2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops

Formal Modeling of Socio-technical Collective Adaptive Systems

A. Coronato*, V. De Florio†, M. Bakhouya‡, G. Di Marzo Serugendo§
*National Research Council, 80131 Napoli Italy
E-mail: antonio.coronato@na.icar.cnr.it
†University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium
E-mail: vincenzo.deflorio@ua.ac.be
‡Aalto University, Otakaari 4, FIN-00076 Aalto, Finland
E-mail: mohamed.bakhouya@aalto.fi
§Universite de Geneve, Rte de Drize 7, CH-1227 Carouge, Switzerland
E-mail: Giovanna.DiMarzo@unige.ch

***Abstract*—Socio-technical collective adaptive systems (CAS) are composed of different heterogeneous parts or entities (e.g., individuals, groups, computers, robots, agents, devices, software, services, sensors) that interact collectively in a complex and largely unpredictable manner. Their ability to be adaptive requires incorporating mechanisms that allow entities to interact and perform actions favoring the emergence of a global desired behavior or service. Therefore, analyzing and discovering new emerging behaviors and/or unexpected abnormal behaviors, as well as new opportunities of services emergence, require methods and tools for formally specifying, verifying, and validating foundational properties at design time and while running (runtime verification). In this paper, three emerging formal methods—situation calculus, ambient calculus, and bigraphical reactive systems—are first studied to shed more light on their appropriateness for specifying and verifying socio-technical CAS. A case study is used and its formal model using these methods is presented to show their fundamental features and limitations for modeling these systems.**

***Keywords*-Socio-technical systems; social organizations; collective adaptive systems; Formal modeling.**

I. Introduction

Socio-technical collective adaptive systems are composed of different heterogeneous parts or entities that interact and perform actions favoring the emergence of global desired behavior. In this type of systems entities might join or leave without disturbing the collective, and the system should self-organize so as to continue performing its tasks.

In socio-technical CAS, analyzing and discovering new emerging behaviors and/or unexpected abnormal behaviors, as well as new opportunities of services emergence, require methods and tools for formally specifying, verifying, and validating foundational properties at design time and while running (runtime verification). Run-time verification allows verifying that the system continues to meet its design objectives as it evolves. This paper presents a preliminary study of formal modeling of socio-technical CAS focusing on three recent and emerging formal methods: situation calculus, ambient calculus, and bigraphical reactive systems. In order to show their practical use, their fundamental features, and their limitations for modeling these systems, a small scale case study, represented by a triggering alarm scenario in a hospital, is used and its formal model using these methods is presented. The triggering alarm system can be seen as a socio-technical CAS of modest size, essentially made of tens of artificial entities, characterized by limited scope, a time frame of minutes to some hours, space scale ranges from a room to a building, and social scope (e.g., patient monitoring.)

The remainder of this paper is structured as follows. In Sect. II a brief description of existing work of formal specification methods proposed in the literature is presented. Due to page limitation reader might refer to [1], [2] for more details about research issues in designing and operating principles of socio-technical CAS. Section III briefly introduces the formal model of the triggering alarm scenario in a hospital based on ambient calculus, situation calculus, and bigraphical reactive systems. Section IV finally summarizes the features provided by these methods and briefly sketches future research direction for formal modeling and verification of socio-technical CAS.

II. Related work

Socio-technical CAS are composed of humans (individuals or groups) and machines (sensors, service, computers) that form networks of entities collectively collaborating to provide one or more services according to the current context. Principles for engineering socio-technical CAS are mainly classified into two categories [1], [3], [4]: design principles and operating principles. Design principles are necessary to build and manage the system by enabling the emergence of behaviour and facilitating prediction and control of those behaviours. Operating principles should define techniques that allow the system to operate taking into consideration the diversity of objectives within the system,

978-0-7695-4895-1/12 $26.00 © 2012 IEEE
DOI 10.1109/SASOW.2012.40

conflicts resolution, long term stability, and the need to reason in the presence of uncertainty (e.g., partial, noisy, out-of-date and inaccurate information).

Analyzing and discovering new emerging behaviors and/or unexpected abnormal behaviors in socio-technical CAS is one of issues that should be addressed. Recently, formal modeling and verification methods and tools have been developed to increase dependability and eliminate errors at early design stages of the development of adaptive systems. The focus is mainly on developing formal methods for modeling pervasive and ubiquitous applications that are characterized by many desired properties such as the mobility of users and devices. These methods can be categorized into two main families [5]: context-based modeling methods and behaviour-based methods. Context-based modeling methods have used techniques like ontologies and situation models. An example is CML (Context Modeling Language) [6]. Based on Object-Role Modeling (ORM), CML was developed for conceptual modeling of databases. CML provides a graphical notation designed to support software developers in analyzing and formally specifying the context requirements of context-aware applications. Its graphical notation for analyzing and designing context-aware applications as well as its support for capturing and evaluating historical information constitute the main strength of this method. Situation calculus (SC) is a logical language that was proposed for representing changes in dynamic environments [7]. This method provides basic concepts that are situations, actions and fluents. Actions performed by agents allow dynamic world change from one situation to another one. Fluents are used to describe the effects of actions.

Behaviour-based methods have focused on modeling and verifying the behavior of system entities. Examples of emerging formal methods include *ambient calculus* and *bigraphical reactive systems*. Ambient calculus (AC) has mainly derived from π-*calculus* [8] for modeling and analyzing ubiquitous and pervasive systems [9]. Bigraphical reactive systems (BRS) was proposed in [8], in addition to ambient calculus, to deal with the interactions between mobile entities. This method allows designers to specify both the location of entities and their interactions. According to [8], BRS is still in its infancy and it has the potential to become a foundational model—if supported with appropriate software tools for analysis like model checking. In the rest of this paper, the major features of SC, AC, and BRS methods are presented and a small scale socio-technical CAS is then specified by each of these methods.

III. Case Study and Formal Model

The case study considered describes a triggering alarm scenario in a hospital according to the following scenario:
Communities ={Hospital};
Roles = {Nurse, GeneralPractitioner, Doctor};
Actants/entities = {Mary, Jane, Bob, House, Alice, Mike, Hanna};
Rooms within the hospital ={1, 2, Consulting Room};
Systems = {MonitoringSystem};
Messages = {Alarm};
Activities = {doInjection, doMedication, Talk, Listen};
Initial World = {Mary and Jane are in room 1; Bob is in room 2; House, Alice, Mike, and Hanna are in the Consulting Room; House is talking to Alice and Hanna}.

A. Ambient calculus

In ambient calculus, all entities are described as processes. An important kind of process is an "ambient", which is a bounded place where computation happens. Each ambient has a name and can be moved as a whole, into and out of other ambients, by performing *in* and *out* operations. Ambient calculus allows a description of complex phenomena in terms of creation and destruction of nested ambients, and movement of processes into and out of these ambients.

In ambient calculus, *n*[P] represents an ambient, *n* is the name of the ambient, and *P* is the process running inside it. The process "0" is the one that does nothing. Parallel execution is denoted by a binary operator "|" that is commutative and associative. "!P" denotes the unbounded replication of the process P within its ambient. The following expression:

$$n[P_1 \mid \ldots \mid P_j \mid m_1[...] \mid \ldots \mid m_k[...]]$$

describes an ambient called *n*, which contains j processes named P_1, ..., P_j and k ambients named $m_1, ...m_k$. Figure 1 reports the equivalent graphic representation for *n*.

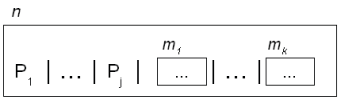

Figure 1. Ambient calculus: graphical representation for ambients.

Some processes can execute an action that changes the state of the world around them. This behavior of processes is specified using capabilities. Process $M.P$ executes an action described by the capability M and then continues as the process P. There are three kinds of capabilities: for entering (*in*), for exiting (*out*) and for opening up (*open*) an ambient, respectively. Figure 2 shows these three capabilities. The process "in *m*.P" instructs the ambient surrounding "in *m*.P" to enter a sibling ambient named *m*. The reduction rule, which specifies the change in state of the world, is:

$$n[\text{in } m.\text{P}] \mid m[\text{Q}] \rightarrow m[n[\text{P}] \mid \text{Q}]. \quad \text{(a)}$$

The action "out *m*.P" instructs the ambient surrounding "out *m*.P" to exit its sibling ambient named *m*. The reduction rule is:

$$m[n[\text{out } m.\text{P}] \mid \text{Q}] \rightarrow n[\text{P}] \mid m[\text{Q}]. \quad \text{(b)}$$

The action "open *m*.P" provides a way of dissolving the boundary of an ambient named *m* located at the same level

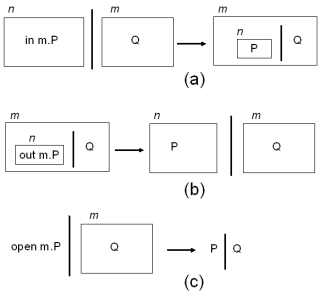

Figure 2. Ambient calculus: capabilities.

as open. The rule is:

$$\text{open } m.\text{P} \mid m[\text{Q}] \rightarrow \text{P} \mid \text{Q}. \quad \text{(c)}$$

Output and input actions are respectively denoted by "$\langle$M$\rangle$.P" and "(x).P", to release and capture capabilities. Finally, the restriction operator (Vn)P creates a new (unique) name n within a scope P. The new name can be used to name ambients and to operate on ambients by name.

To show how this tool can be used for formal specification we consider the alarm triggering scenario. Figure 3 describes the initial world. Figure 4, instead, represents the initial status when an alarm is generated at time T_0. After that, several transitions (here not reported for the sake of brevity) trigger until the final status shown in Fig. 5 is reached. Each transition describes an elementary movement, of a person or another kind of ambient, which is performed within the environment. AC mainly allows the specification of movements. Unfortunately, it doesn't allow the specification of interactions (e.g. speaking, listening to, etc.). Moreover, it doesn't come with automatic tools for checking the specification.

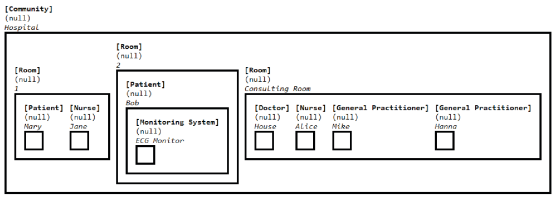

Figure 3. Ambient Calculus: a model of the initial world.

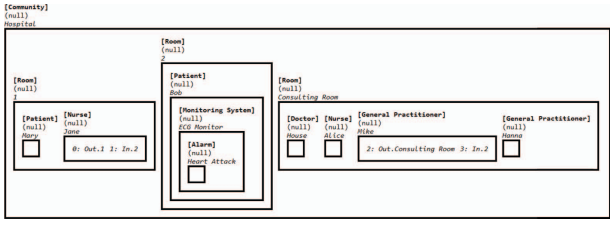

Figure 4. Ambient Calculus: movements after a heart attack alarm.

B. Bigraphical Reactive Systems

BRS is a graphical model of mobile computing-based applications that incorporates both locality and interactions. A bigraph is a structure that enables the description of both the location of entities and their interactions. A bigraph includes two graphs, the *topograph* (or place graph) that describes locations of nodes and the *monograph* (or link graph) that describes their links. An example of bigraph is reported in Fig. 6. In the picture, locations are represented by nodes of type r_i, v_j, and s_k. Nodes of type r_i are called roots and are part of the *outer face* of the bigraph, as well as names of type y_l. Nodes of type s_k, which are called sites, represent a sort of hole that can be replaced by other bigraph. Sites and inner names (x_m) form the *inner face* of the bigraph. The bigraph *B* has, then, an interface like *B*: <3, $\{x_0,x_1\}$> → <2, $\{y_0,y_1,y_2\}$>, indicating that *B* has three sites and inner names x_0, x_1, and two roots and outer names y_0, y_1, and y_2.

It is possible to compose the bigraph *B* with another bigraph *A* by nesting *A* into *B* (written *B*o*A*) if, and only if, the inner face of *B* matches the outer face of *A*. Other possible operations are parallel product and merge product, as shown in Fig. 7. Such a model deals with static structures. For dynamics, there will be reaction rules that change the state of the system. A reaction rule is a pair of bigraphs called *redex* (or pre-condition) and *reactum* (or post-condition). An example of reaction rule is described in Fig. 8, which depicts the effect of the movement of the node v_2 into the node v_3.

Figure 9 shows a generic model for a social organization. In particular, we model a *Society* as a root of a bigraph; *Organizations* and *Communities* as clouds within the society; we use a variety of symbols for *Roles*; and links for communication channels. In addition to this, not shown in the figure, we also take into account physical locations.

Figure 10, instead, specifies a rule for partitioning the society. Indeed, once three different roles of type A, B and C are inactive within the organization, such roles will arrange themselves into a new community and interact each another as described by the reaction rule R.

For the case study described above, the initial world is modeled as in Fig. 11. The society has initially one organization of inactive people and a community formed by Dr. House, nurse Alice and the general practitioner Hanna

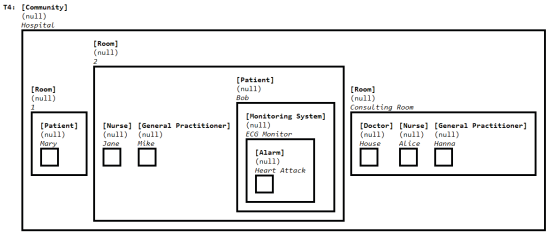

Figure 5. Ambient Calculus: the final status

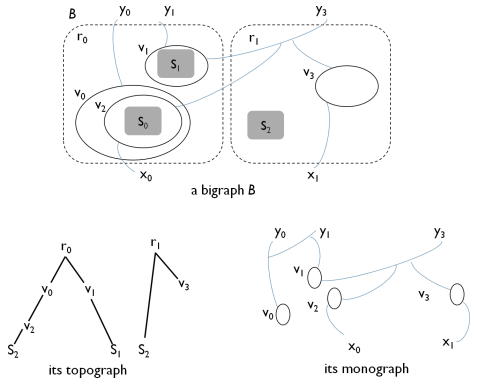

Figure 6. BRS: Example of bigraph.

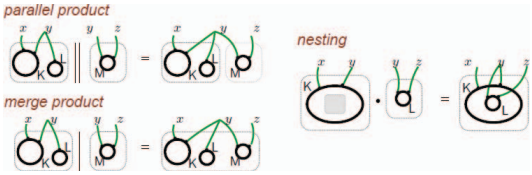

Figure 7. BRS: Bigraph operations.

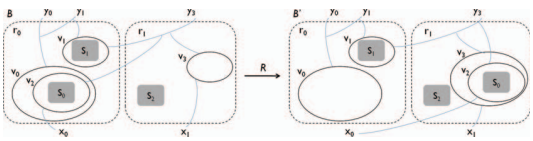

Figure 8. BRS: Example of reaction.

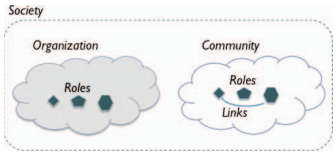

Figure 9. A generic model for a social organization.

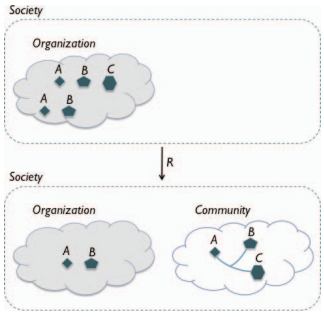

Figure 10. An example of partitioning.

who are talking for planning some medical treatment.

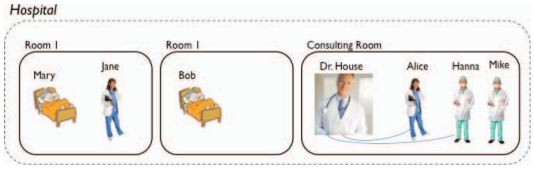

Figure 11. BRS: A model of the initial world.

Figure 12 describes the consequences of an alarm generation, which is an exogenous perturbation for the society. It is possible to specify the movements of both nurse Jane and the general practitioner Mike. All such people will form a new community for handling the alarm and having care of the patient.

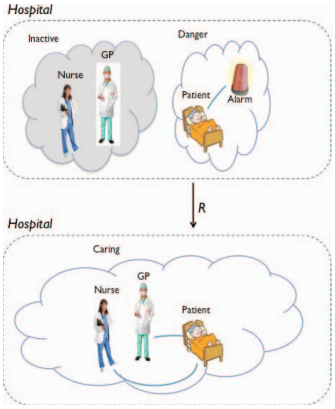

Figure 12. BRS: Movements of the nurse and general practitioner.

Moreover, it is possible to specify a generic rule for the case of an alarm, as shown in Fig. 12. Here we are specifying that one nurse and one general practitioner must approach the patient. As for AC, BRS has no automatic tool for checking the specification neither for reasoning on properties.

C. Situation Calculus

The basic *situation calculus* (*SC*) is due to John McCarthy [7] and has been adopted to model dynamically changing worlds. Three basic sorts in *SC* are: **Actions**, which can be performed in the world and can be quantified; **Fluents**, that describe the state of the world (these are predicates and functions whose value may change depending on situation); and **Situations**, which represent a history of action occurrences.

A dynamic world is modeled through a series of situations as a result of various actions being performed within the world. It is important to note that a situation is not a state of the world, but just a history of a finite sequence of actions.

The constant S_0 denotes the initial situation; whereas, *do(a,S)* indicates the situation resulting from the execution of the action *a* in situation *S*.

The dynamic world is axiomatized by adding **initial world axioms**, **unique names axioms**, **preconditions**, **effect axioms**, and **successor state axioms**. The **initial world axioms** describe the initial status of the environment, its objects, their position into the environment, their properties, etc. A unique name axiom for situations states that if the execution of actions a_1 and a_2, respectively from S_1 and S_2, lead the environment to the same situation, then, necessarily, $a_1 = a_2$ and $S_1 = S_2$. Unique name axioms also define the set of basic actions that can be performed within the environment. A precondition is formalized using the binary predicate symbol *Poss(a,S)*, which describes a condition that must hold in order to execute the action *a* in situation *S*. An effect axiom, instead, describes the effect on a fluent (e.g. *F(x⃗,S)*) caused by the execution of an action in a specific situation ($F(\vec{x}, do(a, S))$). Unfortunately, effect axioms are not sufficient to describe the changing world. Indeed, it must be specified for each fluent not only the effect of each affecting action, but also the non-effect of the other actions. This is a well known problem—the frame problem—that entails the specification of $2 \times A \times F$ axioms being A the number of actions and F the number of fluents. To reduce such a problem, we refer to success state axioms of the form:

$$F(\vec{x}, do(a, S)) \equiv \gamma_F^+(\vec{x}, a, S) \vee (F(\vec{x}, S) \wedge \neg\gamma_F^-(\vec{x}, a, S)) \quad (1)$$

where γ_F^+ (x⃗,a,S) is a first-order formula—with free variables among x⃗, a, and S—that makes the F's truth value changing to true. Analogously, γ_F^- (x⃗,a,S) is a first-order formula that makes the F's truth value changing to false. Intuitively, it is possible to state that a fluent's truth value is true after executing an action *a* if, and only if, the action has the effect to make the fluent true (as stated by one of the effect axioms) or, the fluent was already true before executing *a* and the action has not the effect to make it false. In such a case, only F successor state axioms must be formalized.

A *basic action theory* is a set of axioms including the initial world axioms, unique names axioms, precoditions, and successor state axioms, that describe a dynamically changing world. A basic action theory is defined in the next section to represent and identify dangerous or anomalous situations of our case study.

Another remarkable characteristic of *SC* is the possibility of indicating goals and planning. Such a characteristic is exploited to search and identify strategies that would enable to recover from a dangerous situation. The planning problem, under such hypotheses, can be reformulated as follows: starting from an axiomatized initial situation S_0 and having a goal *G*, find a sequence of actions S in which the goal *G(S)* is true:

$$Axioms \models (\exists S).executable(S) \wedge G(S) \quad (2)$$

where executable(S) concerns the possibility of executing actions of S based on precondition axioms. Planning, thus, can be viewed as a side-effect of theorem-proving. A basic mechanism that can be adopted to solve the planning problem is regression. Suppose that the goal *G(S)* includes a relational fluent *F(x⃗,do((α,σ))*, where F's successor state axiom is *F(x⃗,do((a,S))* ≡ ϕ_F*(x⃗,a,S)*. As a consequence, by substituting *F(x⃗,do((a,S))* with ϕ_F*(x⃗,a,S)* we obtain an expression *G'(S')* that is closer to S_0. In our scenario, a goal is specified in response to a dangerous situation. In particular, once a dangerous situation is detected, an intelligent agent is queried to plan a strategy that will bring the patient and the environment in a safe situation *G(S)*. The agent uses regression to find a sequence of executable actions from current unsafe situation.

It is finally remarkable to note that under certain conditions (Clark's theorem), an executable Prolog program is directly obtained by appling Lloyd-Topor transformations to the basic action theory. This can be interpreted by a Golog interpreter and represent an intelligent agent for the detection of anomalous and dangerous situations. As well, another intelligent agent may be triggered to plan a sequence of actions for the goal $G(S)$.

Concerning the case study, the initial world is described by means of the following facts (written directly in Golog):

```
/* **** Initial world *** */
isNurse(Jane).  isNurse(Alice).
isDoctor(House).
isPatient(Mary).  isPatient(Bob).
isGeneralPractitioner(Mike).  isGeneralPractitioner(Hanna).

isRoom(1).  isRoom(2).  isRoom(ConsultingRoom).

isCommunity(Hospital).

isIn(1,Hospital).  isIn(2,Hospital).  isIn(ConsultingRoom,Hospital).
isIn(Mary,1).  isIn(Jane,1).  isIn(Bob,2).
isIn(House,ConsultingRoom).  isIn(Alice,ConsultingRoom).
isIn(Mike,ConsultingRoom).  isIn(Hanna,ConsultingRoom).

isMonitoring(ECGMonitor,Bob).

isTalkingTo(House,Hanna).  isTalkingTo(House,Alice).
isListeningTo(Hanna,House).  isListeningTo(Alice,House).
```

In addition to AC and BRS, we can describe activities as in what follows (written directly in Golog):

```
/* **** Primitive control actions*** */
primitive_action(riseAlarm(Patient)).
primitive_action(switchOff(Alarm)).
primitive_action(startTalking(Person,toPerson)).
primitive_action(stopTalkink(Person,toPerson)).
primitive_action(startListening(Person,toPerson)).
primitive_action(stopListening(Person,toPerson)).
primitive_action(startHavingCare(Person,Patient)).
primitive_action(stopHavingCare(Person,Patient)).
primitive_action(startInjection(Person,Patient)).
primitive_action(stopInjection(Person,Patient)).
primitive_action(startMedication(Person,Patient)).
primitive_action(stopMedication(Person,Patient)).
primitive_action(goInto(Person,Room)).
```

We can also write down preconditions for the execution of such actions (omitted here for the sake of brevity). Fluents describe the current situation and successor state axioms specify how they change after the execution of actions. For example, the following fluents can be specified for the case study:

isHavingCare(Patient); It is true when there exists a nurse who is taking care of patient *Patient*.

isInDanger(Patient); It is true when the monitoring system detects a heart attack and rises an alarm for the patient *Patient*.

The following successor state axiom concerns the fluent *isHavingCare*:

$$\begin{aligned}\mathbf{isHavingCare}(\mathbf{Patient}, \mathbf{do}(\mathbf{a}, \mathbf{S})) \equiv \\ \{a = startHavingCare(Nurse, Patient) \vee \\ isHavingCare(Patient, S) \wedge \\ \{a \neq stopHavingCare(Nurse, Patient)\}\end{aligned} \quad (3)$$

Axiom 3 specifies that the fluent becomes true whenever there is a nurse who executes the action *startHavingCare* or, it remains true if it was already true in the previous situation and the nurse doesn't execute the action *stopHavingCare(Patient)*.

IV. Summary and discussion

Table I describes some characteristics of the formal methods presented in this paper and of interest for socio-technical CAS. Although all methods provide mechanisms for the modeling of concepts, *AC* and *BRS* support only limited kinds of static relationships. Indeed, *BRS* focuses on two kinds of relations (i.e. "contains" and "interacts with"). *AC* doesn't support any mechanism for interaction, whereas in *SC* any kind of relationship can be defined as a fact. As a consequence, dynamic modeling concerns movements for *AC*; movements and interactions for *BRS* and any kind of action for *SC*. Real-time constraints are substantially uncovered by such methods though extensions of *AC* and *SC* exist aiming at taking someway into account temporal aspects.

As of verification, none of such methods are equipped with model checking tools (for *AC*, however, algorithms have been defined); whereas, some runtime verification is possible with *SC* as long as the method is interpreted by Prolog dialects such as Golog; thus, verification can be performed by ad-hoc devised intelligent agents.

Finally, it is possible to state that *SC* is generally the most appropriate logic for the situation awareness. However, it must also be underlined that *AC* and *BRS* may be more proficiently adopted with the aim of focusing on phenomena concerning movements or interactions and without the aim of having verification. This is mainly due to the existence of specific mechanism supported by a graphic representation, which ease both the writing and reading of specifications.

In summary, despite the simple formulation of our case study, none of reviewed methods allows a complete coverage of requirements and properties for this type of socio-technical CAS and a research direction in this area is required. Scalability of these methods should be also studied to access their suitability for modeling large scale socio-technical CAS with increasing level of complexity. For example, a system with active human participation, with a larger scope, time, space scale and social scope e.g. space scale ranges from several buildings to a city with a time frame of hours or days, with thousands of interacting entities. A case study within a large hospital with a complex organizational structure that refers to different levels of management is a typical example, e.g., networked cooperation of the different actors of the ambulatory health care: doctors, hospitals, nursing homes, care services at home and associations. Our current work on service-oriented communities and fractal social organizations focuses on this type of socio-technical systems.

Table I
Summary of the characteristics of the reviewed formal methods.

Socio-technical CAS characteristics	**Method**		
	AC	BRS	SC
Social concepts	✓	✓	✓
Relationships	~	~	✓
Interactions		✓	✓
Dynamics	~	~	✓
Real-time constraints	~		~
Design-time verification	~		
Run-time verification			~
Situation awareness	~	~	✓

References

[1] D. Miorandi, F. De Pellegrini, O. Mayora, and R. Giaffreda, "Collective adaptive systems: Scenarios, approaches and challenges," ftp://ftp.cordis.europa.eu/ pub/fp7/ict/docs/fet-proactive/shapefetip-cas10_en.pdf.

[2] S. Kernbach, *Handbook of Collective Robotics: Fundamentals and Challenges*. Pan Stanford Publishing, 2011.

[3] Anonymous, "Collective adaptive systems," *Expert Consultation Workshop, Report*, European Commission, 2009.

[4] S. Kernbach, T. Schmickl, and J. Timmis, "Collective adaptive systems: Challenges beyond evolvability," *http://arxiv.org/abs/1108.5643v1*, 2011.

[5] M. Bakhouya, R. H. Campbell, A. Coronato, G. D. Pietro, and A. Ranganathan, "Introduction to special section on formal methods in pervasive computing," *ACM TAAS*, vol. 7, pp. 258–267, 2012.

[6] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: Models and approach," *Pervasive Mob. Comput.*, vol. 2, pp. 37–64, February 2006.

[7] J. McCarthy, "Actions and other events in situation calculus," in *Proc. of the 8th Int.l Conf. on Principles of Knowledge Representation and Reasoning (KR2002)*, p. 615–628, 2002.

[8] R. Milner, "Bigraphs and their algebra," *Electr. Notes Theor. Comput. Sci.*, vol. 209, pp. 5–19, 2008.

[9] L. Cardelli and A. D. Gordon, "Mobile ambients," *Theor. Comput. Sci.*, vol. 240, pp. 177–213, June 2000. [Online]. Available: http://portal.acm.org/citation.cfm?id=340593.340606