A Framework for Systemic Coordination in Open Computational Systems

Martin Fredriksson Alessandro Ricci
Andrea Omicini Rune Gustavsson

***Abstract*—The material presented in this paper is focused on the characteristics of open computational systems and, in particular, the issue of coordinating behavior in such systems, i.e. *systemic coordination*. Firstly, we introduce a conceptual framework for characterizing systemic coordination at different levels abstraction. Secondly, challenges and opportunities of this framework is outlined as a matter of exploiting a *Service-oriented layered architecture for communicating entities* (SOLACE) in combination with a particular coordination model (TuCSoN). In the context of systems engineering, we consider SOLACE and TuCSoN to be examples of the basic tools required for support of systemic coordination, i.e., the continuous process of construction and observation of open computational systems and their evolving interaction space. Finally, we outline the impact of these tools on empirical aspects of systemic coordination.**

I. INTRODUCTION

The material presented herein is focused on coordination of behavior in *open computational systems* [8], i.e. physical environments where multiple agents have the capability to access and interact with a networked and computationally empowered system. The general vision of the future considers the involved networks of devices to be interconnected with each other in an *autonomic* and *conjunctive* manner [28]. A device is simply powered on and automatically integrated into the surrounding fabric of interaction, either by means of a physical connection or by bringing the device into the proximity of a wireless network. Consequently, we envision future multiagent systems to rely on such a conjunctive fabric of interaction, populated by personally tailored conjunctive services. As such, the notion of open computational systems explicitly accounts for ambient and autonomic computing [11], [14], i.e. real world application scenarios in the context of ubiquitous and pervasive computing [34], proactive computing [28], embedded Internet [6], network-centric Computing [13], [30], grid computing [1], and service-oriented computing [15].

From a general perspective, open computational systems can be conceived as communities of individuals, characterized by a shared overall *mission* that can be achieved by means of support from information systems [12]. Consequently, *coordination* is a fundamental aspect of open computational systems, accounting for the *global coherence* of system behaviour, according to a particular mission and *context* in which the members of some community are situated.

An applicable perspective on the complexity of coordination in open computational systems is provided by the computer-supported cooperative work context (CSCW) [24] and, in particular, in terms of the concept of *articulation of work*. Typically the involved complexity of coordination requires the flexible deployment of multiple *coordination mechanisms* [25], adopting approaches that span from *subjective* — mostly relying on the capabilities of individuals to support coordinated behavior — to *objective* — exploiting coordination services provided by an underlying infrastructure and interaction medium [27].

Given this background on systemic coordination, the material outlined in this paper introduces a conceptual framework for systemic coordination. The framework explicitly addresses a number of abstraction levels that are necessary in identifying current approaches of coordination and putting them into an applicable context of empirical investigation. The framework should be of particular interest in the context of methodologies for agent-oriented software engineering methodology, i.e., the framework aims at bridging the gap between theory and practice of systemic coordination in open computational systems.

In order to emphasize the practical implications of our methodological framework, we introduce a concrete example of systemic coordination by means of the application scenario of *Trustworthy and sustainable operations in marine environments* (TWOSOME). In fact, SOLACE was used in the development of this system and, as it accounts for systemic coordination, is now subject to further investigations of using multiple coordination mechanisms. One example of such an evaluation is to introduce TuCSoN as an additional coordination service in the context of TWOSOME. Throughout this paper, this approach will be extensively discussed, always with the support of our methodological framework, in which we also outline a case study for future exploration.

II. OPEN COMPUTATIONAL SYSTEMS

A. Systemic coordination in practice

Research and development of information systems for defense and warfare have changed most dramatically during the last decade; from weapons of mass destruction to sustainable systems of coordinated and computationally empowered entities, i.e. network-centric warfare. From a holistic perspective, the involved systems are comprised by a wide range of agents and services: sensor and actuator systems, detection

M. Fredriksson and R. Gustavsson are affiliated with the Department of Software Engineering and Computer Science at Blekinge Institute of Technology, Box 520 SE-372 25 Ronneby, Sweden – email: martin.fredriksson@bth.se, rune.gustavsson@bth.se. A. Omicini and A. Ricci are affiliated with DEIS, Università degli Studi di Bologna, via Rasi e Spinelli 176, 47023 Cesena (FC), Italy – email: aomicini@deis.unibo.it, aricci@deis.unibo.it

and weaponry systems, as well as communication, decision, and support systems. To that end, the behavior of each system component, as well as their synthetic behavior, has to be dependable and trustworthy in operations under dynamic and hostile conditions, and — perhaps even more challenging — the notion of system lifespan has to be considered in terms of decades. Consequently, at the core of a prototype system for Trustworthy and sustainable operations in marine environments (TWOSOME) was the development of a multiagent system, where interacting and coordinated entities and services temporarily come together in a physical setting in order to perform a particular assignment under dynamic and hostile conditions. The system developed is subject to a validation of qualities such as information fusion, adaptation, shared awareness and decision making, i.e., coordination in military missions and operations.

The systemic coordination in TWOSOME involves two interrelated application domains — the viewpoints of attackers and defenders — and missions, i.e., creating and removing physical threats at some particular environment location. The various agents involved are positioned in a physical environment and therefore, by means of their autonomous and cooperative behavior, create an evolving state of affairs that is impossible to handle by means of a single coordination mechanism. An attacker — a smart mine — is positioned in the environment and set to detonate whenever the presence of a particular vessel is identified in the surroundings. Correspondingly, three coordinated defenders — autonomous vessels emanating acoustic and magnetic signatures — are assigned the role of sweeping different environment locations and aim at removing potential attackers by means of making mines detonate in a harmless way. The role of a team of defenders is to outsmart the mine by means of providing fake signatures of real vessels. In TWOSOME, the issue of sustainable coordination is explicitly accounted for by the three defenders by means of continuously executing the articulation, construction, observation, and instrumentation feedback loop and thereby maintaining a particular signature at hand. In reality, this involves the continuous adaptation and reconfiguration of roles. In accordance with the layered perspective of open computational systems, the following material introduces an outline of TWOSOMEs characteristic environment, fabric, and system properties that account for the concept of deception in network-centric warfare.

B. Systemic coordination in theory

The most characteristic property of open computational systems can be described as a natural support for construction and observation. In order to explicitly account for the issues of construction and observation of system behavior, we introduce a formal model for construction that is grounded in process algebra, with the operators of join (integration) and connect (interaction). This approach enables us to observe the fundamental constructs involved in sustainable coordination and, consequently, in what way instrumentation of system behavior most appropriately can be supported in terms of continuous articulation, construction, and observation. As such, construction is considered to be a matter of integration and interaction, i.e., the ability of an agent to enter or leave a particular location in some system and the ability of agents in the system to interact with each other, by means of integration.

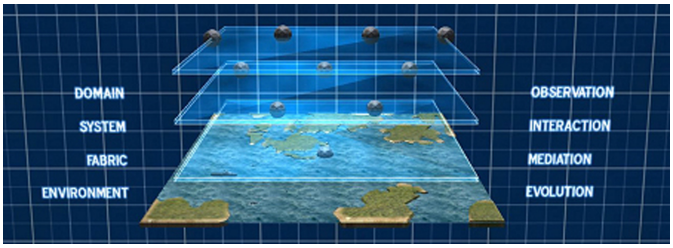

Fig. 1. *Framework.* The systemic coordination of behavior in open computational systems is characterized by means of four distinct layers that exhibit certain key features: evolution, mediation, interaction, and observation.

Coming back to the main issue addressed in this paper, it stands clear that both subjective as well as objective approaches to multiagent systems coordination implicitly assume that the articulation of some particular mission is an integral component of the system in question and, consequently, must not change during its life time. However, as accounted for by TWOSOME, there are indeed situations — i.e., context switches — that require the global coherence of some particular systems mission and quality as a whole to be continuously constructed and observed. Consequently, when some complex multiagent system is situated in an evolving physical environment, it is of fundamental importance that the role of articulation is made explicit. This also implies that multiple coordination mechanisms are required as a matter of the context switching taking place in the particular system under study. In summary, if we consider systemic coordination to be a matter of continuous articulation, construction, observation, and instrumentation of open computational systems, we need to address this type of continuous system adaptation in terms of a comprehensive and methodological approach.

III. FRAMEWORK

The phenomena in focus of this paper is open computational systems. In particular, we aim at bridging the gap between theory and practice of coordination in such systems. However, we believe that the only way to bridge this gap is to develop a common methodological framework. This would enable us to address common challenges and opportunities of systemic coordination of open computational systems. In essence we consider certain characteristic features, classes, and instances of systemic coordination.

A. Environment and evolution

A key aspect of open computational systems is their characteristic property of being open and dynamic in nature. We consider such systems as situated in environments where new agents and services can enter and leave the environment at will. In this environment entities, computational or not, observe and interact with each other. This environment is also the origin of unpredictable events affecting sustainable behaviors and, hence, coordination of any system situated in this context:, agents and services can not only enter and leave a system at will, they can also be forced to change state as a result from

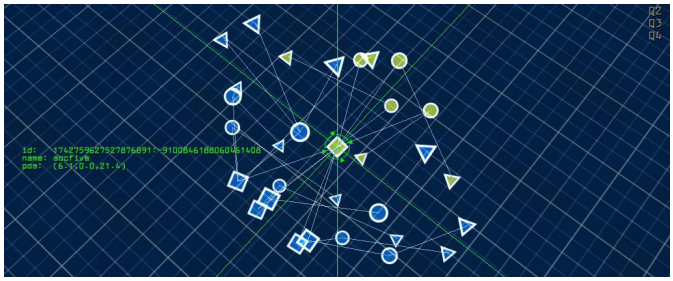

Fig. 2. *Construction*. In SOLACE, empirical studies of open computational systems are supported by means of (multiple) qualitative perspectives, i.e., online models of structures in open computational systems, that can be observed dynamically by DISCERN. The figure shows the DISCERN dynamic representation of some structures at the system level — services and agents — and structures at the domain level — such as concepts.

external stimuli that cannot be predicted at design time. Agents and services can also be forcefully removed from the environment as a result from unknown events. In a sense, an agent that proliferates in an open computational system is by all means subject to a continuous struggle for survival and evolution.

B. Fabric and mediation

The fabric of interaction is an integral part of an open computational systems environment. Any set of agents and coordination artifacts require some form of support for mediation. We consider the fabric of an open computational system to explicitly support the (local) existence of agents and services, as well as the dynamic coupling of a dynamic set of fabric nodes. Each fabric node must support the construction and observation of cognitive constructs as well as the interaction between agents. In summary, the fabric of an open computational system is considered in terms of the artifacts involved as well as their connectivity and required support for mediation, interaction, and observation. However, the fabrics support for mediation must never involve the actual coordination of agents and services. This feature is solely imposed on domain specific constructs at the system and domain levels of open computational systems.

C. System and interaction

An integral part of the interaction fabric, there exist a dynamic set of computational entities. As such, each entity, e.g., agent or service, at the system level exhibit an autonomous behavior by means of observing the surrounding environment and interacts accordingly. In doing so, a computational entity requires the explicit support for mediation provided by the fabric of interaction. Due to the occurrence of frequent context switching in open computational systems, the entities that interact and form some particular behavior also require the capability to dynamically create cognitive constructs — i.e., models — of themselves and their particular behavior. These constructs are used by other entities in the continuous process of articulation, construction, observation, and instrumentation.

D. Domain and observation

From a holistic perspective, the domain layer is one of the most important perspectives of an open computational system. By means of a dynamic set of cognitive constructs — i.e., constructed and instrumented by the computational entities at the system level — the activity of observation is primarily dependent on an agents capability to discover potential entities to interact with in some articulated mission. It is by means of observation of these cognitive constructs that an agent implicitly is capable of measuring characteristic qualities in a system. In essence, the domain layer of open computational systems is supported by each node in the fabric of interaction, but articulated, constructed, observed, and instrumented by the dynamic set of entities residing at the system level.

IV. PRACTICE

Coming back to the main issue addressed in this paper, it stands clear that both subjective as well as objective approaches to multiagent systems coordination implicitly assume that the articulation of some particular mission is an integral component of the system in question and, consequently, must not change during its life time. However, as accounted for by TWOSOME, there are indeed situations i.e., context switches that require the global coherence of some particular systems mission and quality as a whole to be continuously observed and instrumented. Consequently, when some complex multiagent system is situated in an evolving physical environment, it is of fundamental importance that the role of articulation is made explicit. This also implies that multiple coordination mechanisms are required as a matter of the context switching taking place in the particular system under study. In summary, if we consider sustainable coordination to be a matter of continuous articulation, construction, observation, and instrumentation of open computational systems, we need to address this type of continuous system adaptation in terms of a comprehensive and methodological approach.

A. SOLACE *for construction*

We have developed a distributed service-oriented layered architecture for communicating entities (SOLACE), that supports the construction of open computational systems. This architecture explicitly supports the three abstraction layers of fabric, system, and domains. Support of the environment perspective has been omitted in Solace because the architecture as such is only supposed to support the abstract perspectives characterizing constructs that are an integral part of the physical environment. That is, we consider the architecture to be an integral part of the physical environment and not vice versa. Our architecture therefore supports the layered perspective of open computational systems in terms of fabric, system, and domain. In particular, both the system and domain level, providing means for entities located at the system level to observe and constructs structures at the domain level, with a clear *separation of concerns*: a physical instantiation of a service or agent at the system level can dynamically define its cognitive manifestations as structures and relationships at the domain level, in order to be discovered and observed by the other entities populating the system level.

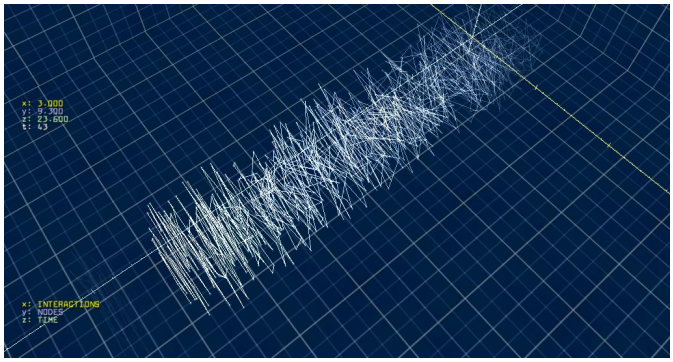

Fig. 3. *Observation.* In DISCERN, empirical studies of open computational systems are supported by means of (multiple) quantitative perspectives, i.e., online models of patterns in open computational systems. A specific observation must necessarily encompass important quality dimensions as well as their quantification.

B. DISCERN *for observation*

In practice, the activity of observation aims at acquiring certain quantifications of some domain's characteristic qualities. Acquiring these measurements can be done by means of certain instruments. Also, we consider these quantifications to be the basis in an actual instrumentation of some particular system under study. We therefore have implemented a *Distributed interaction system for complex entity relation networks* (DISCERN) that explicitly addresses the dynamic and real time observation and instrumentation of open computational systems residing on SOLACE, e.g., TWOSOME. DISCERN enables the observation of *qualities* and *quantities* in multiple domains of open computational systems [9]:

- *Qualities*. In the context of the layered perspective of qualities provided by the framework, DISCERN is therefore applied in the observation of domain specific qualities with respect to these four perspectives. However,even though SOLACE does not address the notion of a physical environment, this is of utmost importance in DISCERN in order to immediately provide a human agent and observer with the current context. In a similar manner, the fabric, system, and domain layers of open computational systems are represented in DISCERN by means of a volumetric space that is navigable (see Fig. 2).
- *Quantities*. In addition to the possibility of a human agent to observe and instrument the qualities of open computational systems, DISCERN also provides support of quantification. That is, by means of the constructs accounted for by SOLACE, any quality available in some accessible open computational system can dynamically be quantified and further analyzed in DISCERN (see Fig. 3). Examples of the sustainable coordination of such qualities and quantities are currently studied in TWOSOME, in terms of information fusion, adaptation, awareness, and policies. Since DISCERN supports real time observation and instrumentation of open computational systems, the quantities acquired in some particular context can be used in further studies on automated analysis and in the continuous feedback loop of articulation, construction, observation, and instrumentation.

C. TuCSoN *for coordination*

TuCSoN is a model for enabling and supporting coordination in multiagent systems [20]. TuCSoN coordination relies on *tuple centres* [19], as programmable tuple space which are exploited first to enable agent interaction, then to specify and enact agent coordination — as interaction management — through tuple centre behaviour. More specifically, tuple centres are Linda-like tuple spaces [10], enhanced with the notion of behaviour specification, espressed in terms of first-order logic specification language ReSpecT[18]. Each tuple centre is independently programmed through its own ReSpecT specification, defining its behaviour in response to communication events. As a result, a tuple centre encapsulates the laws of coordination in terms of ReSpecT specification, which are both inspectable and dynamically configurable.

TuCSoN promotes the vision of of *coordination as a service*[29], which is they key to understand the deployment of the model in the SOLACE service-oriented context (see Fig. 4 for an overall perspective). From a topological and infrastructure point of view, coordination services are provided by nodes, where tuple centres resides as run time coordination abstractions. Each TuCSoN node defines a coordination context where the coordination services are situated; a coordination context can be conceived as an open set of services enacted through tuple centres, organisational rules (roles, policies, ...) and dynamics that concern a node, its static and dynamic context. The collection of all the coordination contexts provided by the nodes constitute the TuCSoN *coordination space*. In order to access the services, an agent must enter the coordination context provided by the node, and this is where the agent coordination context notion comes into account. Agents can enter the node coordination context by negotiating a TuCSoN *agent coordination context* (ACC) with the node. The ACC is the means that allows the agent to exploit the coordination services provided by a node, providing the set of the possible actions and observation that the agent can do in the context: as for the control room metaphor described in [17], it establishes the admissible agent inputs and the admissible agent outputs. The idea (and related modelling and engineering discipline) is to use the ACC to embed and enforce *subject-centered* rules, i. e. rules constraining agent actions by virtue of its position (role) inside the organisation, its actions in the specific coordination context related to that role, but also as *the* means to model and provide the quality it negotiated for the coordination. Instead, as already described extensively in other papers [19], [4], the tuple centres are used to embed and enforce coordination laws toward the fulfilment of social tasks, the prescription of social norms, so rules and constraints that are more *society/organisation* centered, and that characterise TuCSoN as an objective-coordination model [27]. ACC and tuple centres together provide subjective and objective means for modelling and shaping agent interaction space within the context of an agent society/organisation, ACC with focus on the subjects, tuple centres with focus on the society.

D. Challenges and opportunities

From the TuCSoN perspective, SOLACE can be conceived as one of the service-oriented multiagent service-oriented infrastructure which allows TuCSoN coordination services to be

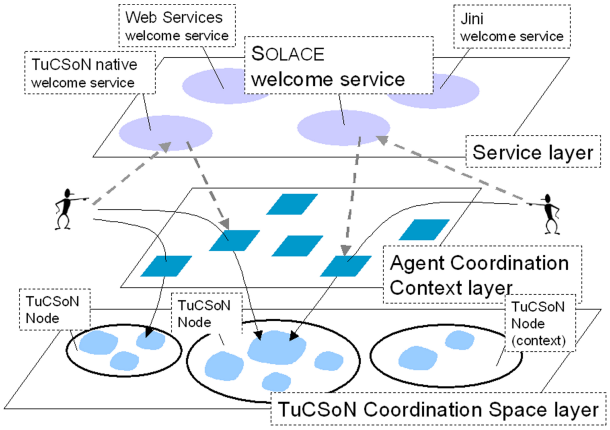

Fig. 4. SOLACE *from* TuCSoN *perspective*. SOLACE is conceived as one of the service-oriented multiagent service-oriented infrastructure which allows TuCSoN coordination services to be discovered and deployed.

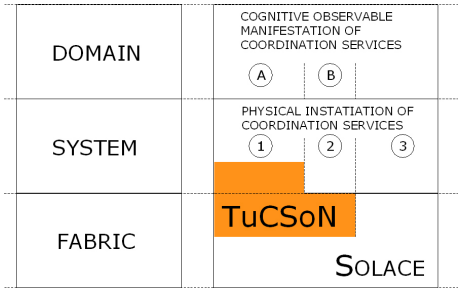

Fig. 5. TuCSoN in the multilayered perspective of the framework and of SOLACE.

discovered and deployed, as depicted in Fig. 4. From the SOLACE perspective, TuCSoN supports the layered perspective of open computational systems with specific focus on the interaction and coordination Fig. 5:

- *Fabric*. The fabric layer is supported by the service-oriented infrastructure that provide TuCSoN services, that could be TuCSoN native infrastructure or any other multiagent system service-oriented infrastructure such as SOLACE (this issue will be described in details in following sessions); In Fig. 5, TuCSoN is partially considered also part the fabric layer, with SOLACE, because from the point of view of coordination, it provides the basic enabling mechanism (in the CSCW perspective [25]) or media / languages (according to objective coordination models and language ontology [2]) to construct high level coordination structures (and processes). The basic objective coordination mechanisms provided have been demonstrated — from a formal point of view — to be expressive enough to support the specification and enactment of any high level coordination pattern [18]; in this context it means that any special-purpose coordination service can be constructed and deployed at the system level, and manifested and observed at the domain level;
- *System*. The system layer is supported by physical instantiation of tuple centres, wrapped as specific instantiation of SOLACE services; As remarked in Fig. 5, three cases can be considered. The case tagged with number three is the case in which coordination is constructed and deployed by composing the basic interaction / construction primitives offered directly by the SOLACE infrastructure, according a subjective coordination style. The cases two and one involve the deployment of TuCSoN services, which refers to the same case from a system point of view, but conceptually provide some differences useful to be remarked. The case tagged with number two refers to the availability for agents of TuCSoN services as general purpose mechanisms to setup (articulate) a coordination context, by suitably programming the behaviour of any (possibly new) tuple centres of a node; the term *new* used in this context means tuple centre with an empty behaviour specification, which make them flat tuple spaces. From this perspective, agents consider the tuple centres that they need for the purpose and exploit the service provided by the infrastructure — in the form of the agent coordination context illustrated in previous section — to specify the coordination laws and forge tuple centre behaviours. In other word, they dynamically forge new special purpose coordination services that they as well as other agents can deploy according to the roles defined by the coordination context [22]. Actually, this is the case tagged with one in Fig. 5, referring to the deployment of already programmed tuple centres, as SOLACE specific coordination services constructed for some specific mission (as it could be in the `Twosome` case);
- *Domain*. The domain layer is supported by providing a suitable cognitive manifestation of the coordination contexts which TuCSoN coordination services instances refer to, and which can be dynamically observed by agents. In particular, the ReSpecT specification describing a tuple centre behaviour in joint effort with the tuple set content of the `config` tuple centre residing on each TuCSoN node — describing static and dynamic issues of the coordination and organisation context related to the node [22] — can be considered a direct cognitive manifestation of the coordination services; as promoted by SOLACE vision, these manifestations in form of (multi-set of) logic tuples[1] can be discovered and observed at the domain level by the entities populating the system level. According to the distinction we consider in the previous point, cognitive manifestations could refer to either specific coordination services (case tagged with letter A in Fig. 5), and so not-empty set of ReSpecT specification tuples defining medium behaviour, or general purpose one (case tagged with letter B in Fig. 5).

V. INVESTIGATION

In previous sections we showed how observation and construction concept are generally supported by SOLACE according to the multilayered perspective defined by the framework, and how can be conceived the integration of TuCSoN in such a perspective. Now we discuss the impact of such integration, in particular focusing on the new perspectives on observation and construction — in the context of systemic coordination —

[1] also ReSpecT program (specification) defining the behaviour of a tuple centre are constituted by multi-set of logic tuples [18]

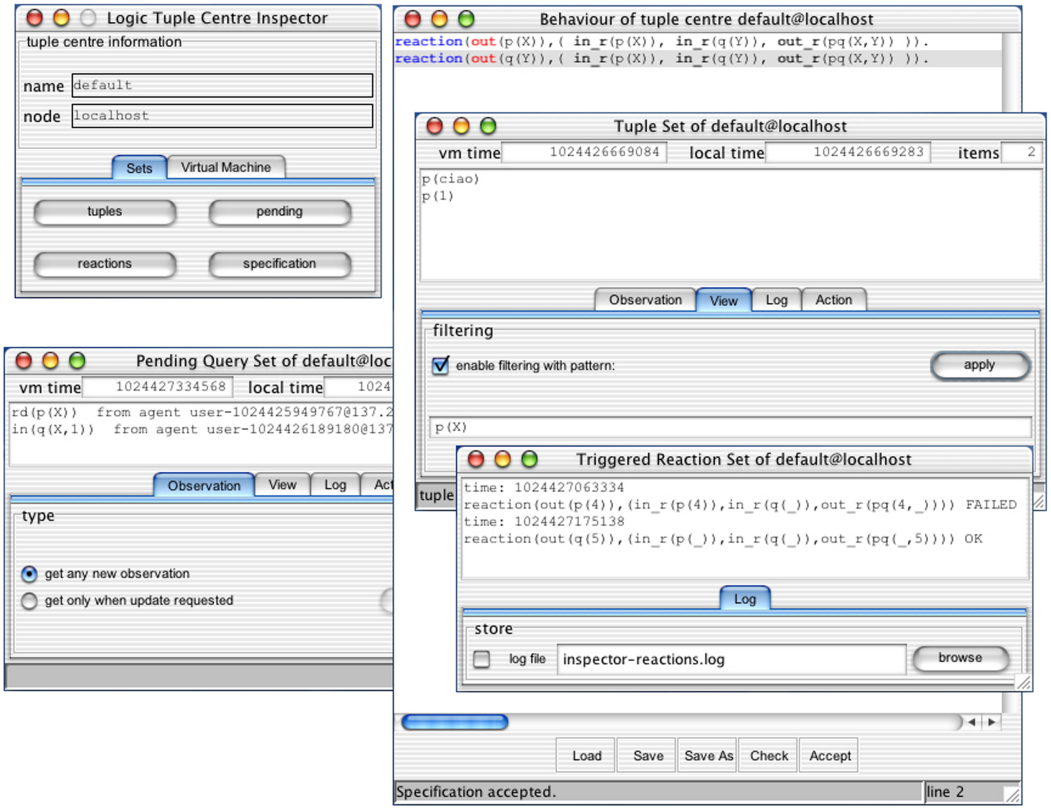

Fig. 6. Some observation views provided by TuCSoN Inspector on the communication and coordination dynamic and static state of a tuple centre (`default` in the picture)

that emerge by deploying TuCSoN as a coordination service provider for SOLACE.

Generally speaking, TuCSoN, by virtue of its objective coordination model makes it possible to balance/distribute dynamically, according to the need, the burden of coordination between agents and the tuple centres, as coordination (media) services provided by the infrastructure, in this case. So agents populating the system level of SOLACE can choose either to engage coordination subjectively or deploy the coordination services that the infrastructure makes them available, contextually to their capability, the social tasks (mission) of the community and the situated environments where they live.

This approach models suitably the coordination perspectives that CSCW accounts for, and which is required in the context of open computational systems. In particular, the coordination media provided by TuCSoN can be considered as the embodiment of *coordination artifacts* explicitly considered in the context of CSCW *articulation* of work. The possibility of exploiting programmable media whose behaviour can be changed dynamically mirror the need expressed in the CSCW context to have flexible artifacts that can be forged / changed dynamically, according to the situated context, possibly by structuring/composing basic coordination mechanisms in order to define any complex collaborative activity [26], [3], [16]. In this way both the situatedness of the work and the benefits of *coordination* — according to CSCW and Activity Theory ontology — are exploited in the same coordination context, as required by complex coordination scenarios in the context of WfMS and CSCW [23].

A. Observation of the agent interaction space

Observation is supported by means of inspectability of tuple centres: both the communication and the coordination state can be dynamically observed, described in terms of logic tuples. In particular, the capability to observe the coordination state refers both to the coordination laws that define medium behaviour, and the dynamic state of the coordination activities. Therefore, according to the systemic ontology, the model makes it possible to observe the *processes* and *collective behavioural patterns* that characterise the coordinated system;

The observation can be exploited at different abstraction levels, according the multilayered perspective provided by the framework: in particular, the cognitive structures related to TuCSoN coordination services populating the domain level — as described in Section IV-D — can be discovered and observed dynamically by agents populating the system level supported by SOLACE. Actually, on the one side TuCSoN infrastructure provides a tool — called Inspector — used to observe and possibly affect the communication and coordination static and dynamic state of a tuple centre [5] (see Fig. 6 for an overall picture about the observation views provided by the Inspector tool). The synergy of this tool with the DISCERN tool provided on top of SOLACE (described in Section IV-B) provides interesting benefits: from the SOLACE (DISCERN) side, the new capability to observe structures and processes specifically concerning interaction and coordination; from the TuCSoN (Inspector) side, the

enhancement of the presentation and analysis of real time information related to communication and coordination state of tuple centres. This is a valuable tool to explore (emerging) patterns in the system interaction behaviour, test related systemic qualities, and their evolution (stability) over time.

B. Construction of the agent interaction space

Construction is supported at two different levels. First of all, a tuple centre *enables* entities interaction and communication, providing means to *construct* communication according to protocols that can be established / composed / changed dynamically. This is possible by composing the basic coordination primitives provided by the coordination language, exploiting the properties of *generative communication* which characterise tuple-based models [10]. Moreover, the adoption of logic tuples as communication language makes it possible to exploit meaningfully the model with agents with heterogeneous computational model.

Then, the most relevant aspect concerns the capability of supporting dynamically the construction / adaptation / evolution of the coordination strategies reflecting the mission of the community, or rather the social rules and norms defining the system as agent society, or again the collaboration policies defining the agent organisation. In systemic terms, that means the capability of dynamically constructing / adapting / constraining *processes* and *collective behavioural patterns*, without acting necessarily on the autonomous entities that compose system *structure*. This construction capability in TuCSoN is provided in particular by three points: *(i)* the programmability of the coordination media (tuple centres); *(ii)* the fact that the programmability is supported dynamically; *(iii)* the expressiveness of the coordination model and, in particular, of the language used to program the coordination media (ReSpecT).

C. Empirical aspects of systemic coordination

According to points sketched, TuCSoN approach to observation/construction can be useful to support the *bottom-up* approach in building system behaviour as accounted in [7], essential in the context of systemic coordination. In particular it provides:

- *Common awareness*. Support for the dynamically increasing of the knowledge/awareness about the system, that grows in time with the going on of the interactions. From this point of view, the coordination media can act as artifacts embedding the *social history* of the system, which can be observed in order to reason about system evolution and to make prediction;
- *Global coherence*. Support for the dynamic development of collective behavioural patterns, toward the global coherent behaviour accounted by the mission of the community. From this point of view, the development of coordination aspects of the system can be conceived as a sort of scientific experiment: system behaviour designers (that could be humans as well as agents) make hypothesis and theories about the laws that can be enforced, in order to guide the system toward the global coherent behaviour that the mission account for; they set the laws by suitably programming medium behaviour, and then they can validate or in-validate their theories, by observing system behaviour evolution (through the inspection of the media). These observations can guide them to change and improve their theories and the coordination laws of the media accordingly;
- *Dissipative Structure*. Support for designing and deploying *dissipative structures* for the open computational system [21]. Tuple centres can be conceived as shared artifacts that provide the function of entropy drainers (dissipative structures) for the agent interaction space, by means of the coordination laws constraining dynamically the space and keeping order despite of the openness of the system.

The continuous feedback loop of articulation, construction, and observation is the cornerstone of *sustainable* coordination, i.e. the capability of measuring and providing some holistic qualities related to systemic coordination and the collective behaviours of the open computational systems [9]. Related ideas and frameworks have been proposed in the theoretical areas of interaction [31] and empirical computer science [32], [33], and we consider them the fundamental scientific and practical approach in understanding, as well as engineering, the structures, patterns, and processes involved in open computational systems.

VI. Conclusions and future work

The paper outlined a conceptual framework that can be used to understand and compare approaches exploited for the theoretical and empirical investigation in the context of systemic coordination in open computational systems. In particular, some investigations about the deployment of TuCSoN as a coordination service provider in the context of SOLACE service-oriented infrastructure, and related benefits for supporting the construction and observation of collective behaviour in open computational systems, have been sketched. In this context, future work accounts for deepening these investigations by comparing the engineering solutions to the coordination issues characterising the TWOSOME system case study provided both from exploiting the native basic interaction mechanisms provided by the SOLACE infrastructure and from deploying TuCSoN coordination services. SOLACE will be enhanced by introducing mission packages comprised of multiple coordination services, e.g., coordination patterns related to computational markets and resource management and objective coordination patterns such as TuCSoN. Finally, complex computational entities will be introduced to provide context-sensitive domain reasoning, e.g., about system stability and maintenance.

Acknowledgments

The authors would like to express their gratitude towards the members of the *Societies of computation laboratory* (SOCLAB) in Ronneby/Sweden (Jimmy Carlsson, Johan Lindblom, Jonan Rosquist, Christian Seger, Robert Sandell, Tomas Sareklint, and Bjorn Törnqvist), who developed the practical tools (SOLACE and DISCERN) and system (TWOSOME) outlined herein — and

whose interaction was fundamental to outline the points expressed in the paper.

REFERENCES

[1] F. Berman, G. Fox, and T. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, Dec. 2002.

[2] P. Ciancarini, A. Omicini, and F. Zambonelli. Multiagent system engineering: the coordination viewpoint. In N. R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI — Agent Theories, Architectures, and Languages*, volume 1767 of *LNAI*, pages 250–259. Springer-Verlag, Feb. 2000.

[3] M. Cortes. A coordination language for building collaborative applications. *International Journal of Computer Supported Cooperative Work (CSCW)*, 9(1):5–31, 2000.

[4] E. Denti, A. Natali, and A. Omicini. Programmable coordination media. In D. Garlan and D. Le Métayer, editors, *Coordination Languages and Models – Proceedings of the 2nd International Conference (COORDINATION'97)*, volume 1282 of *LNCS*, pages 274–288, Berlin (D), 1–3 Sept. 1997. Springer-Verlag.

[5] E. Denti, A. Omicini, and A. Ricci. Coordination tools for the development of agent-based systems. *Applied Artificial Intelligence*, 16(9), Oct. 2002.

[6] D. Estrin, R. Govindan, and J. Heiderman. Embedding the internet: Introduction. *Communication of the ACM*, 43(5), 2000.

[7] M. Fredriksson and R. Gustavsson. A methodological perspective on engineering of agent societies. In A. Omicini, P. Petta, and R. Tolksdorf, editors, *Engineering Societies in the Agents World II*, volume 2203 of *LNAI*, pages XI–195. Springer-Verlag, Dec. 2001. 2nd International Workshop (ESAW'01), Prague, Czech Republic, 7 July 2001, Revised Papers.

[8] M. Fredriksson and R. Gustavsson. Theory and practice of behavior in open computational systems. In R. Trappl, editor, *Cybernetics and Systems 2002*, Vienna, Austria, 2002. Austrian Society for Cybernetic Studies. 16th European Meeting on Cybernetics and System Research (EMCSR 2002), 2–5 Apr. 2002, Vienna, Austria, Proceedings.

[9] M. Fredriksson, R. Gustavsson, and A. Ricci. Sustainable coordination. Technical report, Department of Software Engineering and Computer Science of the Blekinge Institute of Technology, Ronneby, Sweden, Oct. 2001. Submitted for publication as a chapter in M. Klusch, S. Bergamaschi, P. Edwards and P. Petta, editors, *Intelligent Information Agents: An AgentLink Perspective*, LNCS State of the Art Surveys Series, Springer Verlag, Heidelberg, Germany, 2002.

[10] D. Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[11] I. A. Group. Software technologies, embedded systems and distributed systems: A european strategy towards an ambient intelligent environment – european commission IST report. `http://www.cordis.lu/ist/istag.html`, June 2002.

[12] R. Gustavsson and M. Fredriksson. Coordination and control in computational ecosystems: A vision of the future. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 18, pages 443–469. Springer-Verlag, Mar. 2001.

[13] M. Huhns. Networking embedded agents. *IEEE Internet Computing*, 3(1):91–93, Jan. 1999.

[14] IBM. Autonomic computing: IBM's perspective on the state of information technology – white paper. `http://www.ibm.com/research/autonomic`, 2002.

[15] V. Kotov. Towards service-centric system organization. Technical Report HPL-2001-54, Computer Systems and Technology Laboratory – Hewlett Packard (HP), Palo Alto, Mar. 2001.

[16] D. Li and R. R. Muntz. COCA: Collaborative objects coordination architecture. In *Computer Supported Cooperative Work*, pages 179–188, 1998.

[17] A. Omicini. Towards a notion of agent coordination context. In D. Marinescu and C. Lee, editors, *Process Coordination and Ubiquitous Computing*, pages 187–200. CRC Press, 2002.

[18] A. Omicini and E. Denti. Formal ReSpecT. In A. Dovier, M. C. Meo, and A. Omicini, editors, *Declarative Programming – Selected Papers from AGP'00*, volume 48 of *Electronic Notes in Theoretical Computer Science*, pages 179–196. Elsevier Science B. V., 2001.

[19] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, Nov. 2001.

[20] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, Sept. 1999. Special Issue: Coordination Mechanisms for Web Agents.

[21] V. D. Parunak. 'Go To The Ant': Engineering principles from natural agent systems. *Annals of Operations Research*, 75:69–101, 1997.

[22] A. Ricci and A. Omicini. Agent coordination context: Experiments in TuCSoN. In *WOA'02 – Dagli oggetti agli agenti: tendenze evolutive dei sistemi software*, Milano, Italy, 17–19 Nov. 2002. This Workshop.

[23] A. Ricci, A. Omicini, and E. Denti. Activity theory as a framework for mas coordination. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *3rd International Workshop on Engineering Societies in the Agent World (ESAW '02)*, Sept. 2002.

[24] K. Schmidt and L. Bannon. Taking CSCW seriously: Supporting articulation work. *International Journal of Computer Supported Cooperative Work (CSCW)*, 1(1):7–40, 1992.

[25] K. Schmidt and C. Simone. Coordination mechanisms: Towards a conceptual foundation of cscw systems design. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, 5(2–3):155–200, 1996.

[26] K. Schmidt and C. Simone. Mind the gap! towards a unified view of CSCW. In *The Fourth International Conference on the Design of Cooperative Systems COOP 2000*, 2000.

[27] M. Schumacher. *Objective Coordination in Multi-Agent System Engineering – Design and Implementation*, volume 2039 of *LNAI*. Springer-Verlag, Apr. 2001.

[28] D. Tennenhouse. Pro-active computing. *Communication of ACM*, 43(5):43–50, may 2000.

[29] M. Viroli and A. Omicini. Coordination as a service: Ontological and formal foundation. In A. Brogi and J.-M. Jacquet, editors, *Foundations of Coordination Languages and Software Architectures – Papers from FOCLASA'02*, volume 68 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science B. V., 2002.

[30] J. Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7):76–82, July 1999.

[31] P. Wegner. Why interaction is more powerfull than algorithm. *Communication of ACM*, 40(5):80–91, May 1997.

[32] P. Wegner. Interactive foundations of computing. *Theoretical Computer Science*, 192(2), Feb. 1998.

[33] P. Wegner. Toward empirical computer science. *The Monist*, 82(1), Jan. 1999.

[34] M. Weiser. Hot topics: Ubiquitous computing. *IEEE Computer*, 26(10), Oct. 1993.