Applying algebraic approaches for modeling workflows and their transformations in mobile networks

Paolo Bottoni[a], Fabio De Rosa[a], Kathrin Hoffmann[b] and Massimo Mecella[a]
[a]*Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, Rome, Italy*
E-mail: {bottoni, derosa}@di.uniroma1.it; mecella@dis.uniroma1.it
[b]*Technische Universität Berlin, Institut für Softwaretechnik und Theoretische Informatik, Berlin, Germany*
E-mail: hoffmann@cs.tu-berlin.de

Abstract. In emergency scenarios we can obtain a more effective coordination among team members, each of them equipped with hand-held devices, through the use of workflow management software. Team members constitute a Mobile Ad-hoc NETwork (MANET), whose topology both influences and is influenced by the workflow. In this paper we propose an algebraic approach for modeling workflow progress as well as its modifications as required by topology transformations. The approach is based on Algebraic Higher-Order Nets and sees both workflows and topologies as tokens, allowing their concurrent modification.

Keywords: Mobile Ad-hoc NETworks (MANETs), adaptive workflow management, algebraic approaches

1. Introduction

The widespread availability of network-enabled hand-held devices (e.g., PDAs with WiFi capabilities) has made the development of pervasive computing environments an emerging reality. Mobile (or Multi-hop, according to some authors) Ad-hoc NETworks (MANETs, [3]) are networks of mobile devices that communicate with one another via wireless links without relying on an underlying infrastructure; this distinguishes them from other types of wireless networks, e.g., cell networks or infrastructure-based wireless networks. In order to achieve communication, each device in a MANET acts both as an endpoint and as a router forwarding messages to devices within radio range. MANETs are a sound alternative to infrastructure-based networks in cases when an infrastructure has never been available, is no longer available, or cannot be used, as in emergency scenarios.

Operators acting in such scenarios would benefit from software support to their collaboration; in particular, a *coordination layer* would allow human actors to execute sets of activities (in sequence, concurrently, etc.) through specific applications (e.g., computer supported cooperative work – CSCW – tools [27], workflow management applications [39], etc.) running on hand-held devices, thus supporting the execution of *cooperative processes*. All such applications typically require the device to be continually connected (e.g., for data/information sharing, activity scheduling and coordination, etc.); but in general continual connection is not guaranteed in MANETs.

Hence, we are investigating a specific architecture, targeted to CSCW and workflow management applications constituting the coordination layer, that is able to maintain the continual connection of MANET devices, and to modify the workflow schema at run time. Basic problems for such an architecture concern the prediction of possible disconnections of devices, addressed in [11], and the formal definition of transformations of the workflow schema on the basis of the changed network topology and workflow execution state, which is the aim of the present work.

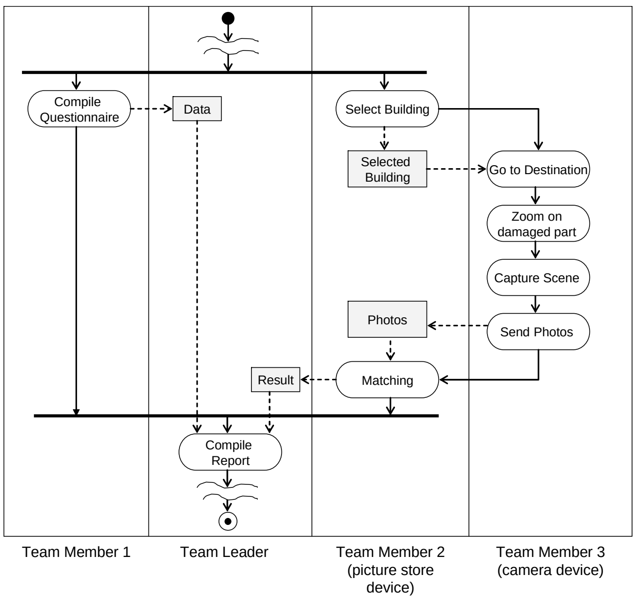
As a motivating example, the reader should consider a scenario in archaeological disaster/recovery: after an earthquake, a team (led by a team leader) is equipped with mobile devices (laptops and PDAs), and sent to the affected area to evaluate the state of archaeological sites and the state of precarious buildings; the goal is to draw a situation map in order to schedule restructuring jobs. A typical cooperative process to be enacted by the team would be as shown in Fig. 1(a) (depicted as a UML Activity Diagram); it represents the initial workflow schema to be enacted on the MANET:

– the team leader has previously stored (for example by a previous download from the headquarter server) all the details of the area, including a map of the site, the list of the most sensible objects located in the site, and precedent reports/materials;
– the team is considered as an overall MANET, in which the team leader's device (requiring the most computational power, therefore usually a laptop) coordinates the other team member devices, by providing suitable information (e.g., maps, sensible objects, etc.) and assigning activities;
– team members are equipped with hand-held devices (PDAs), which allow them to execute some operations, but do not have much computational power. Such operations, possibly provided through the support of particular hardware (e.g., digital cameras, GPRS/UMTS/satellite connections, computational power for image processing, main storage, etc.), are offered as software services to be coordinated. Team member 1 (by using his/her device) could fill in some specific questionnaires (after a visual analysis of a building), to be analyzed by the team leader with the help of a specific software, so as to schedule next activities; team member 3 could take some pictures of the precarious buildings, whereas team member 2 is in charge of specific image processing tasks on previous and recent pictures (e.g., for first identification of architectural anomalies).

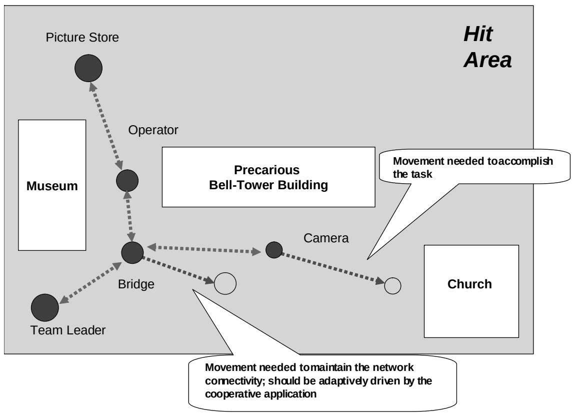
In the latter case, it might be useful to match the new pictures with previously stored ones; then it is necessary that the device with the high-resolution camera and the one storing the old pictures are connected, in order to execute this match.

But in a particular scenario, as the one depicted in Fig. 1(b), it could happen that the movement of the operator/device equipped with the camera would result in a disconnection from the others. A cooperative architecture should be able to predict such a situation and to alert the coordination layer to select a possible "bridge" device (e.g., the one owned by team member 4) to follow (i.e., to move afterwards) the "going-out-of-range" operator/device, thus maintaining the connection and ensuring a path among devices. This in general may result in a change of the MANET topology. In such a way the coordination layer, on the basis of the disconnection prediction, schedules the execution of new and not previously scheduled activities, as shown in Fig. 1(c) (note the new activity for team member 4), possibly also producing a change in the MANET topology. Specifically, the coordinator transforms the current workflow schema (i.e., the graph related to activity diagram representing the cooperative work) in order to adapt it to the evolving network topology graph.

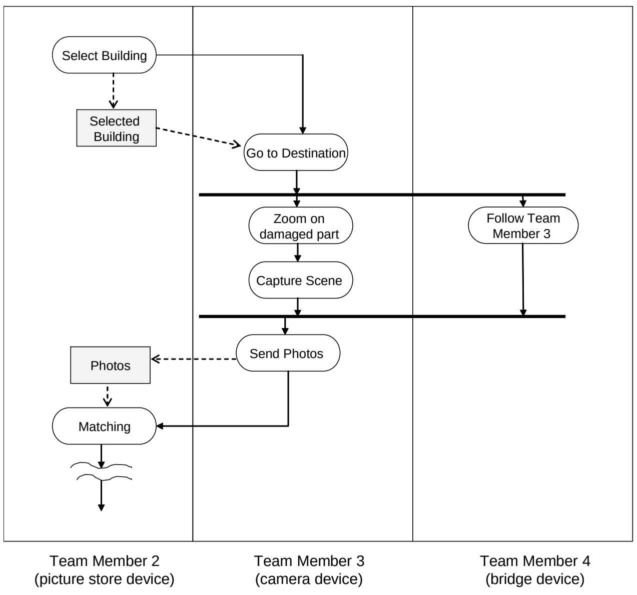
In the literature, topology graphs are a common representation for the description of a MANET configuration [3]; on the other hand, the definition of the workflow is usually expressed through various formalizations, such as Petri nets [64] or Activity Graphs [14], with widely different formal reasoning



(a) Process


Hit Area
Picture Store
Operator
Museum
Precarious Bell-Tower Building
Movement needed to accomplish the task
Camera
Bridge
Church
Team Leader
Movement needed to maintain the network connectivity; should be adaptively driven by the cooperative application

(b) Critical situation and adaptive management



(c) Modified process (detail)

Fig. 1. Adaptive process management in MANETs.

features over such representations. The heterogeneity of the two descriptions makes it difficult to provide an integrated management of the topology, the workflow, and the assignment of tasks to mobile devices.

In this paper, we propose a uniform representation of both network and workflow in terms of an algebraic approach, which are coupled to form complex objects used within a system. The system is based on Algebraic Higher-Order Nets (AHO-NETS [31]), which is a high-level net class combining Petri nets and a suitable higher-order data type part. In particular, the paper describes the integration of place/transition (P/T) systems (P/T-nets taking into account also initial markings), topology graphs, and rule-based transformations [20,32,55] into AHO-NETS, showing its adequacy to model scenarios of cooperative work on MANET. Indeed, we capture together dynamic aspects, due to the high mobility of the mobile devices that we have in MANET contexts, as well as state aspects of workflow execution.

On the basis of the previous formal model, we are implementing a pervasive architecture, in which the coordination layer basically enacts such transformations, with correctness guarantees, and we are going to validate such an architecture in the context of some research projects [1] in the emergency management scenario.

The paper is organized as follows: after describing the workflow architecture constituting the reference framework for cooperative work on MANET in Section 2, in Section 3 we describe the integration of Petri nets, topology graphs, and rule-based transformations into AHO-NETS, whereas in Section 4 the application of the model to our MANET disaster/recovery scenario is shown. In Section 5 relevant research work is presented with respect to the used approach. Finally, in Section 6 we discuss about validation of our approach, offer some considerations and outline future work.

2. Workflow architecture

In Fig. 2, the workflow architecture for supporting cooperative work on MANETs is shown. The different devices of the MANET are equipped with some wireless network interface and specific hardware for calculating distances from neighbors [46,49,50,59] (*Wireless Stack* in the figure); on top of this a *Network Service Interface* [12,13] offers the basic services to upper layers for sending and receiving messages (through multi-hop paths) to/from other devices, by abstracting over the specific routing protocols. Services are offered and may be accessible to other devices, and can be coordinated and composed in a cooperative process. Some of these services are applications that do not require human intervention (e.g., an image processing utility), whereas others act as proxies in front of human actors (e.g., the service for instructing a team member to follow a peer is a simple GUI that alerts the human operator by displaying a pop-up window and emitting a signal).

The coordinator device, conversely, presents the *Predictive Layer* on top of the Network Service Interface, that signals probable disconnections to the upper *Coordination Layer*. The Predictive Layer implements a probabilistic technique [11] that is able to predict if, in the next instant, all devices will still be connected. Indeed, at a given time instant t_i in which all devices are connected, the coordinator device collects all distance information from the other devices (in our assumptions each device sends to the coordinator a message with the distances to its surrounding within-radio-range devices); on the basis of such information the coordinator builds a *next connection graph*, that is the most likely graph at the next time instant t_{i+1} in which the predicted connected and disconnected devices are highlighted. The coordinator layer can thus enact, in the interval $[t_i, t_{i+1}]$, appropriate actions in order to have at t_{i+1} all the devices still connected. The minimal length of the interval may be established in an empirical or experimental manner, by taking into account the time spent by the coordination layer to restructure the workflow schema plus the one needed for receiving all messages with distances at t_{i+1} step.

[1] *MAIS – Multichannel Adaptive Information Systems,* http://www.mais-project.it, and the recently funded IST FP6 *WORKPAD – An Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios* [41].

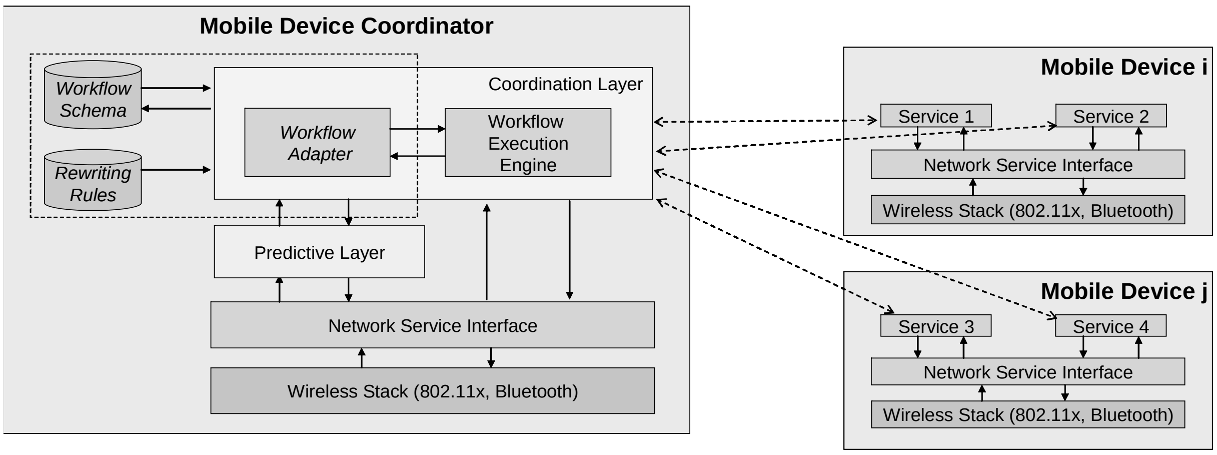

Mobile Device Coordinator
Workflow Schema
Rewriting Rules
Coordination Layer
Workflow Adapter
Workflow Execution Engine
Predictive Layer
Network Service Interface
Wireless Stack (802.11x, Bluetooth)
Mobile Device i
Service 1
Service 2
Network Service Interface
Wireless Stack (802.11x, Bluetooth)
Mobile Device j
Service 3
Service 4
Network Service Interface
Wireless Stack (802.11x, Bluetooth)

Fig. 2. The proposed architecture for supporting cooperative work on MANETs.

The Coordination Layer is in charge of managing those situations when a peer is going to disconnect (e.g., by signaling a specific device to “Follow Peer X”) so as to maintain the network connectivity. Specifically, the *Workflow Adapter* module is in charge of catching disconnection events incoming from the Predictive Layer and, on the basis of the current workflow execution state (taken from the *Workflow Execution Engine* module, which also is in charge of managing activity assignments), applies transformation rules, modifying the workflow schema of the cooperative work (e.g., it adds a new node in the process graph representing the “Follow Peer X” activity). In this paper we model the above described machinery using the AHO-NETS formalism which will be introduced in Section 3. Here, the tokens appearing in the places are transformation rules, topology graphs, and workflow processes.

3. Modeling workflow schemas and their transformations

In this section we present a powerful model, which coordinates restructuring of workflow schemas using rewriting rules as well as workflow execution.

The formalism used for our model provides a two-level modeling technique. The *object level* contains elements manipulated in the process, i.e., workflows schemas, topology graphs, and rewriting rules. Specifically, a team is represented by a tuple consisting of: the workflow schema which is actually executed by it, the topology graph which describes the connection between the team members, and a relation between the workflow schema and the topology graph modeling the assignment of tasks to team members. The rewriting rules are used to transform both the workflow schema and the topology graph, respectively. The *system level*, instead, describes how objects are processed, i.e., how workflow schemas are executed and how workflow schemas and topology graphs are modified under particular conditions.

A schematic view of the system is depicted in Fig. 3. It presents two places, where teams and rewriting rules are stored, and two transitions, for workflow execution and the adaption of workflow schemas and topology graphs, respectively. Transitions are fired under the satisfaction of some conditions, expressed through an algebraic formalism.

We give here an overview of the main formal tools employed in our approach: *Petri nets*, *high level replacement systems*, and *Algebraic Higher-Order Nets*. We report here the formal definition and setting for our model based on the concept of higher-order partial algebras. Higher-order partial algebras [48,

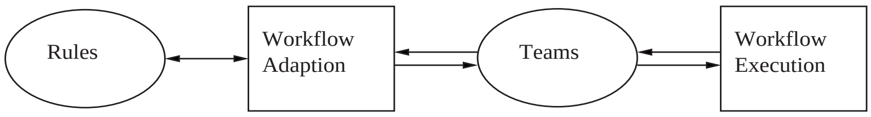


Fig. 3. Schematic view of the system level.

57,69] are close to the set-theoretic semantics of classical algebras [19]. Higher-order functions take functions as parameters and/or yield functions as results, so that the behavior of a number of functions can be summarized in an elegant and abstract way. Conceptually, a higher-order signature consists of basic sorts and operations symbols, where basic sorts are recursively extended to a set of higher-order types including a single type $unit$, product types and function types. Moreover, for each function type $(type_1 \to type_2)$, we have an application symbol $apply^{type_1,type_2} : (type_1 \to type_2) \star type_1 \rightharpoonup type_2$. We use the notation $(op.term)$ as an abbreviation of $apply^{type_1,type_2}(op, term)$, for a term $term$ of type $type_1$ and an operation symbol op of function type $(type_1 \to type_2)$. Note that in the higher-order case with product types and partiality, predicates are fully determined by partial functions into a singleton set, i.e., the domain of definition reflects the trueness of the predicate. The semantics of a higher-order signature Σ is defined by a class of higher-order algebras with partial functions. In general, the carrier of a Σ-algebra A is extended to the set of higher-order types by $A_{unit} := \{()\}$ and $A_{type_1 \star \ldots \star type_n} := \mathrm{A}_{type_1} \times \ldots \times \mathrm{A}_{type_n}$.

First, we provide specific higher-order signatures and corresponding algebras for workflow schemas, topology graphs, and rewriting rules, in order to use them as objects in the system. Finally, starting from the resulting signatures and algebras, we define the corresponding Algebraic Higher-Order Net for our mobile network scenario.

3.1. Workflow schemas

Workflow schemas are modeled through Petri nets with an initial marking, called place/transition (P/T) systems. Petri nets [42,52] are formal tools to describe process behaviors. They can be represented as bipartite directed graphs, with nodes of types *place* and *transition*. For each transition t, nodes which are sources of edges leading into t represent t's *pre-conditions*, while nodes which are targets of edges exiting t are its *post-conditions*. Each place has a certain *capacity*, represented by the number of tokens it can accommodate, and possibly a restriction on their types, where tokens are abstract representations of resources needed by the process. A marking M is an assignment of tokens to each place in the net, in a way compatible with the capacities and the typing of each place. In the following we describe the behavior of a P/T-system, referred to as token game. A transition can occur only if it is enabled in the current marking, i.e., its pre-conditions are satisfied, and produces a legal marking, i.e., the capacity of the post-condition nodes is not exceeded by the insertion of the new tokens. Then each transition consumes some tokens from its pre-condition nodes and produces tokens into its post-condition ones. In this paper, we use the algebraic approach of Petri nets presented in [42], where Petri nets are regarded as free commutative monoids. Instead of using the monodal construction, we will consider the equivalent notion of multisets: more than one token can be in the same place for any given marking and can be moved along a transition by applying the token game.

For example the P/T-system in Fig. 4 describes the workflow cooperatively executed by a specific team (it is the Petri net version of the cooperative process depicted in Fig. 1(a) as a UML Activity Diagram).

We provide a (higher-order) signature WF-SIG, where the WF prefix refers to workflow schemas.

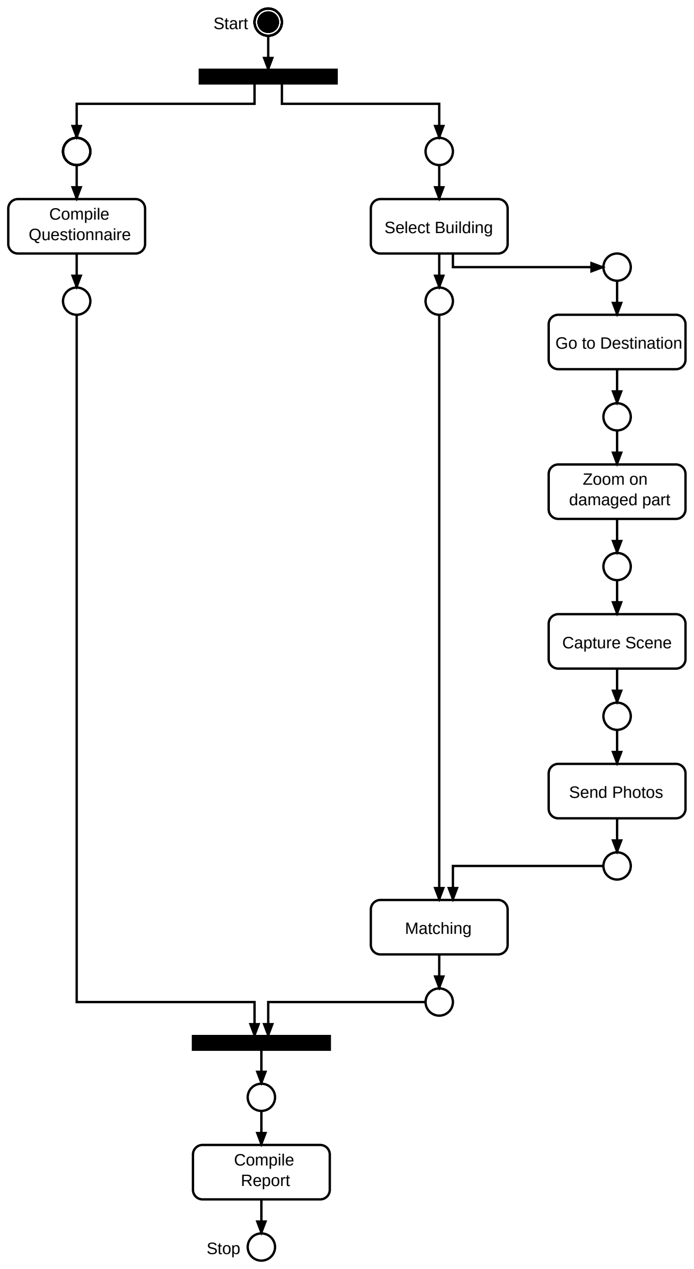


Fig. 4. Cooperative process as P/T-system.

Definition 1. (*WF-SIG Signature*). *We define the higher-order signature WF-SIG* $=$ (*WF-S,WF-OP*) *for workflow schemas as a couple of basic sorts and operation symbols sets by*

WF-SIG $=$

sorts: $Transitions,\ Systems$

opns: $fire:\ (Systems\ \star\ Transitions\ \rightarrow Systems)$

The WF-SIG-algebra *WF-A* is defined by an *WF-S*-sorted carrier set and a suitable realization with the operation symbol in *WF-OP* is associated.

Definition 2. (*WF-SIG Algebra WF-A*). *Given the vocabularies T_0 for transitions and P_0 for places, the carrier of the WF-SIG-algebra WF-A for workflow schemas is defined by the set of all P/T-systems over T_0 and P_0, i.e.,*

$$\textit{WF-A}_{Systems} = \{PN | PN = (P, T, pre, post, M) \textit{ P/T-system}, P \subseteq P_0, T \subseteq T_0\}.$$

A partial function of the WF-SIG-algebra WF-A realizes the token game as described above.

$$(fire.(PN, t))^{\mathrm{WF-A}} = \begin{array}{ll} (P, T, pre, post, M \ominus pre(t) \oplus post(t)) & \textit{if } t \in T, pre(t) \leqslant M \\ \textit{undef} & \textit{else} \end{array}$$

for a P/T-system $PN = (P, T, pre, post, M) \in$ WF-A$_{Systems}$ and a transition $t \in T_0$. $\ominus$ and $\oplus$ are multiset addition/subtraction of elements defined as componentwise addition/subtraction of coefficients.

3.2. Topology graphs

For the description of a MANET configuration we use the graph approach [55], where nodes are labeled over a set of labels. Thus, we provide a (higher-order) signature TG-SIG, where the TG prefix refers to topology graphs. The signature TG-SIG only consists of one basic sort $Graphs$. Given vocabularies L_0 for labels, E_0 for edges, and V_0 for nodes, the carrier set of the TG-SIG-algebra TG-A is defined by the set of all labeled graphs over L_0, E_0, and V_0, i.e.,

$$\begin{aligned}\textit{TG-A}_{Graphs} = \{&TG | TG = (V, E, source, target, nlabel) \text{ (labeled) graph}, \\ &V \subseteq V_0, E \subseteq E_0\}.\end{aligned}$$

For example, the topology graphs in Figs 1(a) and 8 are elements of the carrier set *TG-A*$_{Graphs}$.

3.3. Relation between workflow schemas and topology graphs

In Fig. 5 the dashed lines illustrate the existence of a relation among tasks and team members. This relation is not part of the P/T-system itself, but is an additional information defined by a relation among them. E.g., the task *Compile Questionnaire* is performed by the team member 1 while the task *Select Building* is executed by the team member 2.

In Fig. 6 the higher-order signature Rel-SIG is depicted, where the Rel prefix refers to relation. This signature is based on the union of the WF-SIG and TG-SIG signatures. Moreover, the couple of basic sorts and operation symbols sets is extended in such a way that a connection between specific workflow schemas and topology graphs can be drawn. Here, and in what follows, the symbol Δ refers to the usual interpretation as diagonal duplication and π_i as projection on the i-th component.

Definition 3. (*Rel-SIG Algebra Rel-A*). *Given the vocabularies T_0 for transitions and V_0 for nodes, the carrier of the Rel-SIG-algebra Rel-A for relations between workflow schemas and topology graphs is defined by*

$$Rel - A_{SetTrans} = \mathcal{P}(T_0), Rel - A_{SetNodes} = \mathcal{P}(V_0), Rel - A_{Rel} = \mathcal{P}(T_0 \times V_0).$$

The partial operations of the Rel-SIG-algebra Rel-A are defined by

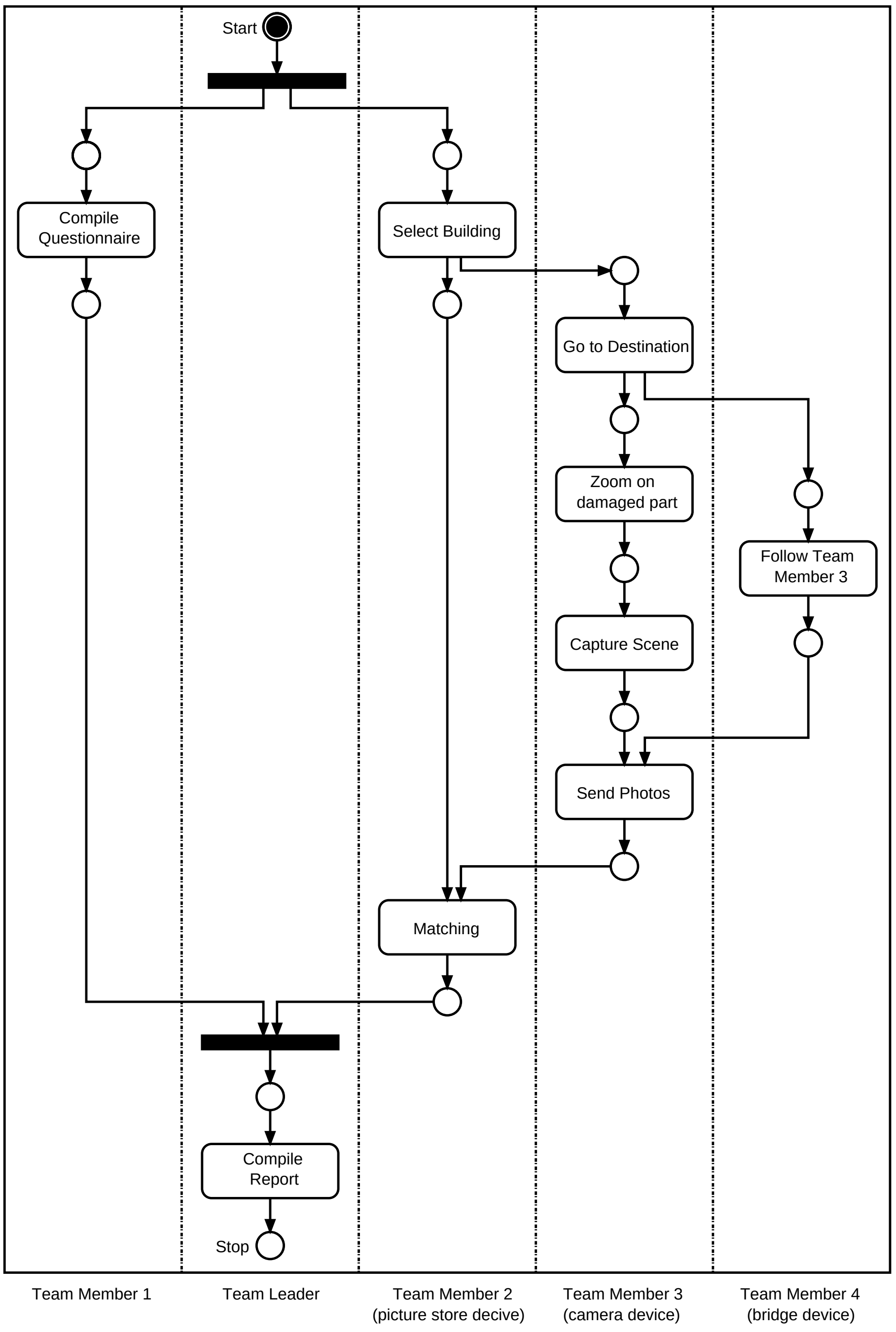


Fig. 5. P/T-system with relation to topology graph.

– *the projection of transitions for P/T-systems, i.e.,* $(trans.(PN))^{\mathrm{Rel-A}} = T$ *for a P/T-system* $PN = (P, T, pre, post, M) \in$ *WF-A*$_{Systems}$*,*
– *the projection of nodes for* (*labeled*) *graphs, i.e.,* $(nodes.(TG))^{\mathrm{Rel-A}} = V$ *for a* (*labeled*) *graph* $TG = (V, E, source, target, nlabel) \in$ *TG-A*$_{Graphs}$*,*

$$\begin{tikzcd}
Systems \arrow[d, "trans"'] & & & & Graphs \arrow[d, "nodes"] \\
SetTrans \arrow[dd, "\Delta"'] & & & & SetNodes \arrow[dd, "\Delta"] \\
& & Rel \arrow[llu, "dom"'] \arrow[rru, "range"] & & \\
SetTrans \star SetTrans \arrow[rrd, "=_T"'] & & & & SetNodes \star SetNodes \arrow[lld, "\subseteq_N"] \\
& & unit & &
\end{tikzcd}$$

Fig. 6. Higher-order signature Rel-SIG.

– *the usual interpretation of equivalence and subset operations on sets for the operation symbols* $=_T$ *and* $\subseteq_N$*,*
– *the usual interpretation of the domain and range of relations, i.e.,*

$$(dom.(C))^{\mathrm{Rel-A}} = \{t \in T_0 | \exists v \in V_0 : (t, v) \in C\} \textit{ and}$$
$$(range.(C))^{\mathrm{Rel-A}} = \{v \in V_0 | \exists t \in T_0 : (t, v) \in C\}$$

for $C \in$ *Rel-A*$_{Rel}$.

3.4. Rewriting rules

High-level replacement systems [16,17] are a generalization of the algebraic (in the double push-out *DPO* [21] version) approach to graph transformation, by which transformation rules are defined through morphisms acting on objects of a category CAT. Without entering into the technicalities of DPO, suffice it to say that a rule r is constituted of three components: an antecedent L, a consequent R, and an interface I which describes what is to be preserved through the application of a rule. Formally, a rule $r = (L \overset{i_1}{\leftarrow} I \overset{i_2}{\rightarrow} R)$ is defined by a span of morphisms i_1 and i_2. Applying a rule means substituting a match $m : L \rightarrow O$ for L in a source object O of CAT with a match of R producing a target object O' again in CAT. This process is called a direct transformation $O \overset{(r,m)}{\Longrightarrow} O'$ from O to O' via a rule r and match m. Moreover, rules have no side effects, i.e., O' differs from O only for the removal of elements present in L but not mentioned in I, and the insertion of elements mentioned in R, but not present in L.

Differently from the graph transformation approach [55] where rules and transformations describe dynamic behavior, in the net transformation approach [20,32] rules and transformations are used to represent stepwise development of structure nets. These kinds of transformations are considered as vertical structuring techniques, known as rule-based transformations. We are going to use both kinds of transformation in our model: rule-based graph transformation and rule-based net transformation; the latter is used to evolve a cooperative workflow of a specific team, while the former is used to reflect the dynamic structure of the topology graph.

Applying a rule to a P/T-system informally means replacing a subnet specified by the antecedent of the rule with a net specified by the consequent of the rule. Thus, the P/T-systems *PN1* and *PN2* in Fig. 8 describe the workflow schema before and after the application of the rule *Rule1* =(*L1* $\leftarrow$ *I1* $\rightarrow$ *R1*) (with match $in1 : L1 \rightarrow PN1$) that modifies the net structure, so as to insert activities devoted to maintaining the network topology ("to move afterwards the going-out-of-range" operator/device in our example).

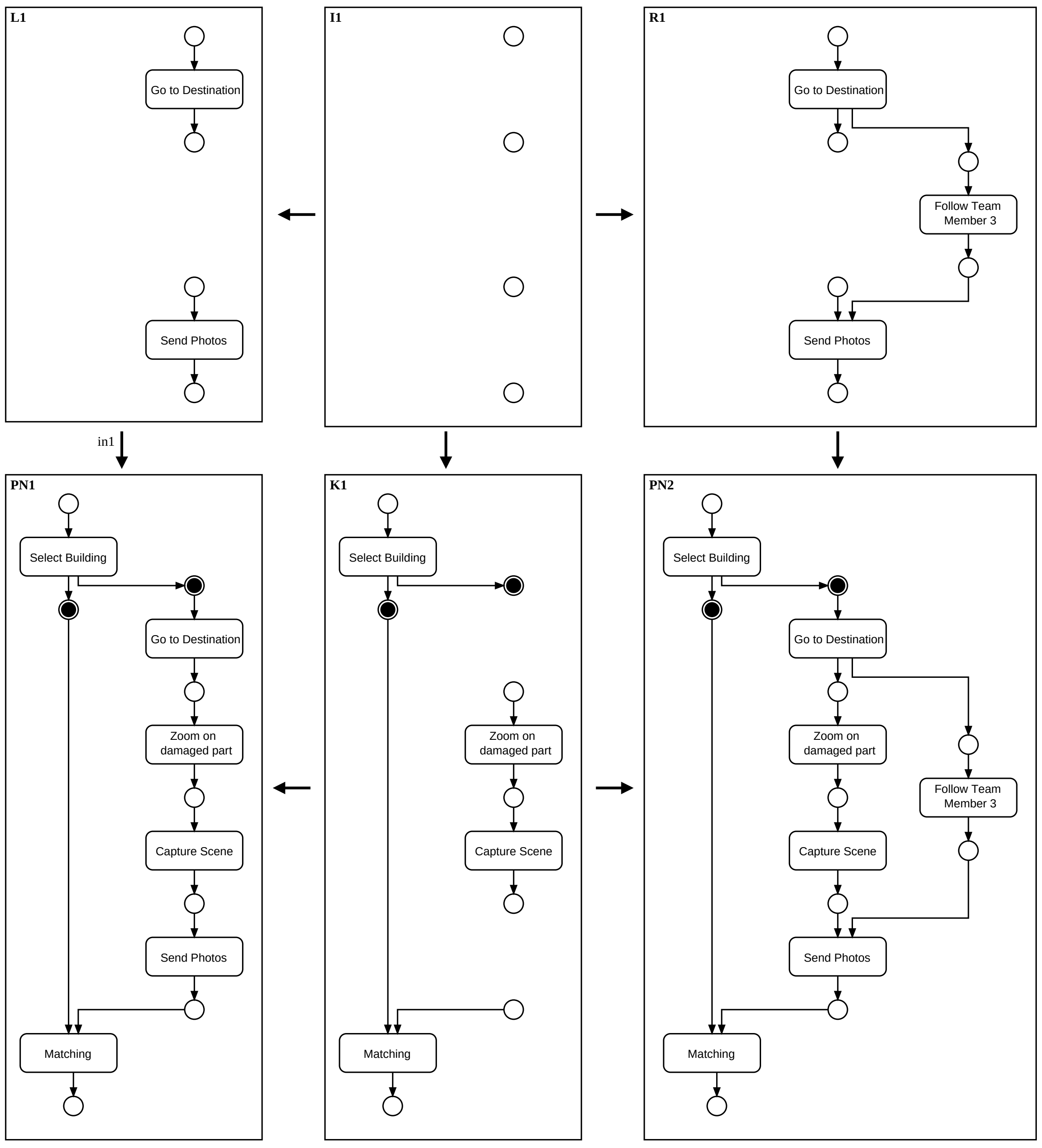


Fig. 7. Rule-based net transformation.

Here, *Rule1* reflects a situation in which team member 3 has been disconnected from others, because of the his/her movement. Hence, a movement of team member 4 is needed to maintain the network connectivity, i.e., he/she has to follow the camera device. The right-hand side of *Rule1* introduces a specific task for team member 4 to react in a suitable way in this situation.

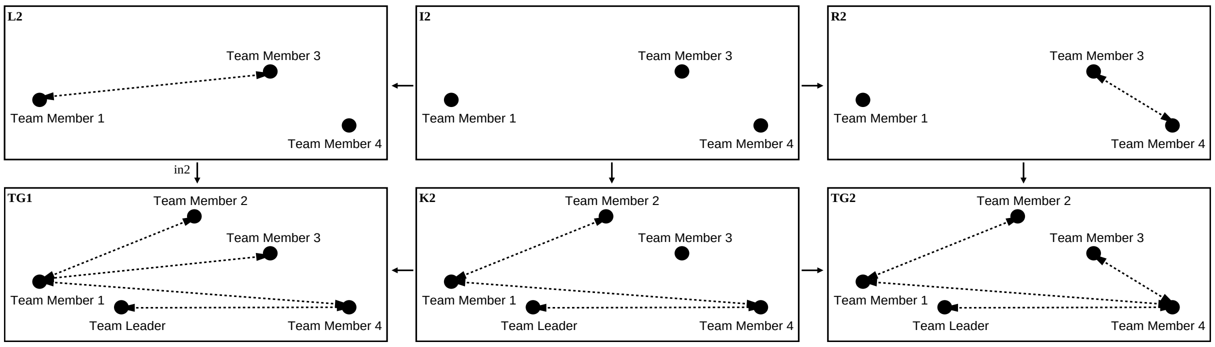


Fig. 8. Rule-based graph transformation.

$$\begin{tikzcd}
GraphRules & & GraphMor \arrow[dd, "cod"] \\
& GraphRules \star GraphMor \arrow[ul, "\pi_1"'] \arrow[ur, "\pi_2"] \arrow[dr, "transform_G"'] & \\
& & Graphs \arrow[dl, "\Delta"] \\
& Graphs \star Graphs \arrow[d, "=_G"] & \\
& unit &
\end{tikzcd}$$

Fig. 9. Higher-order signature GR-SIG.

Analogously, in Fig. 8 the rule *Rule2* =(*L2* ← *I2* → *R2*) is applied to the topology graph *TG1*, such that there is a direct transformation $TG1 \overset{Rule2}{\Longrightarrow} TG2$. Here, *Rule2* reflects a situation in which team member 3 is disconnected from team member 1. To maintain the network connectivity a new link between team member 3 and team member 4 is established in the right-hand side of the rule *Rule2*.

We provide a (higher-order) signature GR-SIG for rule-based graph transformation, where the GR prefix refers to graph rules. The signature is depicted in Fig. 9 and is based on the signature TG-SIG for topology graphs. Especially the intention of the operation symbol $transform_G$ is a realization in such a way that it exactly computes the application of one graph rule to a specific topology graph in the sense of rule-based graph transformation realizing the adaption of the topology graph.

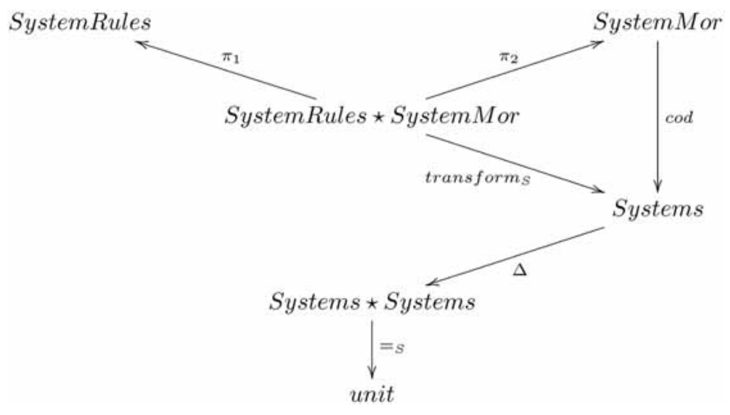
Definition 4. (*GR-SIG Algebra GR-A*). *The carrier of the GR-SIG-algebra GR-A for rule-based graph transformation is defined by the set of all graph morphisms for TG-A*$_{Graphs}$, *i.e.,*

$$GR\text{-}A_{Mor} = \{m | m : TG_1 \to TG_2 \text{ graph morphism with } TG_1, TG_2 \in TG\text{-}A_{Graphs}\}$$

and the set of all graph rules, i.e.,

$$GR\text{-}A_{GraphRules} = \{r | r = (L \overset{i_1}{\leftarrow} I \overset{i_2}{\rightarrow} R) \text{ graph rule with inclusions } i_1, i_2 \in GR\text{-}A_{Mor}\}.$$

The partial operations of GR-SIG-algebra GR-A are defined by



$$\begin{tikzcd}
SystemRules & & SystemMor \arrow[dd, "cod"] \\
& SystemRules \star SystemMor \arrow[ul, "\pi_1"'] \arrow[ur, "\pi_2"] \arrow[dr, "transform_S"'] & \\
& & Systems \arrow[dl, "\Delta"] \\
& Systems \star Systems \arrow[d, "=_S"] & \\
& unit &
\end{tikzcd}$$

Fig. 10. Higher-order signature SR-SIG.

– *the rule-based transformation of graphs as described above, i.e.,*

$$(transform_G.(r,m))^{\text{GR-A}} = \begin{matrix} \textit{TG'} & \textit{if } r \textit{ is applicable at match } m \\ \textit{undef} & \textit{else} \end{matrix}$$

where for $r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R) \in$ *GR-A*$_{GraphRules}$, $(m : L \to TG) \in$ *GR-A*$_{Mor}$ *and* r *is applicable at match* m, *we have a direct transformation TG* $\overset{(r,m)}{\Longrightarrow}$ *TG'*,

– *the usual interpretation of the codomain of morphisms, i.e.,*

$$(cod.(m))^{\text{GR-A}} = TG_2 \textit{ for } (m : TG_1 \to TG_2) \in \textit{GR-A}_{Mor},$$

– *and the usual interpretation of equivalence on graphs for the operation symbol* $=_G$.

Analogously, we provide a (higher-order) signature SR-SIG for rule-based net transformation, where the SR prefix refers to P/T-system rules. The signature is depicted in Fig. 10 and is based on the signature WF-SIG for workflow schemas.

Definition 5. (*SR-SIG Algebra SR-A*). *The carrier of the SR-SIG-algebra SR-A for rule-based net transformation is defined by the set of all P/T-system morphisms for WF-A*$_{Systems}$, *i.e.,*

$$\textit{SR-A}_{Mor} = \{m | m : PN_1 \to PN_2 \textit{ morphism with } PN_1, PN_2 \in \textit{WF-A}_{Systems}\}$$

and the set of all P/T-system rules, i.e.,

$$\textit{SR-A}_{SystemRules} = \{r | r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R) \textit{ P/T-system rule with inclusions } i_1, i_2 \in \textit{SR-A}_{Mor}\}.$$

The partial operations of SR-SIG-algebra SR-A are defined by

– *the rule-based transformation of P/T-systems as described above, i.e.,*

$$(transform_S.(r,m))^{\text{SR-A}} = \begin{cases} \textit{PN'} & \textit{if } r \textit{ is applicable at match } m \\ \textit{undef} & \textit{else} \end{cases}$$

where for $r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R) \in$ *SR-A*$_{SystemRules}$, $(m : L \to PN) \in$ *SR-A*$_{Mor}$ *and* r *applicable at match* m, *we have a direct transformation PN* $\overset{(r,m)}{\Longrightarrow}$ *PN'*,

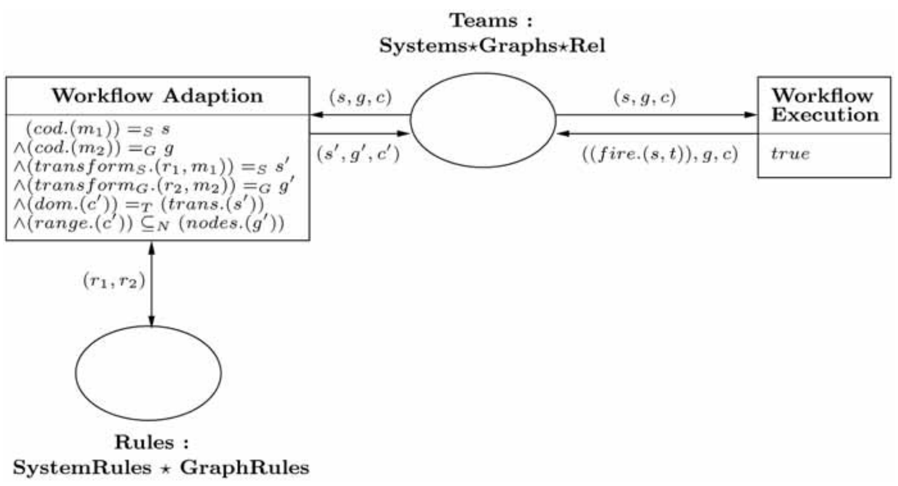


Fig. 11. System level as AHO-NET.

– *the usual interpretation of the codomain of morphisms, i.e.,*

$$(cod.(m))^{\mathrm{SR-A}} = PN_2 \textit{ for } (m : PN_1 \to PN_2) \in \textit{SR-A}_{Mor},$$

– *and the usual interpretation of equivalence on P/T-systems for the operation symbol* $=_S$.

3.5. Data type part for MANETs

The data type part for mobile network scenarios is given by the higher-order signature MANET-SIG and corresponding algebra *MANET-A*, which comprises the signatures and algebras defined above plus an operation symbol $\wedge$ to state the logical connector of conjunction and an operation symbol *true* meant to be the logical value of trueness.

3.6. System level

The formal approach of Algebraic Higher-Order (AHO) Nets [31] is a novel modeling technique, which comprises the advantages of the well-researched high-level net classes of Coloured Petri Nets [38] and Algebraic High-Level Nets [47,18] to support the flexibility and adaptability in an extensive way. Moreover, AHO-NETS can be seen as a formal approach for Higher-Order Object Nets [28], which are well-established for workflow modeling, but have been mainly described informally so far. High-level net classes are obtained by combining Petri nets with an appropriate data type part. While the net structure of AHO-NETS is graphically modeled by Petri nets, the concept of higher-order partial algebras turned out to be well-suited data type part for AHO-NETS. The combination of these techniques is achieved by the inscription of net elements with terms over the given data type part.

In Fig. 11 we show the AHO-NET representing the system level of our model for mobile network scenarios. It consists of two transitions, *Workflow Execution* and *Workflow Adaption*, and two places, *Teams* and *Rules*. In particular, the transition *Workflow Execution* indicates the normal functioning of the system, where no transformation of the topology graph or the workflow schema is involved, and the system evolves according to the law for computing the follower marking in the P/T-system defining the cooperative workflow. Thus this transition models the execution aspect of workflow schemas termed by the net inscription $(fire.(s,t))$ in the AHO-NET. Due to the firing-condition *true*, no constraint has to be respected.

In the transition labeled *Workflow Adaption*, P/T-system rules for transforming the workflow schema are involved to obtain a new P/T-system, while graph rules are used for the rearrangement of the topology graph. Thus, the firing of the transition *Workflow Adaption* realizes both rule-based net transformation and rule-based graph transformation, respectively, as indicated by the net inscriptions $(transform_S.(r_1, m_1))$ and $(transform_G.(r_2, m_2))$. The rules presiding to such transformations are considered to be never removed from the rule collection as indicated by the double arrow between the place *Rules* and the transition *Workflow Adaption*. Hence, they are consumed and immediately restored in their place at each rule application. The firing-condition $(cod.(m_1)) =_S s$ ensures that a P/T-system rule is applied to the workflow schema of a given team, and analogously the firing-condition $(cod.(m_2)) =_G g$ ensures that a graph rule is applied to the topology graph of the given team. The firing-conditions $(transform_S.(r_1, m_1)) =_S s'$ and $(transform_G.(r_2, m_2)) =_G g'$ denote the resulting workflow schema and the resulting topology graph, respectively, after the application of the corresponding rules. Finally, the firing conditions $(dom.(c')) =_T (trans.(s'))$ and $(range.(c')) \subseteq_N (nodes.(g'))$ ensure that the connection between the resulting workflow schema and the resulting topology graph take into account the tasks and the team members after the transformation.

It is to be noted, also, that the place *Teams* is used as a precondition for both the workflow execution and the transformation activities. Hence, the two activities are in conflict, thus preventing the possibility of a concurrent transformation of both the workflow process and its marking. In this way, there is no need to explicitly model the suspension of the workflow process while transformations are occurring.

Definition 6. (*AHO-NETS in Mobile Network Scenarios*). *We define an AHO-NET with respect to our mobile network scenario as the following tuple:*

$$(MANET-SIG, \text{MANET-A}, P, T, pre, post, cond, type)$$

with

- *the data type part given by the higher-order signature MANET-SIG and the corresponding higher-order partial algebra MANET-A,*
- *the net structure given by a set of places $P = \{Teams, Rules\}$ with typing*

  $type(Teams) = Systems \star Graphs \star Rel$ *and*
  $type(Rules) = SystemRules \star GraphRules$

  and a set of transitions $T = \{\text{Workflow Execution, Workflow Adaption}\}$ with corresponding environments $pre, post$ and $cond$, where the pre- and post-condition functions pre and $post$ assign net inscriptions and places to each transition; the firing-condition function $cond$ assigns one predicate to each transition, which is a constraint to be respected, e.g.

  $pre(\textit{WorkflowExecution}) = ((s, g, c), Teams)$
  $post(\textit{WorkflowExecution}) = ((fire.(s,t)), g, c), Teams)$*, and*
  $cond(\textit{WorkflowExecution}) = \text{true}$.

4. AHO-NET in Mobile Network Scenario

In this section we will illustrate the application of the algebraic model defined in Section 3 to our disaster/recovery mobile scenario in terms of object and system level.

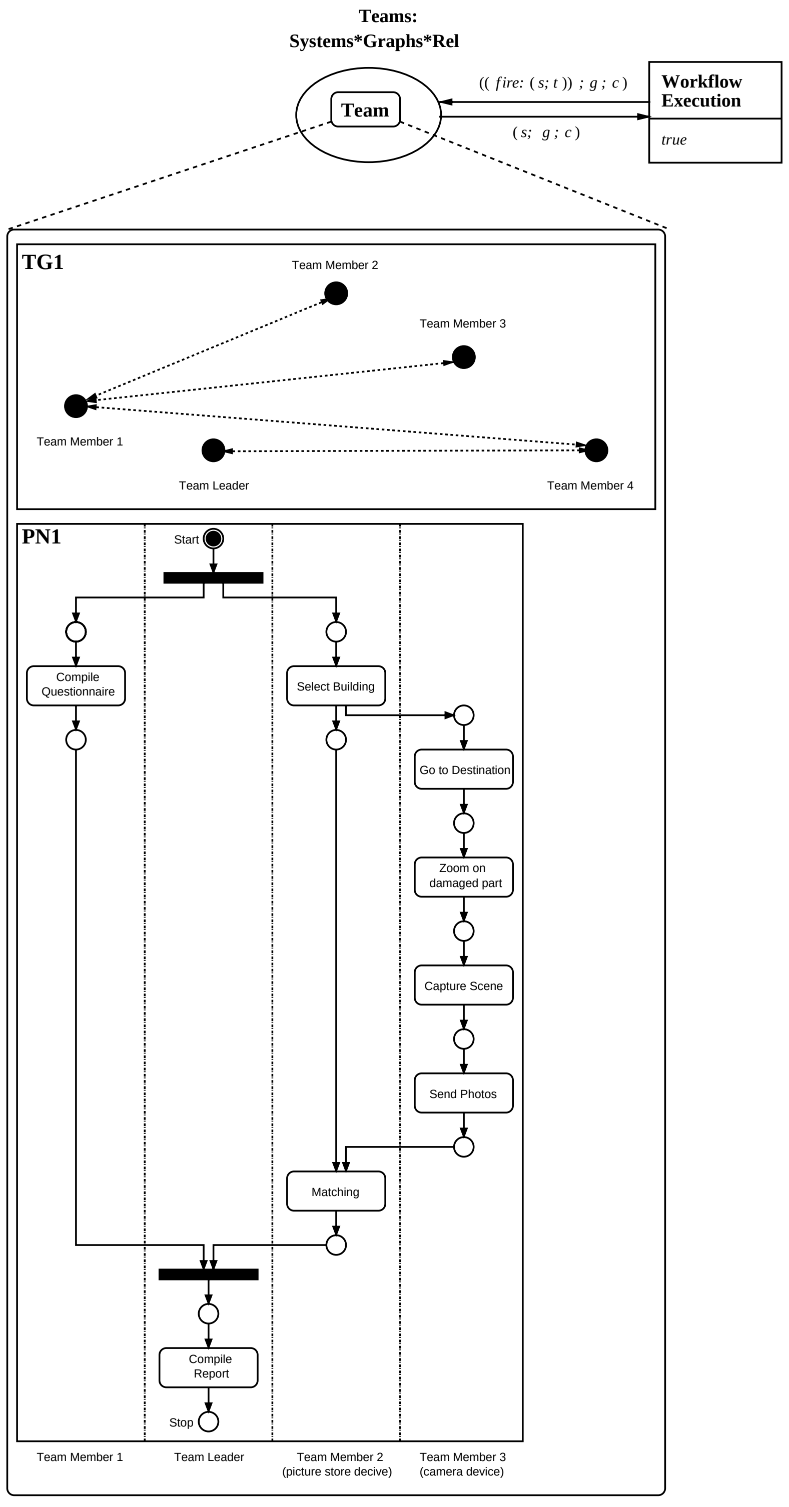


Fig. 12. AHO-NET with one team object.

First, we report on the behavior of the AHO-NET structure above defined; with the symbol $Var(t)$ we indicate the set of the variables of the transition t, i.e., the set of all variables occurring in pre- and post-condition and in the firing-condition of t. The marking determines the distribution of objects in the AHO-NET. Formally, the marking M of an AHO-NET N with the set of places P consists of data values, which are elements from a given higher-order algebra A. For each place, all objects must belong to a specified type and more than one token can be in the same place for any given marking.

Data values can be modified during the firing of transitions. Intuitively, a data value can be moved along a transition, if the firing-conditions are fulfilled. The follower marking is computed by the evaluation of net inscriptions in a variable valuation $v : Var(t) \rightarrow A$. The transition t is enabled in a marking M, if and only if (t, v) is consistent, i.e., if the evaluation of the net inscription is defined. Note that the operations in the data type part are allowed to be partial functions. Then the follower marking after the firing of the transition t is defined by removing tokens corresponding to the net inscription in the pre-condition of t and adding tokens corresponding to the net inscription in the post-condition of t.

4.1. Workflow execution

Initially, there is one team represented by the token *Team* in the place *Teams* in Fig. 12. The team object is a tuple consisting of the P/T-system *PN1*, the topology graph *TG1*, and the relation *C1* among tasks of *PN1* and nodes of *TG1*, where the latter is illustrated by the dashed lines. To start the activities of team member 1 and team member 2, we use the transition *Workflow Execution* of the AHO-NET in Fig. 11. First, the variables $s, g,$ and c are assigned to the token *Team* and the variable t to $t_0 \in T_0$ where T_0 is a given vocabulary of transitions. Because no constraint has to be respected, the evaluation of the term $(fire.(s, t))$ computes the follower marking of the P/T-system (i.e., tokens in the pre-conditions of transitions *Compile Questionnaire* and *Select Building*) and we obtain the new P/T-system *PN1'* depicted in Fig. 13. In the next step, the task *Select Building* of the P/T-system *PN1'* is executed by the firing of the transition *Workflow Execution* resulting in the P/T-system *PN1"* in Fig. 14, etc.

4.2. Workflow adaptation

Next, we model the movement of team member 4 to predict a situation of disconnection. The workflow schema has to be extended by a task to follow team member 3, while the topology graph has to be transformed to ensure a path among devices. For this reason, in Fig. 14, the token (*Rule1*,*Rule2*) in the place *Rules* consists of a P/T-system rule *Rule1* and a graph rule *Rule2*. Formally, we apply *Rule1* to the P/T-system *PN1"* (see Fig. 7) and *Rule2* to the topology graph *TG1* (see Fig. 8) by firing the transition *Workflow Adaption*. We have to give an assignment v for the variables of the transition *Workflow Adaption*, i.e., variables $s, s', g, g', c, c', r_1, r_2, m_1,$ and m_2. The assignment v is defined by

– $v(s) =$ *PN1"*$, v(g) =$ *TG1*$, v(c) =$ *C1*$, v(r_1) =$ *Rule1*$, v(r_2) =$ *Rule2* (see Fig. 14),
– $v(m_1) =$*in1* (see match morphism *in1* : *L1* $\rightarrow$ *PN1* in Fig. 7),
– $v(m_2)$=*in2* (see match morphism *in2* : *L2* $\rightarrow$ *TG1* in Fig. 8),
– $v(s') =$*PN2*, $v(g) =$ *TG2*, and $v(c') =$ *C2* (see Fig. 15).

The firing-conditions $(cod.m_1) =_S s$ and $(cod.m_2) =_G g$ make sure that the codomain of *in1* is equal to *PN1"* and the codomain of *in2* is equal to *TG1*. The left hand side of the firing-condition $(transformations_S.(r_1, m_1)) =_S s'$ computes the direct transformation shown in Fig. 7, i.e., we delete in a first step the transitions *Go to Destination* and *Send Photos* from the P/T-system *PN1"* and add in a second step the transitions *Go to Destination*, *Send Photos*, and *Follow Team Member 3* together with their

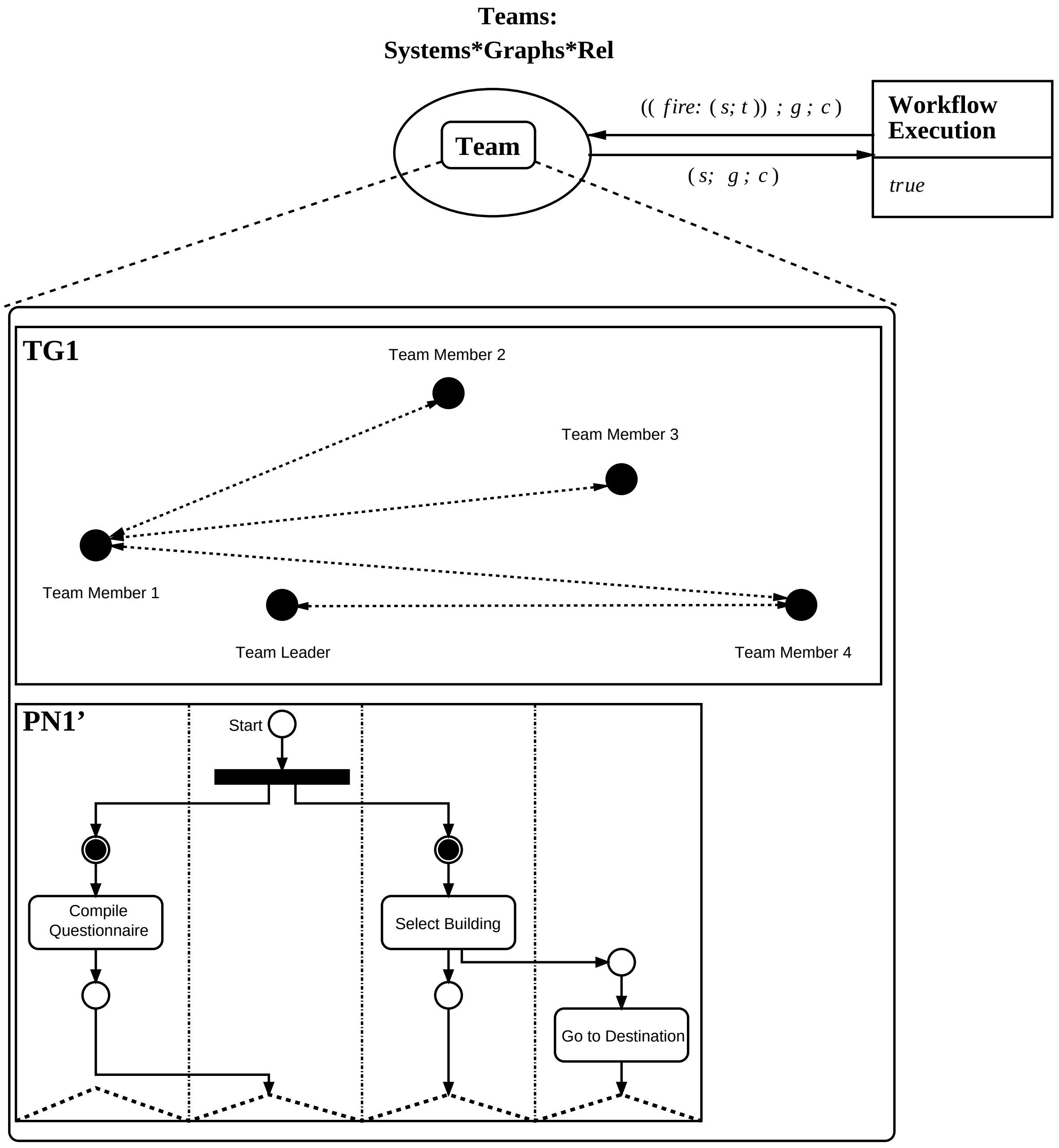


Fig. 13. AHO-NET with team object after workflow execution.

(new) environments. Thus, the firing-condition checks if the application of *Rule1* to the P/T-system *PN1"* leads to the P/T-system *PN2*. Analogously, the firing-condition $(transformation_G.(r_2, m_2)) =_S s'$ checks if the application of *Rule2* to the topology graph *TG1* leads to the direct transformation as shown in Fig. 8. Finally, the firing-conditions $(dom.(c')) =_T (trans.(s'))$ and $(range.(c')) \subseteq_N (nodes.(g'))$ state the new relation *C2* among the transitions of the P/T-system *PN2* and the nodes of the topology

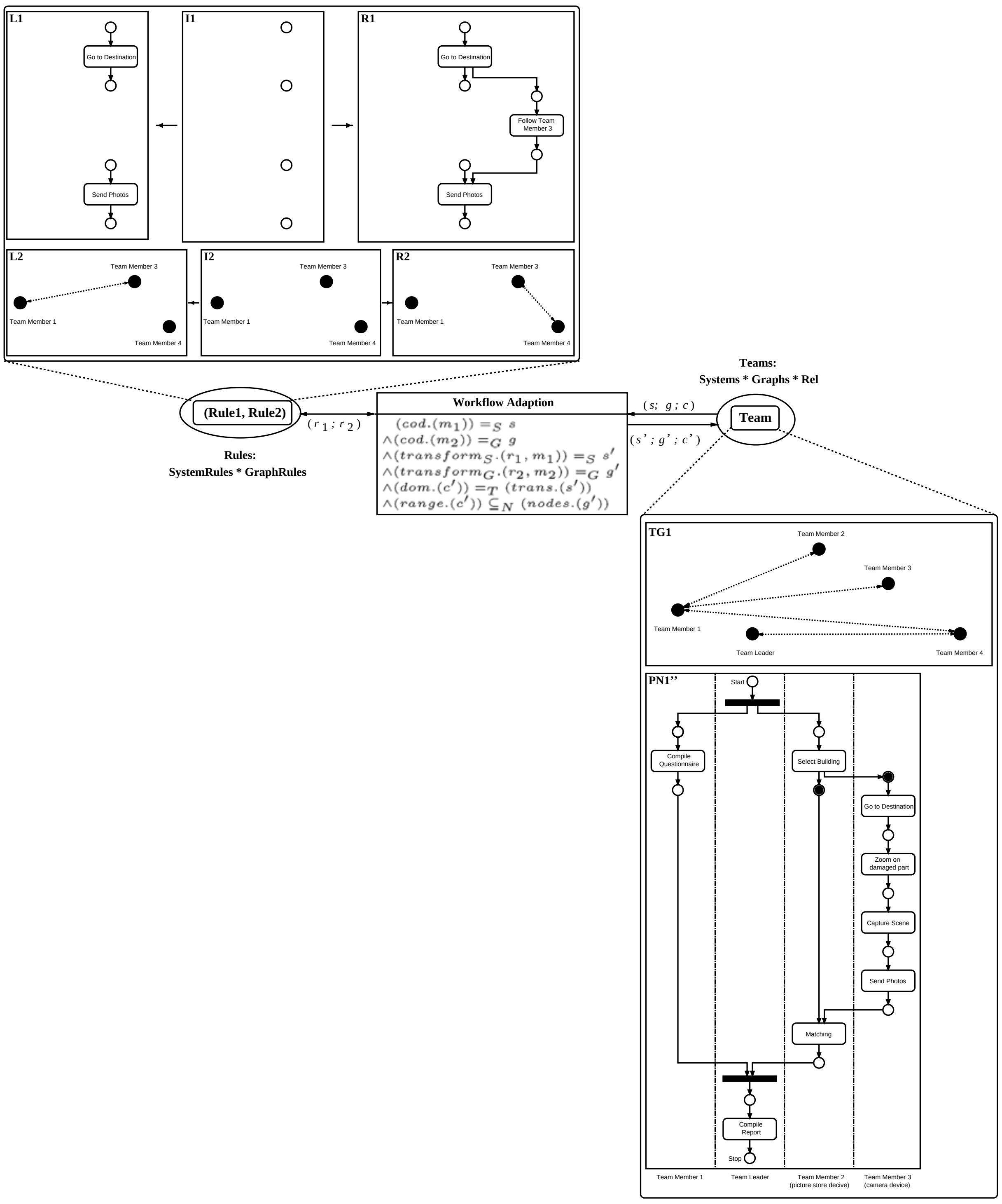


Fig. 14. AHO-NET with rule objects.

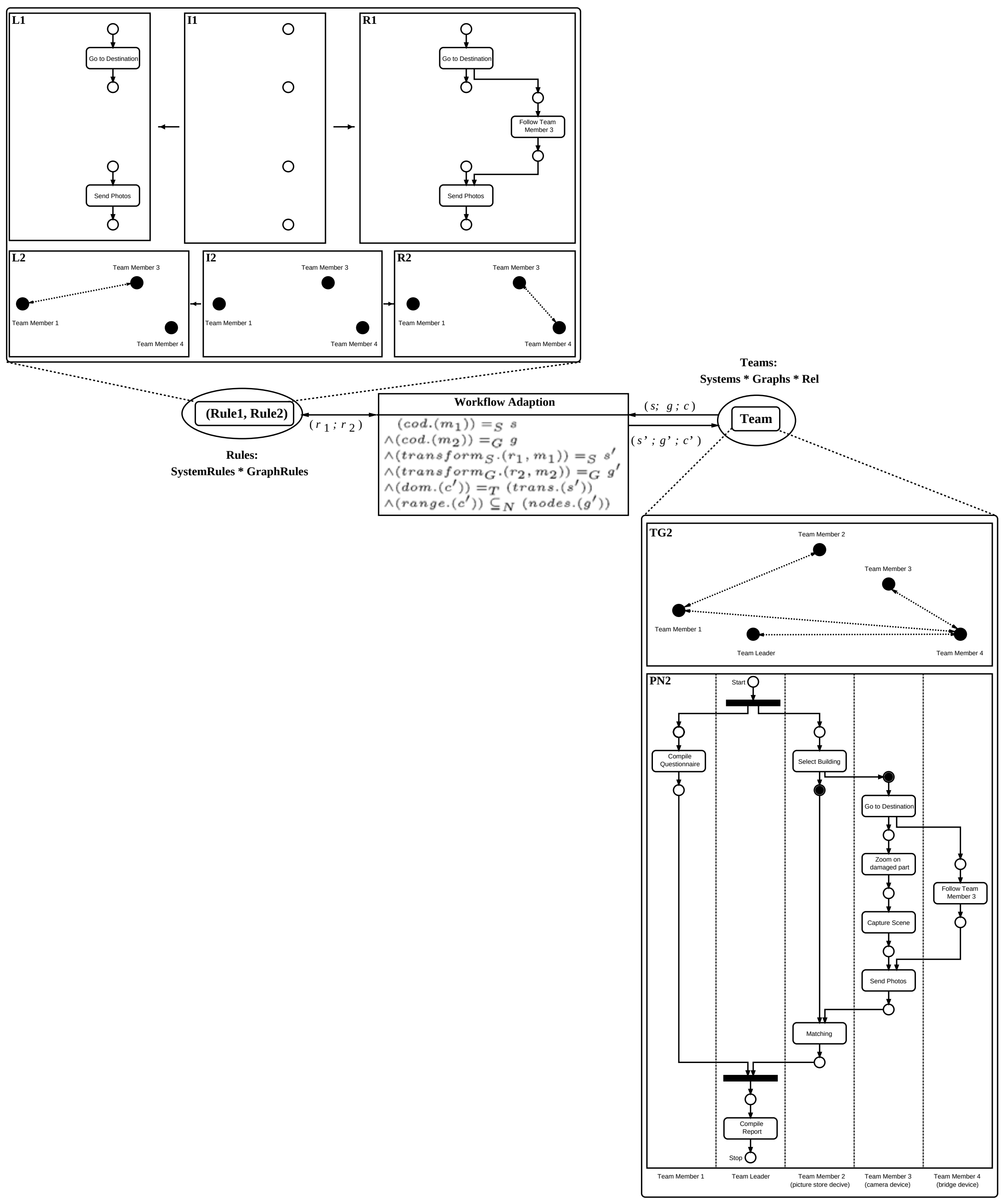


Fig. 15. AHO-NET with objects after rule based transformation.

graph *TG2*. After the firing of the transition *Workflow Adaption* we get the new team object in Fig. 15 consisting of the workflow schema *PN2*, the topology graph *TG2*, and the relation *C2*.

Summarizing, the effect of firing the transition *Workflow Adaption* with assignments of variables as discussed above is the transformation of the team object *Team* by using the P/T-system rule *Rule1* and the graph rule *Rule2*, i.e., the removal of the P/T-system *PN1"*, the topology graph *TG1*, and the relation *C1* from place *Teams* and the adding of a new team object consisting of the P/T-system *PN2*, the topology graph *TG2*, and the relation *C2* to the place *Teams*.

We can add further firing-conditions concerning the relation between workflow schemas and topology graphs, e.g., we can force the system so that relation *C2* reuses the "old" information of *C1*. In this paper workflow schemas are modeled by a special subclass of P/T-systems, called workflow (WF) nets [64]. A WF-net is a P/T-system with two special place, *start* and *stop,* where place *start* is the only source (i.e., it cannot be used in a postcondition) and place *stop* is the only sink (i.e., it is not precondition to any transition). Moreover, by adding a link between the *stop*-place and the *start*-place the resulting P/T-system must be strongly connected. In our current version rule-based net transformations using P/T-system rules are not guaranteed to produce a WF-net as the resulting P/T-system. However, this can be ensured by further firing-conditions, which check if the resulting P/T-system is a WF-net. Obviously, there may be further objects for the same system, so as to reflect the presence of other teams as well as transformations of workflow schemas and topology graphs. We do not model here the mobility aspects of the rules, i.e., the possibility of transferring specific rules to different members of the team, which could however be easily accommodated in our framework by adding a new transition in the system level AHO-NETk. Furthermore, one could also need modification of rules during system execution as proposed in [35].

5. Related work

The adaptation of workflow to possible exceptional cases or to changes in management policies has been soon recognised as a necessity for practical uses of workflow systems. Solutions to the related problem of dynamic change – i.e., how to transform the workflow without suspending all its instances or waiting for all instances to have come to conclusion – and to the possibility of creating inconsistent states of workflow instances have been studied in formal frameworks, typically defined by Petri nets.

The pioneering work of [22] reasons on Petri nets, by identifying safe and unsafe states and paths based on the type of transformation the net has to incur, typically sequentialisation and parallelisation of activities. With the rise of UML, statecharts and activity diagrams have also received some attention as languages for workflow modeling [43,45] and their relations with variants of Petri nets have been investigated, notwithstanding limitations due to some insufficiencies of the defined semantics [24].

Event-Condition-Action rules [68] have largely been used in workflow modeling, as they allow the connection of a workflow engine with active database systems, so that aspects of transactionality or access to resources can be left to the database, leaving the workflow modeller mainly to deal with the specification of control flow. However, they are hard to manipulate directly, so that their formal foundation is typically mapped to variants of Petri nets or temporal logic.

A different, not transformational, approach to workflow modeling is represented by the use of extensions of regular expressions, as derived from studies on synchronisation of parallel programs [6]. Typically, rather than describing all possible compositions of intercommunicating workflows, these expressions describe sets of permissible execution sequences of actions. Interaction graphs extend these notions to paths on graphs [30] to describe fully deterministic behaviours, and can be used to check conditions for safe transformations in adaptive contexts [54].

Therefore, current approaches supporting adaptive workflows are based on different process models. Very often, the solutions offered by them are dependent on the expressiveness as well as on the formal and operational semantics of the used formalism. In [36] a classification of models is reported, with respect to their operational semantics and the evaluation strategies applied for executing workflow process instances during runtime. The first strategy uses only one type of (control flow) token passing through each workflow instance (*True-Tokens*). The other strategy is based on two types of tokens: *True-* and *False-Tokens*. Simplistically, True-Tokens trigger tasks that are to be executed next and False-Tokens describe skipped tasks. Formalisms which solely use True- Tokens include Petri nets-based models. Examples of them are: WF nets [63,65], Flow nets [22,23], and MILANO nets [1,2] (that is, marked, acyclic Free-Choice Petri nets). All those Petri nets-based approaches abstract from internal task states, i.e., they only differentiate between activated and non-activated transitions. Moreover, in such models data flow issues are excluded or not explicitly considered, and resource availability and resource (re-)assignment are not taken into account during the state evolution of the process instance.

Approaches which use True- and False-Tokens to represent skipped tasks or skipped execution branches can be found in the area of graph-/activity-based models [7,37,51,56,67]. As opposed to the Petri nets-based approaches, they distinguish between the different states a task may go through. Generally, the initial status of a task is set to `NotActivated`. It changes to `Activated` when all preconditions are met. Task execution is then either started automatically or corresponding worklist entries are generated. When starting task execution, its status changes to `Running`. Finally, at successful termination, status passes to `Completed`. In addition, some of the models assign status `Skipped` to tasks belonging to non-selected execution branches. Usually, an execution history is also maintained for each process instance; for each started task X the values of process data elements read by X and for each completed task Y the values of data elements written by Y, are logged. Those approaches can be further divided according to the way they represent the tokens. One possibility is to gain them from *execution histories* (e.g., WIDE [7] and TRAMs [37] approaches), which log events like task start and completion. Alternatively, special (*model-inherent*) task markings, which represent a consolidated view on the history logs, can be used (e.g., ADEPT [51], Breeze [56] and WASA$_2$ [67] approaches). Differently to Petri nets-based approaches, all these formalisms consider data flow issues, but also in them resource availability and resource (re-)assignment are not modeled during the state evolution of the process instance.

A more detailed discussion of all these approaches can be found in [53].

With respect to the above classification, the approach presented in this work can be group together with those using a True-Tokens strategy. Additionally, the proposed model is able to capture dynamic evolutions of the process instance, both at the state and at the structure level, and resource availability as well as resource (re-)assignment at the same time. This is achieved by modeling the resource set as a graph and using a relation (mapping) between workflow process (Petri net) and topology graph. However, data flow issues are not currently considered in our model, but will be studied in future work. The approach is situated in the framework of the algebraic approach to rewriting, originally proposed for graph grammars [21], which has been widely used and extended to several formal systems, such as term rewriting, graph transformations in general, higher-order structures, and put to work in several contexts (for surveys on the approach and applications, see [9,15]). In particular, the DPO approach has been applied to the description of several types of process, to describe both the behaviour of systems and changes in their structures. For example, Distributed Graph Transformations exploit a hierarchical view of distributed systems, where high-level "network" graphs define the overall architecture of a distributed system, while low-level "specification" ones refer to the specific implementation of local systems [60]. Rules act at two levels, i.e., a modification of the network graph must be accompanied by a consequent transformation in the associated specification graphs. Moreover, low-level graphs must agree on transformations of interface nodes i.e., nodes which represent common objects or relations. This approach has been applied to manage dynamic change in distributed databases in [61].

The notion of graph refinement through graph transformations is also associated with the idea of having graphs modelling different levels of abstraction or specification [26]. Recently, this idea has been extended to the refinement of software architectures, seen abstractly as instances of architectural styles, each described by a graph transformation system [40]. Transformations from platform independent to platform dependent specifications can thus be specified as transformations on the rules defining processes at a high level [4] and the preservation of their properties can be assessed [29].

6. Conclusion and future work

In this paper, we have presented a novel approach for modeling the complex processes involved in the activity of a team cooperating over a MANET. The main problem solved by our approach concerns the need for restructuring the workflow, by inserting activities related to maintain the network connectivity, and not directly relevant for the assigned tasks. This restructuring activities are triggered by a specific prediction layer, which is part of a complex architecture that we are currently investigated (and which has been outlined in the paper).

Our solution is based on the use of AHO-NETS, allowing the uniform modeling of workflows, topologies and rules as tokens, i.e., resources to be produced and consumed by the transitions of the net. Moreover, both workflows and topologies are modeled as graphs, so that proper categorical constructions can be exploited. Thus, it is possible to reason about properties such as deadlock avoidance and termination of workflows, in their original configuration, as well as after transformations.

On the basis of the formal approach presented in this paper, we are going to implement an adaptive workflow management system for MANET, specifically targeted to emergency teams equipped with PDAs and laptops (i.e., teams with not too powerful devices). Such a system, referred to as *mobidis*, is partly realized,[2] and will be completed and then validated in the context of some research projects we are currently involved.[3]

In the future, from the theoretical point of view we will develop sound techniques for reasoning about specific interesting properties targeted to the peculiarities of our scenario, as well as we will develop specific methodologies on how to design processes and their transformation rules.

References

[1] A. Agostini and G. De Michelis, *Improving flexibility of workflow management systems*, in *Proceeding of the BPM'00*, LNCS, vol. 1806, 2000, 218–234.

[2] A. Agostini and G. De Michelis, A light workflow management system using simple process models, in: *Int. J. Collab. Comp.*, (Vol. 9(34)), (Special issue on adaptive workflow systems), M. Klein, C. Dellarocas and A. Bernstein, eds, 2000, pp. 346-356.

[2]cfr. http://www.dis.uniroma1.it/pub/mecella/projects/MobiDIS/ where the Network Service Interface layer of the proposed architecture are made available, and where the Predictive layer will be very soon released.

[3]*MAIS* – http://www.mais-project.it, and the recently funded IST FP6 *WORKPAD* – http://europa.eu.int/information_society /activities/atwork/collaboration_at_work/events/2006_02_07_cwe_launch/cwe/index_en.htm.

[3] D.P. Agrawal and Q.A. Zeng, *Introduction to Wireless and Mobile Systems*, Thomson Brooks/Cole, 2003.

[4] L. Baresi, R. Heckel, S. Thöne and D. Varro, *Style-based refinement of dynamic software architectures,* in *WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture*, 2004, 155–164.

[5] P. Bottoni, U. Borghoff, P. Mussio and R. Pareschi, *Reflective agents for adaptive workflows,* in *Proceedings of 2nd Int. Conf. on* (*PAAM-97*), London, UK, 1997, 405–420.

[6] R.H. Campbell and A.N. Habermann, The specification of process synchronization by path expressions, in: *Operating Systems*, volume 16 of *LNCS*, E. Gelenbe and C. Kaiser, eds, 1974, pp. 89–102.

[7] F. Casati, S. Ceri, B. Pernici and G. Pozzi, Workflow evolution, *Data and Knowledge Engineering* **24**(3) (1998), 211–238, Elsevier.

[8] F. Casati and M.C. Shan, Dynamic and Adaptive Composition of *e*-Services, *Information Systems* **6**(3) (2001).

[9] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel and M. Löwe, Algebraic approaches to graph transformation; basic concepts and double pushout approach, in: *Handbook of Graph Grammars and Computing by Graph Transformation, volume 1: Foundations*, G. Rozenberg, ed., World Scientific, 1997.

[10] G. De Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, K. Pohl, J. Schmidt, C. Woo and E. Yu, Cooperative Information Systems: A Manifesto, Technical report, University of Hamburg, April 1996.

[11] F. De Rosa, A. Malizia and M. Mecella, Disconnection Prediction in Mobile Ad hoc Networks for Supporting Cooperative Work, *IEEE Pervasive Computing* **4**(3), 2005.

[12] F. De Rosa, V. Di Martino, L. Paglione and M. Mecella, *Mobile Adaptive Information Systems on MANET: What We Need as Basic Layer?,* in Proc. WISE 2003 Workshop on Multichannel and Mobile Information Systems (MMIS'03), IEEE, 2003.

[13] F. De Rosa, M. Mecella, P. Faraglia and F. Pascucci, *Designing and Implementing a MANET Network Service Interface with Compact .NET on Pocket PC,* in Proc. International Conference on .NET Technologies 2005.

[14] M. Dumas and A.H.M. ter Hofstede, *UML Activity Diagrams as a Workflow Specification Language,* in Proc. UML 2001, LNCS 2185, 2001.

[15] H. Ehrig, G. Engels, H.-J. Kreowski and G. Rozenberg, eds, *Handbook of Graph Grammars and Computing by Graph Transformation, volume 2: Applications*, *Languages and Tools*, World Scientific, 1999.

[16] H. Ehrig, A. Habel, H.-J. Kreowski and F. Parisi-Presicce, Parallelism and Concurrency in High-Level Replacement Systems, *Math. Struct. in Comp. Science* **1** (1991), 361–404.

[17] H. Ehrig, M. Gajewsky and F. Parisi-Presicce, High-Level Replacement Systems Applied to Algebraic Specifications and Petri Nets, in: *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 3: Concurrency, Parallelism and Distribution*, World Scientific, Singapore, 2000, pp. 341–400.

[18] H. Ehrig, K. Hoffmann, J. Padberg, P. Baldan and R. Heckel, *High-Level Net Processes,* Formal and Natural Computing, LNCS 2300, 2002, 191–219.

[19] H. Ehrig and B. Mahr, *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, Vol. 6 of EATCS Monographs on Theoretical Computer Science, 1985.

[20] E. Ehrig and J. Padberg, *Graph Grammars and Petri Net Transformations*, Lectures on Concurrency and Petri Nets. Special Issue Advanced Course PNT LNCS 3098, Springer, 2004, 496–536.

[21] H. Ehrig, M. Pfender and H.J. Schneider, *Graph Grammars: an Algebraic Approach,* in Proc. 14th Annual IEEE Symposium on Switching and Automata Theory, 1973.

[22] C.A. Ellis, K. Keddara and G. Rozenberg, *Dynamic Change Within Workflow Systems,* in Proc. COOCS '95, ACM Press, 1995, 10–21.

[23] C.A. Ellis and K. Keddara, *A workflow change is a workflow,* in Proc. BPM '00, LNCS, vol. 1806, 2000, 516–534.

[24] R. Eshuis and R. Wieringa, Comparing petri net and activity diagram variants for workflow modelling – a quest for reactive petri nets, in: *Petri Net Technology for Communication Based Systems, volume 2472 of Lecture Notes in Computer Science*, H. Ehrig, W. Reisig, G. Rozenberg and H. Weber, eds, Springer, 2003, pp. 321–351.

[25] M. Gajewsky, K. Hoffmann and J. Padberg, Place Preserving and Transition Gluing Morphisms in Rule-Based Refinement of Place/Transition Systems, Technical Report 99-14, Technical University Berlin, 1999.

[26] M. Große-Rhode, F. Parisi Presicce and M. Simeoni, *Spatial and temporal refinement of typed graph transformation systems,* in Proc. Math. Foundations of Comp. Science 1998, volume 1450 of Lecture Notes in Computer Science, 1998, 553–561.

[27] J. Grudin, Computer-Supported Cooperative Work: History and Focus, *IEEE Computer* **27**(5) (1994), 19–26.

[28] Y. Han, *Software Infrastructure for Configurable Workflow System – A Model-Driven Approach Based on Higher-Order Nets and CORBA*, PhD Thesis, Technische Universität Berlin, 1997.

[29] R. Heckel and S. Thöne, *Behavior-preserving refinement relations between dynamic software architectures,* in Proc. of the 17th Int. Workshop on Algebraic Development Techniques (WADT 2004), volume 3423 of LNCS, 2004, 1–27.

[30] C. Heinlein, *Workflow and process synchronization with interaction expressions and graphs,* in ICDE, Proceedings of the 17th International Conference on Data Engineering, April 2–6, 2001, Heidelberg, Germany, 2001, 243–252.

[31] K. Hoffmann, *Formal Approach and Applictions of Algebraic Higher Order Nets*, PhD Thesis, Technische Universität Berlin, 2005.

[32] K. Hoffmann, H. Ehrig and T. Mossakowski, *High-Level Nets with Nets and Rules as Tokens,* in Proc. 26th International Conference On Application and Theory of Petri Nets and Other Models of Concurrency, 2005.

[33] K. Hoffmann and T. Mossakowski, Algebraic Higher-Order Nets: Graphs and Petri Nets as Tokens, in: *Proc. of 16th International Workshop of Algebraic Development Techniques*, M. Wirsing, D. Pattinson and R. Henicker, eds, LNCS 2755, Springer, 2003, pp. 253–267.

[34] K. Hoffmann and T. Mossakowski, Integration of Petri Nets and Rule-Based Transformations, Technical Report 04-57, Technical University Berlin, 2004.

[35] K. Hoffmann, F. Parisi-Presicce and T. Mossakowski, *Higher-Order Nets for Mobile Policies,* in Proc. ICGT'04 Workshop on Petri Nets and Graph Transformation (PNGT), 2005.

[36] B. Kiepuszewski, A.H.M. ter Hofstede and C.J. Bussler, *On structured workflow modelling,* in *CAiSE'00*, LNCS, vol. 1789, 2000, 431–445.

[37] M. Kradolfer and A. Geppert, *Dynamic workflow schema evolution based on workflow type versioning and workflow migration,* in CoopIS '99, Edinburgh, 1999, 104–114.

[38] J. Jensen, Coloured Petri Nets. Basic Concepts,Analysis Methods and Practical Use, *EATCS Monographs in Theoretical Computer Science* **1** (1992), Springer-Verlag.

[39] F. Leymann and D. Roller, *Production Workflows. Concepts and Techniques*, Prentice Hall, 2000.

[40] D. Le Métayer, *Software architecture styles as graph grammars,* in Proc. 4th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Volume 216 of ACM Software Engineering Notes, ACM Press, 1996, 15–23.

[41] M. Mecella, T. Catarci, M. Angelaccio, B. Buttarazzi, A. Krek, S. Dustdar and G. Vetere, *WORKPAD: an Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios,* to appear in Proc. CTS 2006 – Special Session on Mobile Collaborative Work, 2006.

[42] J. Meseguer and U. Montanari, Petri Nets are Monoids, *Information and Computation* **80**(2) (1990), 105–155.

[43] R. Mohan, M.A. Cohen and J. Schiefer, *A state machine based approach for a process driven development of web-applications,* in CAiSE, volume 2348 of Lecture Notes in Computer Science, Springer, 2002, 52–66.

[44] R. Müller, U. Greiner and E. Rahm, *AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation, Data and Knowledge Engineering,* Elsevier, 2004.

[45] P. Muth, D. Wodtke, J. Weißenfels, A.K. Dittrich and G. Weikum, From centralized workflow specification to distributed workflow execution, *J. Intell. Inf. Syst.* **10**(2) (1998), 159–184.

[46] D. Niculescu and B. Nath, *Error Characteristics of Ad hoc Positioning Systems* (*APS*), in Proc. ACM MobiHoc Conference, 2004.

[47] J. Padberg, H. Ehrig and L. Ribeiro, Algebraic High-Level Net Transformation Systems, *Journal of Mathematical Structures in Computer Science* **5** (1995), 217–256.

[48] A. Poigne, On Specifications, Theories, and Models with Higher Types, *Inf. Control* **68**(1–3), (1986), 1–46.

[49] N.B. Priyantha, A. Chakraborty and H. Balakrishnan, *The Cricket Location-support System,* in Proc. ACM MOBICOM Conference, 2000.

[50] N.B. Priyantha, A. Miu, H. Balakrishnan and S. Teller, *The Cricket Compass for Context-aware Mobile Applications,* in Proc. ACM MOBICOM Conference, 2001.

[51] M. Reichert and P. Dadam, ADEPTflex-supporting dynamic changes of workflows without losing control, *JIIS* **10**(2) (1998), 93–129.

[52] W. Reisig, "Petri Nets", volume 4 of *EATCS Monographs on Theoretical Computer Science*, 1985.

[53] S. Rinderle, M. Reichert and P. Dadam, Correctness criteria for dynamic changes in workflow systems a survey, *Data and Knowledge Engineering* **50**(2004), 9–14, Elsevier.

[54] S. Rinderle, M. Reichert and P. Dadam, Flexible Support of Team Processes by Adaptive Workflow Systems, *Distributed and Parallel Databases* **16**(1) (2004), 91–116.

[55] G. Rozenberg, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundation*. World Scientific Publishing, Co., Inc., 1997.

[56] S. Sadiq, O. Marjanovic and M. Orlowska, Managing change and time in dynamic workflow processes, *IJCIS* **9**(12) (2000), 93–116.

[57] L. Schröder and T. Mossakowski, HASCASL: Towards integrated specification and development of Haskell programs, in: *Algebraic Methodology and Software Technology, volume 2422 of Lecture Notes in Computer Science*, H. Kirchner and C. Reingeissen, eds, Springer-Verlag, 2002, pp. 99–116.

[58] L. Schröder, T. Mossakowski and C. Maeder, *HASCASL – Integrated functional specification and programming. Language summary,* available at http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/HasCASL, 2004.

[59] J. Shih, *Wireless LAN Location System,* School of Information Technology and Electrical Engineering, The University of Queensland, Thesis for the Master of Telecommunications Engineering, November 2003.

[60] G. Taentzer, I. Fischer, M. Koch and V. Volle, Visual Design of Distributed Systems by Graph Transformation, in: *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 3: Concurrency, Parallelism, and Distribution*, H. Ehrig, H.J. Kreowski, U. Montanari and G. Rozenberg, eds, World Scientific, 1999, pp. 269–340.
[61] G. Taentzer, M. Goedicke and T. Meyer, *Dynamic change management by distributed graph transformation: Towards configurable distributed systems,* in Proc. TAGT'98 – Theory and Application of Graph Transformations, volume 1764 of LNCS, 1998, 179–193.
[62] W.M.P. van der Aalst, The Application of Petri Nets to Workflow Management, *The Journal of Circuits, Systems and Computers* **8**(1) (1998), 21–66.
[63] W.M.P. van der Aalst and T. Basten, Inheritance of Workflows: an approach to tackling problems related to change, *Theoretical Computer Science* **270**(1–2) (2002), 125–203, Elsevier Science Publishers Ltd.
[64] W.M.P. van der Aalst and K. van Hee, *Workflow Management: Models, Methods, and Systems*, MIT Press, 2001.
[65] W.M.P. van der Aalst, M. Weske and G. Wirtz, Advanced topics in workflow management: Issues, requirements, and solutions, *Int. J. Integrat. Design Process* **Sci. 7**(3) (2003).
[66] D. Wodtke and G. Weikum, *A Formal Foundation for Distributed Workflow Execution Based on State Charts*, ICDT 1997, 1998, 230–246.
[67] M. Weske, *Formal foundation and conceptual design of dynamic adaptations in a workflow management system,* in Prod. of HICSS-34, 2001.
[68] J. Widom and S. Ceri, *Active Database Systems: Triggers and Rules for Advanced Database Processing*, Morgan Kaufmann, 1996.
[69] U. Wolter, *Higher-Order Partial Algebras,* University of Bergen (Norway), Technical Report 299, 2005.

Paolo Bottoni graduated in Physics in 1988 and obtained his Doctoral Degree in Computer Science in 1995. Since 1994, he has been with the Department of Computer Science of the University of Rome "La Sapienza", first as a researcher, and since 2000 as an associate professor. His research interests are mainly in the area of interactive computing, and include: definition of pictorial and visual languages, visual simulation, formal models of visual interactive computing, agent-based computing, multimedia applications for e-learning and entertainment. On these topics, he has published more than 100 scientific papers in international journals, contributed volumes and conference proceedings. Contact him at bottoni@di.uniroma1.it.

Fabio De Rosa is a Ph.D. student in Computer Science at the Department of Computer Science, and a research assistant at the Department of Systems and Computer Science, University of Rome "La Sapienza". His reasearch interests include multichannel and mobile adaptive information systems, cooperative information systems, workflow management and Web Services. He received an MSc in Computer Science from the University of Rome "La Sapienza". Contact him at derosa@dis.uniroma1.it.

Kathrin Hoffmann is a research assistant in the group Theoretical Computer Science/Formal Specification, Technical University Berlin, and a post-doctoral researcher at the Department of Computer Science, University of Rome "La Sapienza", supported by the European Research Training Network "Syntactic and Semantic Integration of Visual Modelling Techniques". Her research interests include integration of visual modelling techniques, formal development and verification techniques, Petri net technology, graph transformation, algebraic specification techniques. She received a Ph.D. in Computer Engineering from the Technical University Berlin. Contact her at hoffmann@cs.tu-berlin.de.

Massimo Mecella is a research associate and a lecturer in the Department of Systems and Computer Science, University of Rome "La Sapienza". His research interests include service oriented computing and inter-organization processes, cooperative systems for e-Government, mobile and adaptive information systems, middleware technologies. He received a Ph.D. in Computer Engineering from the University of Rome "La Sapienza". Contact him at mecella@dis.uniroma1.it.