be the coordination between OrgManagers, using an algorithm such as Ricart and Agrawala. Each OrgManager broadcasts the description of the lock it wants to create together with a timestamp. Other OrgManagers must reply that they allow to create the lock. In case of conflict, the OrgManager that detects it takes a decision based on the timestamps. The lock with earlier timestamp has priority. Tie braking may be implemented, e.g. favouring the lowest agent ID number. The inherent drawback of this solution is its scalability: each agent must receive and send a message coming from the initiator of the procedure. Additionally, unreliable channels or agents that occur in dynamic systems may break this schema,

with the probability of failure growing with the number of OrgManagers.

The organisational model properties can be used to mitigate such negative behaviours. The responsibility of a subset of OrgManagers can be restricted to specific elements of the organisation. As an example, OrgManagers could form groups associated with a role. Additionally, to ensure that major reorganisation spanning across multiple roles can be executed, there must exist a subset of OrgManagers responsible for the whole organisation. As a result, each lock concerning an element of the organisation must be checked by the subset of agents (i.e. in a group) that contain the OrgManagers responsible for this element as well as the globally responsible OrgManagers. The effectivity of such solution is based on the assumption that the major reorganisations are relatively rare. Additionally, the division of elements of the organisation must be done with a granularity that ensures correct system behaviours.

V. Discussion

A. Advantages of Artifact Driven Organisation

Representing organisations with artifacts has an additional added value: agents are controlling the artifacts, thus they have the power to alter even the organisation implementation. For example, agents may initially choose to use the centralised monolithic artifact while the system is of small scale. However, together with the growth of the system agents may decide that this solution has reached its limits and shall be changed to better scale with the new situation. Then, by correct mapping of the existing organisation to new artifacts agents can replicate the existing OE and start to use the new type of artifacts.

Agents can learn how to use different organisational artifacts, what are their strong and weak points in terms of performance, reliability, recoverability, etc. Moreover, this solution can be used to perform updates, maintenance or archive the organisational artifacts. In case one type of artifacts starts to present erroneous behaviour, agents have possibility to choose another one.

VI. Conclusion

The ParaMoise model can be used using various designs and implementations. From the discussed alternatives, we see the Ricart and Agrawala family of algorithms, 2PL, and pessimistic timestamp ordering as the most fitting low-level primitives. We consider that artifacts could play a major role as interfaces between agents and systems that agents use. Artifacts are easy to distribute, can embed specific access control and synchronisation mechanisms, and they enhance the autonomy of agents. The paper also introduces the OrgCarrier role that can facilitate the management of the organisation. The properties of decentralised algorithms may require additional structure of the OrgManagers, for example by adding managers responsibility zones.

The future work includes experimental testing of the discussed solutions as well as implementing ParaMoise as a general purpose framework. We intend to use ParaMoise in a system optimising and managing Cloud Computing infrastructures, which could validate the approach. In this context, the optimisation of an organisation model to achieve the system objectives is a prospective research direction.

Acknowledgment

M. Guzek acknowledges the support of the National Research Fund of Luxembourg (FNR) and Tri-ICT, with the AFR contract no. 1315254. This work was completed with the support of the FNR INTER-CNRS-11-03 Green@cloud.

References

[1] J. Hübner, J. Sichman, and O. Boissier, "A model for the structural, functional, and deontic specification of organizations in multiagent systems," in *Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, G. Bittencourt and G. Ramalho, Eds. Springer Berlin / Heidelberg, 2002, vol. 2507, pp. 439–448.

[2] V. Dignum, "A model for organizational interaction: based on agents, founded in logic," Ph.D. dissertation, Proefschrift Universiteit Utrecht, 2003.

[3] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. N. Prasad, A. Raja, R. Vincent, P. Xuan, and X. Q. Zhang, "Evolution of the gpgp/tæms domain-independent coordination framework," *Autonomous Agents and Multi-Agent Systems*, vol. 9, pp. 87–143, 2004.

[4] J. Hübner, J. Sichman, and O. Boissier, "Using the $\mathcal{M}$oise$^+$ model for a cooperative framework of mas reorganisation," in *Advances in Artificial Intelligence, SBIA 2004*, ser. Lecture Notes in Computer Science, A. Bazzan and S. Labidi, Eds. Springer Berlin / Heidelberg, 2004, vol. 3171, pp. 481–517.

[5] J. Hübner, O. Boissier, R. Kitio, and A. Ricci, "Instrumenting multi-agent organisations with organisational artifacts and agents," *Autonomous Agents and Multi-Agent Systems*, vol. 20, pp. 369–400, 2010.

[6] M. Guzek, G. Danoy, and P. Bouvry, "Paramoise: Increasing capabilities of parallel execution and reorganization in an organizational model," in *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS'13*. IFAAMAS, May 2013, pp. 1029–1036.

[7] J. F. Hübner, J. S. Sichman, and O. Boissier, "Developing organised multi-agent systems using the $\mathcal{M}$oise$^+$ model: Programming issues at the system and agent levels," *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 3/4, pp. 370–395, 2007.

[8] A. Sorici, G. Picard, O. Boissier, A. Santi, and J. F. Hübner, "Multi-Agent Oriented Reorganisation within the JaCaMo infrastructure," in *Proceedings of The Third International Workshop on Iinfraestructures and tools for multiagent systems: ITMAS 2012*, Valencia, Espagne, 2012, pp. 135–148.

[9] A. Ricci, M. Viroli, and A. Omicini, "The A&A programming model and technology for developing agent environments in MAS," in *Programming Multi-Agent Systems*, ser. LNCS, M. Dastani, A. El Fallah Seghrouchni, A. Ricci, and M. Winikoff, Eds. Springer, Apr. 2008, vol. 4908, pp. 89–106, 5th InternationalWorkshop (ProMAS 2007), Honolulu, HI, USA, 15 May 2007. Revised and Invited Papers. [Online]. Available: http://www.springerlink.com/content/92370q174328841j/

[10] A. S. Tanenbaum and M. V. Steen, *Distributed Systems: Principles and Paradigms*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[11] G. Ricart and A. K. Agrawala, "An optimal algorithm for mutual exclusion in computer networks," *Commun. ACM*, vol. 24, no. 1, pp. 9–17, Jan. 1981. [Online]. Available: http://doi.acm.org/10.1145/358527.358537

[12] M. Maekawa, "A n algorithm for mutual exclusion in decentralized systems," *ACM Trans. Comput. Syst.*, vol. 3, no. 2, pp. 145–159, May 1985. [Online]. Available: http://doi.acm.org.proxy.bnl.lu/10.1145/214438.214445

[13] W. Chen, S. Lin, Q. Lian, and Z. Zhang, "Sigma: A fault-tolerant mutual exclusion algorithm in dynamic distributed systems subject to process crashes and memory losses," in *Proceedings of the 11th Pacific Rim International Symposium on Dependable Computing*, ser. PRDC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 7–14. [Online]. Available: http://dx.doi.org/10.1109/PRDC.2005.57

[14] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[15] H. T. Kung and J. T. Robinson, "On optimistic methods for concurrency control," *ACM Trans. Database Syst.*, vol. 6, no. 2, pp. 213–226, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/319566.319567

[16] Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.