A Framework for Formal Modeling and Analysis of Organizations

Catholijn M. Jonker[1], Alexei Sharpanskykh[2],
Jan Treur[2], and pInar Yolum[3]

[1] NICI, Radboud University Nijmegen
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl
[2] Department of Artificial Intelligence, Vrije Universiteit Amsterdam,
De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands
{sharp,treur}@few.vu.nl
[3] Department of Computer Engineering, Bogazici University,
TR-34342 Bebek, Istanbul, Turkey
pyolum@cmpe.boun.edu.tr

Abstract. A new, formal, role-based, framework for modeling and analyzing both real world and artificial organizations is introduced. It exploits static and dynamic properties of the organizational model and includes the (frequently ignored) environment. The transition is described from a generic framework of an organization to its deployed model and to the actual agent allocation. For verification and validation of the proposed model, a set of dedicated techniques is introduced. Moreover, where most computational models can handle only two or three layered organizational structures, our framework can handle any arbitrary number of organizational layers. Henceforth, real-world organizations can be modeled and analyzed, as illustrated by a case study, within the DEAL project line.

Keywords: computational, modeling, analysis, organization model, logic-based, simulation, verification

1 Introduction

Recently computational modeling and analysis of organizations received a special attention in the areas of social science and artificial intelligence. In particular, organizations have proven to be a useful paradigm for analyzing and designing multi-agent systems [7, 10, 42]. Representation of a multi-agent system as an organization consisting of roles and groups can tackle major drawbacks concerned with traditional multi-agent models; e.g., high complexity and poor predictability of dynamics in a system [10]. As has been shown in [19], organizational structure can be used to limit the scope of interactions between agents, reduce or explicitly increase redundancy of a

system, formalize high-level system goals, of which a single agent may be not aware, or enforce certain coordination mechanisms for efficient task execution.

Moreover, organizational research in social science has recognized the advantages of computational models; e.g., for analysis of structure and dynamics of real organizations. In particular, distributed simulation models were created for analyzing organizational adaptation processes [5, 34], social networks [38] and dynamic processes in different organization types. However, in general formal theories, approaches, and tools for designing computational models of organizations are still rare and most of them are dependant of specific social theoretical background (cf. the OrgCon tool for organizational design based on the contingency theory [4]). In this paper, we propose a new modeling approach for analyzing and formal modeling of real or artificial organizations (e.g., agent-based organizations), independent of any organizational theory from social science. This approach is based on a generic representation of organizations that comprises sets of interrelated roles, which are intentionally organized to ensure a desired (or required) pattern of activities. The approach has the following distinct features: (1) it addresses both organization structure and dynamics; (2) the approach has its formal foundation in an expressive order-sorted predicate language with properly defined syntax and semantics; (3) it allows multiple aggregation levels in an organization model; (4) the environment is explicitly incorporated in an organization model; (5) the approach provides formal techniques and tools for different types of analysis of organization models (by performing simulations and verification).

In the next Section, main principles for modeling and analyzing organizations are discussed and related with the new modeling approach. In Section 3, the basic concepts used for specifying an organization model are introduced. Section 4 discusses how an organization model can be specified in a formal manner. In Section 5, a set of dedicated validation and verification techniques are described. The proposed modeling and verification techniques will be illustrated by a running example from the area of logistics. The paper ends with a discussion in Section 6.

2 Principles for modeling and analyzing organizations

Modern organizations are characterized by their complex structure, dense information flows, and incorporation of information technology. To a large extent, the underlying organization model is responsible for how efficiently and effectively organizations carry out their tasks. In literature on organization theory, a range of theories and guidelines concerning the modeling and design of organizations are present [28, 30]. However, no operational general theories or formal models exist that are known to the authors. Scott [39] even stated that no general principles applicable to organizational modeling can be formulated. However, for certain specific organizational types standard modeling and design techniques may still be identified and formalized. In particular, Minzberg proposed a set of guidelines for modeling mechanistic types of organizations [28]. This type of organizations comprises systems of hierarchically linked job positions with clear responsibilities that use standard well-understood technology and

operate in a relatively stable (possibly complex) environment. In contrast to mechanistic (or functional) organizations, a substantial group of modern organizations are characterized by a highly dynamic, constantly changing, organic structure with non-linear behavior [29]. Although the structure and behavioral rules for such organizations can be hardly identified and formalized, nevertheless by performing agent-based simulations with changing attitudes of proactive agents useful insights into functioning of such organizations can be gained.

2.1 Two perspectives

In this subsection, we will briefly discuss two perspectives from which organizations are analyzed. The first perspective emerges from social sciences and the second originates from computational organization theory and artificial intelligence.

In social science theories, the structure of organizations is frequently specified as informal or semi-formal graphical representations [28, 30]. They can provide a detailed organization structure at an abstract level considered from a certain perspective (e.g., information flows, power and authority relations, allocation of resources). The disadvantages of such models are: (1) lack of generality and relations between different specific types of models, and (2) graphically depicted data can not be effectively processed, combined and analyzed. Furthermore, such approaches lack the means to represent the more detailed dynamics and to relate them to the structures present.

A class of models built based on the system dynamics theory allows formal representation of different aspects of organizational behaviour [12]. Organizational models specified in system dynamics are based on numerical variables and equations that describe how these variables change over time. Although such models can be computationally effective (i.e., used for simulations and computational analysis), nevertheless they still lack the ontological expressivity and the possibility for higher abstract (and, e.g., non-quantitative) representations that are needed to conceptualize wide range of relations and phenomena that exist in different types of organizations.

From computational organization theory and artificial intelligence, approaches have been developed that are able to capture both structural and dynamic aspects of organizations. Some of them are dedicated for analyzing particular aspects of an organization considered from a certain viewpoint (e.g., Petri-nets techniques used for modelling and analyzing business processes [8]). Although such approaches can be useful and efficient, the scope of their application is limited to a particular view on an organization, based on a limited number of concepts. Furthermore, techniques from the area of artificial intelligence have been applied for modelling and analyzing multi-agent organizations [3, 7]. In such organizational models (software, hardware or human) agents are allocated to roles that stand in certain relations to each other and often are described by sets of functionalities performed by an organization. Such models can be used for example for coordinating tasks execution in a multi-agent system [19], or for enforcing certain behaviours (e.g., normative systems) upon an agent system [41]. However, many of such models can handle only two or three levels of abstraction; i.e., the level of an individual role, to which an agent(s) will be eventually allocated, the level of a group composed of roles, and the overall organization level, as in

GAIA [42], MOISE [16, 20], MOCA [1], TOVE [13], Aaladin [11] and OperA [7]. In contrast, multiple levels and relations between them need to be described for the representation of complex hierarchical structures of modern organizations; e.g., mechanistic type of organizations [30]. One of the few exceptions known to authors as capable of representing hierarchical structures is a framework for modeling social structures in UML proposed in [33]. This framework allows the possibility of the iterative inclusion of groups represented by holonic agent structures into other groups as their members, thus building hierarchical structures. However, the framework does not provide a general mechanism for handling interactions between roles and groups of different aggregation levels that often occur in such hierarchical structures. Furthermore, there is no possibility to identify and formally specify how dynamics of a composite group is related to the dynamics of its members, which is a prerequisite for the (formal) analysis of behavior of such composite systems. Another framework that supports the hierarchical representation of a multi-agent system is based on teams of agents [17]. A team is a composite component, similar to a group in [33], which is characterized by a number of roles, enacted by agents and other teams. However, this framework lacks means for elaborated conceptual modeling of social structures, probably because its main focus is on the technical side of programming and implementation of multi-agent systems. By introducing for example a (formal) language for specifying dynamics of individual roles and teams in this framework, different interesting types of analysis of system dynamics could be enabled.

Some models (ISLANDER [9], OperA, [26]) consider organizations as electronic institutions; i.e., norms and global rules that govern an organization are explicitly defined. However, in many modern organic organizations with much individual autonomy, the normative aspects do not play a central role and are of minor importance for the prosperity of an organization. Furthermore, a temporary violation of certain norms is inevitable and even necessary in certain organizations.

Independent of the previous distinction in approaches, the importance of explicit modeling of interactions between agents and the environment is recognized (explicitly considered in SODA [32] and AUML [31]). Since most of the modern organizations are open systems that actively interact with the environment, both an organizational structure and behavior are contingent on the environmental conditions.

Moreover, for modeling in general, verification and validation of the models used or generated is of the utmost importance. This is no different for modeling organizations. However, this aspect of modeling organizations is frequently ignored; two of the exceptions are TROPOS [3] and ISLANDER.

2.2 A new perspective

In this paper, we propose an approach for formal modeling and analysis of organizations. It is highly suitable for mechanistic types of organizations with the explicitly defined structure and behavior (i.e., machine and professional bureaucracy), and divisionalized forms of organizations that consist of autonomous units with specialized and formalized inputs and outputs. Furthermore, this approach can also be applied for

modeling organic types of organizations, when extended with organizational change techniques.

The proposed, formal approach can capture both structural and dynamic aspects of the organization and, subsequently, has four advantages:

(1) Representation of organization structure (including specifications of actors (or roles), relations between them, and information flows) and dynamics by generalized (template) models and more specific instantiated (deployed) models.
(2) The means for simulations of different (agent-based) scenarios on the basis of a model and observing their results.
(3) Organization analysis by means of verifying static and dynamic properties (e.g., based on organizational performance indicators) against (formalized) empirical data, taken from real organizations, or against simulated scenarios.
(4) Diagnosis of inconsistencies, redundancies, conflicts, and errors in an organizational model by means of formal verification techniques (e.g., based on model checking [6]).

In the proposed model, organizations are specified as composite roles that can be refined iteratively into a number of (interacting) composite or simple roles, representing as many aggregation levels as needed. The refined role structures correspond to different types of organization constructs (e.g., groups, units, departments). By considering only role hierarchies we achieve the uniform representation of an organization structural model, which is still able to reflect all the major types of organization constructs. The proposed framework provides formal means for specifying structural relations between roles of the same and different aggregation levels.

Behaviour of roles at each aggregation level is defined by sets of dynamic properties specified using an expressive temporal logical language. In the proposed approach different types of dynamic properties are distinguished, which are capable to capture different aspects of organizational dynamics. Note that the behaviour of a composite role is not simply defined as a list of all dynamic properties of its subroles. The dynamics of a composite role may be characterized by properties that emerge from the dynamics of its subroles or represent a more abstracted view on the lower-level dynamics. Therefore, particularly in the design phase when role dynamic properties are identified and specified, inconsistencies and conflicts between the properties of roles of adjacent aggregation levels may occur. Mechanisms to deal with these conflicts can be found in Section 5 and are further developed in the context of the proposed approach.

It is important to stress that the organizational model can be specified, depicted and analyzed at each aggregation level separately. For example, since the whole organization is considered as one composite role, it can be used as a “black box” with formally specified input and output interfaces for modelling and analyzing of high-level inter-organizational processes. In such a way the scalability of an organization model and the proposed approach is achieved.

Moreover, global normative aspects of an organization that are usually specified by organizational policies are defined by static and dynamic properties of the role at the highest aggregation level, without recognizing them as special concepts and placing them on top of an organization.

In addition, the environment is considered as a special component of the organization model. The environment is populated by agents that under certain conditions may be allocated to organizational roles. Furthermore, the environment serves as a source of events for an organization.

The modeling method introduced in this paper incorporates two types of verification and validation techniques: role-centered and agent-centered, as will be discussed in Section 5. The introduction of these techniques is preceded by the introduction of the model itself in the next section and its formal specification in Section 4.

3 Organization Modeling Concepts

In this section, the concepts are introduced on which the organization modeling approach is founded. First, the specification of the organizational structure is described. A template model is generated, which encapsulates the structure of the organization. On all existing levels of aggregation, the behavior of an organization can be described. Taken together, this provides description of the behavior of an organization. In Section 3.2, it will be explained how such dynamic behavior can be specified. In Section 3.3, the transition from template model to deployed model will be discussed. The introduced modeling concepts will be gradually used to represent different aspects of the organizational structure and behavior of an organization from the area of logistics.

3.1 Organization structure

An organization structure reflects patterns of interactions in an organization and is described by relationships between roles at the same and at adjoining aggregation levels and between parts of the conceptualized environment and roles. The specification of an organization structure that constitutes a template model uses the following elements:

(1) A role represents a subset of functionalities, performed by an organization, abstracted from specific agents (or actors) who fulfill them.

Each role can be composed by several other roles, until the necessary detailed level of aggregation is achieved, where a role that is composed of (interacting) subroles, is called a composite role. At the highest aggregation level, the whole organization can be represented as one role. Such representation is useful both for specifying general organizational properties and further utilizing an organization as a component for more complex organizations. Each role has an input and an output interface, which facilitate in the interaction (communication) with other roles. Graphically, a role is represented as an ellipse with white dots (the input interfaces) and black dots (the output interfaces).

(2) An interaction link represents an information channel between two roles at the same aggregation level. Graphically, it is depicted as a solid arrow, which denotes the direction of possible information transfer.

(3) The conceptualized environment represents a special component of an organization model. The environment can be defined by a set of objects with certain properties

and states and by causal relations between objects. On the one hand, agents allocated to organization roles are capable of observing states and properties of objects in the environment; on the other hand, they can act or react and, thus, affect the environment. We distinguish passive and active observation processes. For example, when some object is observable by an agent playing a role and the agent continuously keeps track of its state, changing its internal representation of the object if necessary, passive observation occurs. For passive observation, no initiative of a role or an agent is needed. Active observation is always concerned with the agent's (or role's) initiative. Similarly to roles, the environment has input and output interfaces, which facilitate in the interaction with roles of an organization. Graphically, the environment is depicted as a rectangle with rounded corners. For particular purposes the internal specification for the environment can be conceptualized using one of the existing world ontologies (e.g., CYC, SUMO, TOVE). However, despite the richness and the extensiveness of these ontological bases, more specific and refined types of concepts and relations are required for modeling particular types of organizations and environments.

(4) An environment interaction link represents an information channel between a role of a certain aggregation level and (a part of) the conceptualized environment represented at this aggregation level. Graphically, it is depicted as a dotted arrow, which denotes the direction of possible information transfer.

(5) An interlevel link connects a composite role with one of its subroles. It represents information transition between two adjacent aggregation levels. Graphically, it is depicted as a dashed arrow, which shows the direction of the interlevel transition.

To illustrate the introduced concepts to model the organizational structure and all the following components of an organization model consider a running example based on a case study from the area of logistics. This case study was done within the project DEAL (Distributed Engine for Advanced Logistics). For the project description, we refer to *http://www.almende.com/deal/*. A template organizational model was created, based on the informal description of the structure and functioning of the large Dutch logistics company. Only relevant to the actual delivery process actors (roles) and their properties are specified in this model. To secure anonymity of the company, the real names of the organizational units were substituted by general ones.

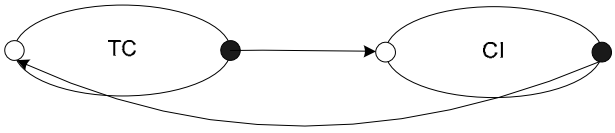

Fig. 1. Representation of the organization at abstraction level 1, which consists of role Transport Company (TC) and role Customer Interaction (CI)

At the highest aggregation level (level 0) the whole organization is represented as one role. At aggregation level 1, the organization consists of two interacting roles: TC and CI (see Fig.1; explanation for this and the following abbreviations and functional descriptions are given in Table 1). Note, that the organizational model is depicted in a modular way; i.e., components of every aggregation level can be visualized and ana-

lyzed both separately and in relation to each other. Consequently, scalability of graphical representation of an organizational model is achieved.

Table 1. Role names, abbreviations, and descriptions for the organizational model in the case study

Role name	Abbreviation	Description
Transport Company	TC	Provides logistic services to customers
Customer Interaction	CI	Identifies interaction rules between a customer and the transport company
Strategy and Tactical Department	ST	Performs analysis and planning of company activities; considers complaints from customers; analyses the satisfaction level of a customer by means of surveys and questionnaires
Custom Relations Department	CR	Handles requests from customers
Operational Department	OP	Responsible for direct fulfillment of the order from a customer
Transport Company Representative	TCR	Mediator role between a customer and the transport company
Customer	C	Generates an order for the transport company; sends inquiries about the delivery status
Sales Person	SP	Assigns an order to a certain load manager, based on the type and the region of a delivery
Load Manager	LM	Assigns orders to suitable trucks and available drivers; assigns fleet managers to drivers; provides CR department with up-to-date information about delivery; provides a driver with instructions in case of a severe problem; informs CR department about possible delays with delivery
Fleet Manager	FM	Keeps constant contact with the assigned drivers; updates automatic support system with actual data on the delivery status; provides consultations for drivers in case of minor problems in transit
Driver	D	Delivers goods; informs a superior fleet manager about the delivery status; interacts (by means of observations and actions) with the conceptualized part of the environment
Environment	Env	Represents the conceptualized environment; in this example only a driver interacts with it

At aggregation level 2 role TC can be refined into three interacting roles: ST, CR, and OP (see Fig.2). All interactions with a customer are conducted within CI role. At aggregation level 2 it consists of two roles: TCR and C (see Fig. 2). Role TCR produces at its output messages from CR and ST departments of the transport company, i.e., CR and ST roles stand as company representatives in certain interactions with a customer. Therefore, the input state of role TCR has influence on the output state of role CR and vice versa. The same holds for role ST.

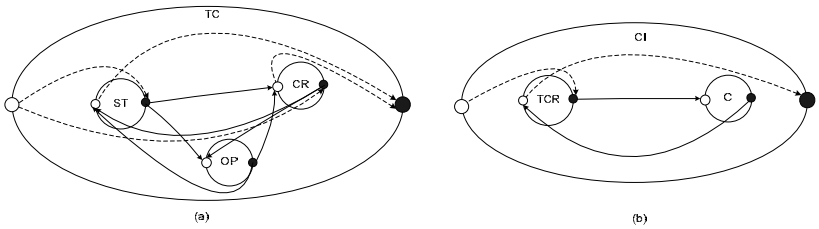

Fig. 2. Representation of (a) the Transport Company (TC) and (b) the Customer Interaction role (CI) at abstraction level 2

The structure of the operational department that is responsible for the direct fulfillment of the order from a customer is depicted at aggregation level 3 in Figure 3. It consists of interacting roles LM, FM, SP and D. Roles LM and SP are able to receive (or transmit) information from (or to) roles outside of role OP by means of interlevel links. Furthermore, in this model only role D interacts with the conceptualized environment.

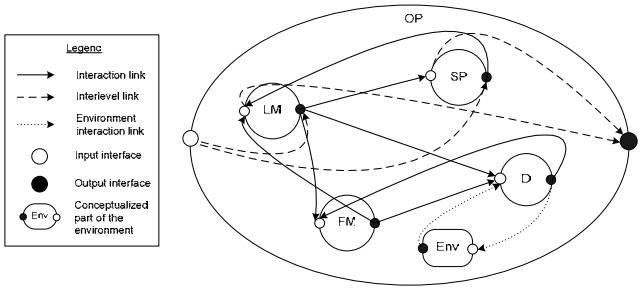

Fig. 3. Representation of the operational department at abstraction level 3

3.2 Organizational dynamics

At each aggregation level, it can be specified how the organization's behavior is assumed to be. To this end, organization dynamics are described by a dynamic representation, for each of the elements in an organization structure. The level of detail for specifying dynamics of an organization depends on its organizational type. Since the behavior of most mechanistic organizations is deterministic, dynamics for such organizations can only be modeled by a set of dynamic properties with high level of detail. In contrast, behavior of many organic organizations is defined loosely. Consequently, the dynamics of models for such organizations can be specified only partially; hence, actors (agents) can act autonomously.

The dynamics of each structural element are defined by the specification of a set of dynamic properties. We define five types of dynamic properties:

(1) ***A role property*** **(RP)** describes the relationship between input and output states of a role, over time. For example, in the settings of the logistics company from the

running example, a role property of a truck driver (role D) can be defined as: if role Driver receives a request from his Fleet Manager to provide his coordinates, then role Driver will generate this data for his Fleet Manager.

(2) ***A transfer property*** **(TP)** describes the relationship of the output state of the source role of an interaction link to the input state of the destination role. Again, in the settings of the logistic company an example of a transfer property is the following: if role Customer generates an order to role Transport Company, then Transport Company will receive this order.

(3) ***An interlevel link property*** **(ILP)** describes the relationship between the input or output state of a composite role and the input or output state of its subrole. Note that an interlevel link is considered to be instantaneous: it does not represent a temporal process, but may give a different view on the same information state. Consider an example of such property: if role TCR obtains the customer order data at its input, then at the same time point role CI generates at its output a number assigned to the customer order in the automated information system.

(4) ***An environment property*** **(EP)** describes a temporal relationship between states or properties of objects of interest in the environment. Consider an environment property from the running example: If a severe incident happens with the truck involved in the delivery process, then it will cease the delivery.

(5) ***An environment interaction property*** **(EIP)** describes a relation either between the output state of the environment and the input state of a role (or an agent) or between the output state of a role (or an agent) and the input state of the environment. For example: if the information about a traffic jam on the way of role D is generated at the output of the environment, then role D will receive (observe) this information at its input.

3.3 Deployed model and agent allocation

The generic or template model of an organization provides abstracted information concerning its structure and functioning. However, for a more detailed analysis, a deployed model is needed. It is based on both unfolded generic relations between roles, as defined in the template model, and on creating new role instances. In such a way, role instances from the deployed model can be related to generic roles from a template model by means of the generalization relations. Moreover, different deployed models may be specified using the same template model of an organization for different purposes.

In the deployed model for the considered running example, all roles specified at abstraction levels 1 and 2 have one-to-one mapping to the role instances. While roles LM, FM, and D (defined at abstraction level 3) have multiple instances; e.g., LM and FM are represented differently in different geographical regions and, subsequently, different types of trucks and professional skills of drivers are required for different kinds of deliveries. The deployed model for the considered example (see Fig. 4) is created based on the template model by unfolding assigned_to and in_region relations between roles. For example, assigned_to(D2, FM1) denotes that a middle-size truck and his driver (D2) are assigned to the fleet manager in eastern Europe (FM1) and the relation

in_region(D1, LM1) specifies that both a big-size truck driver (D1) and a load manager (LM1) should belong to the same region in eastern Europe.

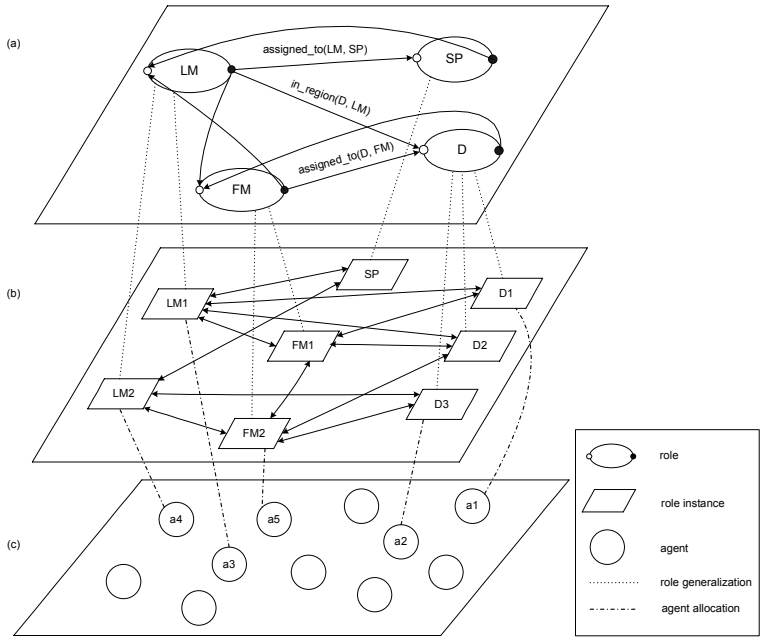

Fig. 4. The operational department of the transport company represented at abstraction level 3, with (a) the template model (b) the deployed model, and (c) agent allocation

The deployed model abstracts from the actual agent allocation but provides the detailed specifications for the behavior of role instances. Based on these specifications, a set of requirements is formulated for each role instance. These requirements (by restricting and defining behavior) are imposed onto the agents, who will eventually enact these roles. In the context of the running example one of the requirements imposed on a driver is that the agent should have a driver license of a certain type and acceptable results of medical tests.

Each agent is characterized by a set of capabilities that describe skills and credentials of an agent. An agent can be allocated to a role only when agent capabilities match the set of role requirements. For example, in order to enact role LM, an agent should have working experience as a senior manager in logistics for at least 3 years.

If, for some reason, an allocated agent is not capable of enacting a certain role anymore, dynamic reallocation of another agent will take place.

In some scenarios, a complex role can act as a single aggregated role and, thus, representing its constituting subroles. In such cases, an (aggregated) agent can be assigned to the complex role. In the literature [36, 37] aggregated (or composite) agents are often called *holons*. A holon is defined by a recursive model of agent groups and appears as a single entity to the outside world. A holon may impose certain structures (i.e., types of relations) and behaviors on its agents, thus limiting their au-

tonomy in certain aspects. Furthermore, a holon may be allocated to a simple (not composite) role, when the joint set of capabilities of agents of the holon satisfies the role requirements.

4 Formal Specification of the Organization Model

In the previous section, the elements of the organizational model were introduced. The current section provides the formal specification of them.

4.1 Structural properties

Structural properties describe elements of an organization structure introduced in Section 3.1 and relations between them.

As it has been shown above, in an organization model roles interact with other roles and the environment by means of input and output interfaces. These interfaces are described in terms of interaction (input and output) ontologies: a vocabulary or a signature specified in order-sorted logic that comprises finite sets of sorts, constants within these sorts, and relations and functions over these sorts. Generally speaking, an input ontology determines what types of information are allowed to be transferred to the input of a role (or of the environment), and an output ontology predefines what kinds of information can be generated at the output of a role (or of the environment). Roles and relations between them and the environment defined in a template model, as well as role instances and relations between them and the environment defined in a deployed model are specified using sorts and predicates from the structure ontology. This ontology includes sorts for all structural elements of an organization model (such as roles, different types of links, environment). The predicates for specifying organizational structure are defined over these sorts in Table 2. For example, in the settings of the logistics company from the running example, subroles Fleet Manager (FM) and Load Manager (LM) belong to the same composite role Operational department (OP). Formally: has_subrole(OP, FM) & has_subrole(OP, LM). Note that input and output ontologies of role instances are constructed by limiting and refining the ontologies of template roles based on which these role instances have been created.

In order to enable interaction between roles at the same aggregation level it is required that the ontologies of interacting roles contain common (or shared) elements (e.g., to specify the speech act s_act (e.g., inform, request, ask) from role-source r1 to role-destination r2 with the content message the predicate communicate_from_to(r1:ROLE, r2:ROLE, s_act:SPEECH_ACT, message:STRING) may be defined as a part of ontologies for both roles).

However, ontologies of roles connected by an interlevel link may not contain common elements. In this case the interlevel link is described by an ontology mapping between the corresponding elements of ontologies. Moreover, an ontology mapping associated with an interlevel link may be used for representing mechanisms of information abstraction. These mechanisms can be applied for transmitting (or generating)

partial, aggregated or generalized information to the input (or from the output) of a role.

Table 2. Ontology for formalizing organizational structure

Predicate	Description
is_role: ROLE	Specifies a role in an organization
has_subrole: ROLE x ROLE	For a subrole of a composite role
source_of_interaction: ROLE x INTERACTION_LINK	Specifies a source role of an interaction
destination_of_interaction: ROLE x INTERACTION_LINK	Specifies a destination role of interaction
interlevel_connection_from: ROLE x INTERLEVEL_LINK	Identifies a source role of an interlevel link
interlevel_connection_to: ROLE x INTERLEVEL_LINK	Identifies a destination role of an interlevel link
initiator_env_interaction: ROLE x ENVIRONMENT_INTERACTION_LINK	Specifies a role-initiator in interaction with the environment
recipient_env_information: ROLE x ENVIRONMENT_INTERACTION_LINK	Identifies a role-recipient of information from the environment
part_of_env_in_interaction: ENVIRONMENT x ENVIRONMENT_INTERACTION_LINK	Identifies the conceptualized part of the environment involved in interaction with a role
has_input_ontology: ROLE x ONTOLOGY	Specifies an input ontology for a role
has_output_ontology: ROLE x ONTOLOGY	Specifies an output ontology for a role
has_input_ontology: ENVIRONMENT x ONTOLOGY	Specifies an input ontology for the environment
has_output_ontology: ENVIRONMENT x ONTOLOGY	Specifies an output ontology for the environment
has_interaction_ontology: ROLE x ONTOLOGY	Specifies an interaction ontology for a role
has_interaction_ontology: ENVIRONMENT x ONTOLOGY	Specifies an interaction ontology for the environment
has_onto_mapping: INTERACTION_LINK x ONTO_MAPPING	Identifies an ontology mapping
to_be_observed: STATE_PROPERTY	Describes a state property that will be observed in the environment
observation_result: STATE_PROPERTY x BOOLEAN_VALUE	Determines if a certain state property holds in the environment
to_be_performed: ACTION	Specifies an action that will be performed in the environment

Often, structural properties are valid during the whole period of organization existence and can be considered as static. But in rapidly developing and adapting organizations, structural change processes gain special importance. Structural properties for such organizations get a temporal dimension and can be considered as a subclass of dynamic properties.

4.2 State and dynamic properties

The dynamics of an organization are defined by the specification of dynamic properties of its components that are formalized using the dynamic ontology (see Table 3) and belong to the following five classes: role properties, transfer properties, interlevel

link properties, environment properties, and environment interaction properties. Each dynamic property represents a relation in time either between (input or output) states of roles or a (input or output) state of a role and a (input or output) state of the environment. States of roles and the environment are defined based on the corresponding ontologies for roles and the environment. More precisely, a state for ontology Ont is an assignment of truth-values to the set At(Ont) of ground atoms expressed in terms of Ont. The set of all possible states for state ontology Ont is denoted by STATES(Ont).

Table 3. Dynamics ontology for formalizing properties of an organization

Sort	**Description**
DYNPROP	Sort for the name of a dynamic property
DPEXPR	Sort for the expression of a dynamic property
Predicate	**Description**
has_dynamic_property: ROLE x DYNPROP	Specifies a role dynamic property
has_dynamic_property: INTERACTION_LINK x DYNPROP	Identifies a dynamic property for an interaction link
has_dynamic_property: ENVIRONMENT x DYNPROP	Identifies a dynamic property for the conceptualized part of the environment
has_dynamic_property: ENVIRONMENT_INTERACTION_LINK x DYNPROP	Identifies a dynamic property for an environment interaction link
has_expression: DYNPROPx DPEXPR	Specifies an expression for a dynamic property

A state property is defined by a formula over a state ontology. For example, communicate_from_to(TCR, customer, inform, order_state(ON, delay, customer_report)) is a state formula expressing the informative speech act in form of a customer report from role TCR to role Customer about the delay state of the order with the number ON.

Dynamic properties (e.g., for roles, environment, and links) are specified in the Temporal Trace Language (TTL) [22, 40], which is a variant of order-sorted predicate logic [27], and in the classification in Galton [14, 15] falls in the class of reified temporal logic.

TTL has some similarities with situation calculus [35] and event calculus [24]. To enable reasoning about the dynamic properties the language TTL includes special sorts, such as: TIME (a set of linearly ordered time points), STATE (a set of all state names of a system), TRACE (a set of all trace names; a trace or a trajectory can be thought of as a timeline with for each time point a state), and STATPROP (a set of all state property names).

Role or environment states are related to state properties via the satisfaction relation |=, formally defined as a binary infix predicate (or by holds as a binary prefix predicate): state(γ, t, output(r)) |= p (or holds(state(γ, t, output(r)), p)), which denotes that state property p holds in trace γ at time t in the output state of role r.

Both state(γ, t, output(r)) and p are terms of the TTL language. Here p is used not as a statement, but as a term for an object in the language which refers to a state proposition; this is called reification; cf. Galton [14, 15]. TTL terms are constructed by induc-

tion in a standard way for sorted predicate logic from variables, constants and functional symbols typed with TTL sorts. Dynamic properties are expressed by TTL-formulae inductively defined by:

(1) If v_1 is a term of sort STATE, and u_1 is a term of the sort STATPROP, then holds(v_1, u_1) is an atomic TTL formula.
(2) If τ_1, τ_2 are terms of any TTL sort, then $\tau_1 = \tau_2$ is an atomic TTL formula.
(3) If t_1, t_2 are terms of sort TIME, then $t_1 < t_2$ is an atomic TTL formula.
(4) The set of well-formed TTL-formulae is defined inductively in a standard way based on atomic TTL-formulae using boolean propositional connectives and quantifiers.

In the context of the running example consider the information distribution property defined for role OP called RP1(OP), specified at abstraction level 2. Informally, when a severe problem with some delivery occurs, OP should generate a message to CR about possible delay. Formally specified in TTL:

∀γ:TRACE ∀t1:TIME ∀T:TRUCK_TYPE ∀D:DRIVER ∀ON:ORDER_NUM state(γ, t1, environment))|= [truck_state(T, incident, severe_incident) ∧ truck_property(T, operated_by, D) ∧ order_property(ON, assigned_to, D)] ⇒
∃t2:TIME t2>t1 state(γ, t2, output(OP))|=communicate_from_to(OP, CR, inform, order_state(ON, delay, severe_incident)),

where Table 4 provides the description of the predicates.

More examples of dynamic properties formalized in TTL will be given in Section 5.1.

The specification of both structural and dynamic properties in TTL is supported by a dedicated editor [2, 23]. The organizational model for the running example that comprises both static and dynamic aspects has been specified in this software. Furthermore, the software tool enables model execution (simulation) under different environmental conditions (i.e., temporal sequences of events). As a result of simulation, a trace can be generated and visualized. A fragment of the trace generated for the organizational model constructed for the running example is illustrated in Figure 5. Here, the time frame is depicted on the horizontal axis. The names of predicates are shown on the vertical axis. A dark box on top of the line indicates that the predicate is true during that time period.

Table 4. Predicates for formalizing the dynamic properties used in the examples

Predicate	Description
communicate_from_to(r1:ROLE, r2:ROLE, s_act:SPEECH_ACT, message:STRING)	Specifies the speech act *s_act* (e.g., inform, request, ask) from role-source *r1* to role-destination *r2* with the content *message*
deliverable_object(on: ORDER_NUM, desc:STRING)	Assigns the order number *on* with the description *desc* to the object that has to be delivered
truck_property(trt:TRUCK_TYPE, operated_by, d:DRIVER)	Assigns the driver *d* to a truck of the type *trt*
order_property(on:ORDER_NUM, assigned_to, d:DRIVER)	Assignes the order *on* to the driver *d*
order_property(on:ORDER_NUM, deadline, d_value:INTEGER)	Identifies the deadline *d_value* for the order *on*
truck_state(trt:TRUCK_TYPE, st:STATE, descr:STATE_DESCRIPTION)	Denotes the state *st* with the state description *descr* of a truck of the type *trt*
order_state(on:ORDER_NUM, st:STATE, descr:STATE_DESCRIPTION)	Specifies the state *st* with the state description *descr* of the order with the number *on*

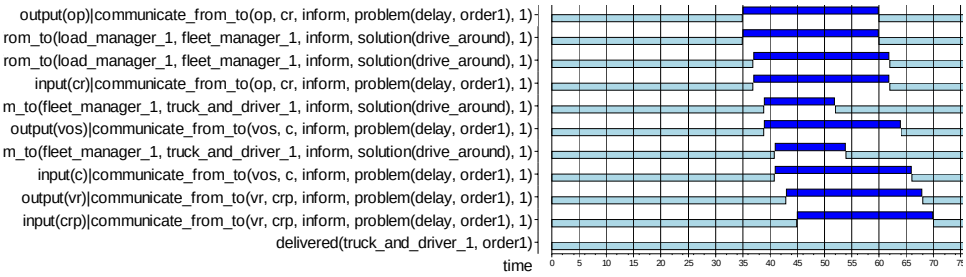

Fig. 5. An example of a visualized trace for the running example

4.3 Formalizing agent allocation principles

The formalization of agent allocation principles is performed in line with the formalization of the template and the deployed models, using the predicates specified in Table 5.

Table 5. Predicates for formalizing agent allocation principles

Predicate	Description
has_allocation_requirement: ROLE x REQUIREMENT	Specifies an allocation requirement for a role
has_capability: AGENT x CAPABILITY	Specifies a capability for an agent
allocated_to: AGENT x ROLE	Specifies an agent allocated to a role
corresponds_to: CAPABILITY x REQUIREMENT	Specifies an agent capability that corresponds to a role requirement

Generally, it is assumed that role requirements and agent capabilities are formulated using the same ontology, i.e., REQUIREMENT = CAPABILITY. However, if these ontologies are different, a necessary ontology mapping should be defined.

An agent can be allocated to a role if for every allocation requirement defined for the role the corresponding (equal in case of the same ontology) agent capability can be found. Formally:

allocated_to(a:AGENT, r: ROLE) ≡ ∀req:REQUIREMENT has_allocation_requirement(r, req) ⇒ [∃c:CAPABILITY has_capability(a, c) & corresponds_to(c, req)]

If after being allocated to the role, the agent looses one of his/her capabilities that correspond to the role requirements, then according to the rule above the current agent allocation will become false and the agent reallocation will be performed.

5 Verification and Validation

The model as introduced in this paper offers the means for both role-centered and agent-centered verification and validation. Role-centered verification techniques are dedicated for checking consistency and integrity of role-based organization models

without allocating agents to roles. These techniques are considered in Section 5.1. Whereas agent-centered verification approaches are applied for checking certain (general) dynamic properties on execution of different scenarios with roles of an organization model allocated to (human) agents. These techniques are described in Section 5.2. Both role- and agent-centered verification techniques are illustrated by applying them for checking the organizational model from the running example.

5.1 Role-centered verification techniques

In this paper two types of role-centered verification techniques are considered: (1) verifying consistency of an organizational model by checking relations between dynamic properties of different aggregation levels using model checking techniques [6] and (2) checking if an organizational role-based model complies with certain general requirements (expressed as dynamic properties) in different role-based simulation scenarios. Let us consider these techniques more in detail.

Checking interlevel relations between dynamic properties of different aggregation levels

When an organization model is specified including dynamic properties at different aggregation levels, it is not automatically guaranteed that the properties defined at adjacent aggregation levels fit to each other. A verification process that addreses interlevel relations between properties at one aggregation level and properties of adjacent aggregation level (e.g., as in compositional verification) can reveal incompleteness or inconsistencies in an organization model. The verification approach based on model checking techniques proposed in [40] can be used for justifying such relations. According to this approach dynamic properties of the lower aggregation level components (i.e., roles, links and the environment) expressed in TTL form a model that by means of the techniques described in [40] can be translated into the input format of one of the existing model checkers, and can be further used for automated verification. For practical verification the model checker SMV [6] has been chosen. A property of the higher aggregation level is required by SMV to be represented as a temporal formula in CTL [6]. This property will be automatically checked against all the possible executions of the translated model of the lower aggregation level by performing model checking. In such a manner, it can be proven that a property of the higher aggregation level is a logical consequence of the model that comprises properties of a lower aggregation level.

Let us illustrate this technique by applying it to the running example. The information distribution property RP1(OP) of role OP defined at aggregation level 2 and specified in Section 4.2 is used as a property of the higher aggregation level. For the purpose of verification, this property is expressed in CTL as follows:

AG (truck_state_T_incident_severe_incident & truck_property_T_operated_by_D & order_property_A20_assigned_to_D →
AF performing_action_preparation_output_OP_communicate_from_to_OP_CR_inform_order_state_A20_delay_severe_incident)

where ***A*** is a path quantifier defined in CTL, meaning "for all computational paths", ***G*** and ***F*** are temporal quantifiers that correspond to "globally" and "eventually" respectively.

The higher level property RP1(OP) can be logically related to the conjunction of dynamic properties of components at the lower aggregation level 3 in the following way:

EP1(Env, T, severe_incident) & EP2(Env, T) & EIP1(Env, D) & RP1(D) & TP1(D, FM) & RP2(FM) & TP2(FM, LM) & RP3(LM) & RP4(LM) & ILP1(LM, OP) $\Rightarrow$ RP1(OP) (1)

The abbreviations for the dynamic properties and their arguments conform to the specification provided in Section 3. Let us consider the informal and formalized expressions for some of the properties from the relation (1) that hold for any trace γ (the complete specification for the dynamic properties in (1) is given in Appendix A and in [21]):

EP1(Env, T, severe_incident) Incident occurrence

Informal description:

In the environment a severe incident with the truck T occurs

Formalization:

$\exists$t1:TIME state(γ, t1, environment)|= truck_state(T, incident, severe_incident)

EIP1(Env, D) Incident observation

Informal description:

If an incident happens with a truck, then a driver responsible for this truck will observe this incident

Formalization:

$\forall$t1:TIME $\forall$T:TRUCK_TYPE $\forall$D:DRIVER $\forall$ins:INCIDENT state(γ , t1, environment))|= [truck_state(T, incident, ins) $\wedge$ truck_property(T, operated_by, D)] $\Rightarrow$ $\exists$t2 t2>t1 state(γ , t2, input(D)) |= observation_result(truck_state(T, incident, ins), true)

RP1(D) Request for incident solution

Informal description:

If a driver observes an incident with his truck, then s/he will react by generating a request for advice to his fleet manager

Formalization:

$\forall$t1:TIME $\forall$T:TRUCK_TYPE $\forall$D:DRIVER $\forall$ins:INCIDENT $\forall$FM: FLEET_MANAGER state(γ , t1, input(D)) |= observation_result(truck_state(T, incident, ins), true) & state(γ, t1, environment) |= assigned_to(D, FM) $\Rightarrow$ $\exists$t2 t2>t1 state(γ , t2, output(D)) |= to_be_performed(communicate_from_to(D, FM, ask, solution_for_problem(ins, T)))

ILP1(LM, OP) Generation of information about the state change of a delivery order object

Informal description:

If a load manager communicates information about the change of a delivery status to the customer relation role, then the operational department role transmits this information to the customer relation department role.

Formalization:

$\forall$t1:TIME $\forall$LM: LOAD_MANAGER $\forall$ON:ORDER_NUM $\forall$st: STATE_TYPE $\forall$r: REASON state(γ, t1, output(LM))|= to_be_performed(communicate_from_to(LM, CR, inform, order_state(ON, st, r)))
$\Rightarrow$ $\exists$t2 t2>t1 state(γ, t2, output(OP))|= to_be_performed(communicate_from_to(OP, CR, inform, order_state(ON, st, r)))

By applying the algorithms and the dedicated software described in [40] to the specification that comprises all identified above properties defined at aggregation level 3 is transformed into the finite state transition system format required for performing mod-

el checking. Such a format consists of transition rules of the form [P → N], where P is a set of (predicate logic) atoms that are true in a current state and N is a set of atoms that will be true in the next state. For example, one of the transition rules from the obtained specification describes the generation of the memory state based on the observation of driver D at the time point t of the state property expressing that a severe incident happened with truck T:

present_time(t) & observed(input_D_truck_state_T_incident_severe_incident) →
memory(t, observed(input_D_truck_state_T_incident_severe_incident))

The following transition rule expresses the persistency of the created memory state:

memory(t, observed(input_D_truck_state_T_incident_severe_incident)) →
memory(t, observed(input_D_truck_state_T_incident_severe_incident))

The complete specification of the obtained finite state transition system for the considered example is given in Appendix B. The details of the procedure for transformation of a behavioral TTL specification into the finite state transition system format, and its application to the considered example are given in [21].

The automatic verification in the SMV model checking tool of the property RP1(OP) on the considered model showed that the previously identified logical relation (1) indeed holds. In general, the formal verification method of logical relations between dynamic properties of adjacent aggregation levels is useful for revealing missing premises or other shortcomings such as inconsistencies.

Checking global organizational properties with respect to a simulated role-based model

Another role-centered verification method is based on checking global organizational properties (or requirements) with respect to different executions of a role-based model by means of dedicated software. Such global organization properties are usually based on performance indicators of an organization, i.e., quantitative indicators that reflect the state, progress or performance of an organization (e.g., delivery time, customer notification time). By performing such verification inconsistencies and bottlenecks in an organization model can be detected.

Different executions (or execution traces) of a formally defined role-based organization model are obtained by performing simulations of different scenarios using a dedicated software environment [2]. Further the generated traces can be loaded into the verification environment, in which the formalized TTL properties can be checked on these traces.

Based on the formal organization model for the running example a simulation trace has been generated (given in Figure 5 partially), then the customer notification property has been checked on this trace.

Customer notification
Informal description:

Always if a severe problem occurs with the truck and the driver, who was fulfilling the order of some customer, then this customer should be notified about possible delay with delivery.
Formalization:

∀γ:TRACE ∀t1:TIME ∀T:TRUCK_TYPE ∀D:DRIVER ∀ON:ORDER_NUM state(γ, t1, environment))|= truck_state(T, incident, severe_incident) ∧ truck_property(T, operated_by, D) ∧ order_property(ON, assigned_to, D) ⇒ ∃t2:TIME t2>t1 ∃TCR:ROLE state(γ, t2, input(customer))|=communicate_from_to(TCR, customer, inform, order_state(ON, delay, customer_report))

An automatic verification confirmed that this property holds on the simulation trace.

5.2 Agent-centered verification technique

In this section an agent-centered verification technique is considered that is based on checking dynamic properties on a formalized empirical trace obtained by executing a particular scenario with roles of an organization model allocated to (human) agents. An empirical trace may be obtained from log-files of a company. If an empirical trace is given informally, the first step is to formalize it (by hand), using formal state ontologies. If it is already given in a formal form, the first step is to translate (e.g., automatically) the formal representation into one based on ontologies used in the organization model. Once such a trace is in the right formal form, it is possible to verify dynamic properties of the organization (including structural properties), using dedicated checking software as in the second role-centered verification technique.

As input for the verification software, a formalized trace and a formalized property have to be provided. Given such input, after automatic verification of the given property against the given trace, the software will generate a result (positive or negative). The positive decision confirms that the property holds with respect to the given trace. In case of a negative decision, the software explains why the property does not hold. In order to illustrate this method of verification, let us briefly consider the scenario reconstructed from empirical data of the transport company from the case study:

(1) A Customer places an order by means of a contact with TCR (CR department in this case) in CI.
(2) Inside TC this order is being transmitted from CR to OP.
(3) Within OP the order is distributed by SP to LM1.
(4) LM1 assigns the order to D1, D1 is associated with FM1 (see Fig. 4).
(5) D1 starts delivery, then after some time a severe incident occurs with his truck.
(6) D1 asks for help FM1, who is incapable of making a decision in this case.
(7) FM asks for a solution LM1, who decides to send another truck to proceed with delivery.
(8) Now D1 is reallocated to another truck and driver, who picks up goods and continues delivery.
(9) At the same time LM1 informs CR about possible delay with delivery.
(10) CR, who shares the same knowledge with TCR, informs the Customer about possible delay.
(11) D1 successfully finishes delivery and the Customer is being informed about that.

Using formal state ontologies (see Tables 2 and 3), we formalized this trace in the dedicated software environment. After that we identified several properties of interest that can be automatically verified against the trace. Let us consider two of them.

Delivery successfulness
Informal description:
The order has been fulfilled.
Formalization:

∃t:TIME ∃O:ORDER_NUM state(γ, t, environment)|= order_state(O, delivered, final_report)

An automatic verification confirmed that this property holds against the formalized empirical trace.

Delivery accuracy
Informal description:
The order has been fulfilled on time.
Formalization:
$\exists t{:}\text{TIME}\ \exists O{:}\text{ORDER\_NUM}\ \exists d\_value{:}\text{integer}\ state(\gamma, t, environment) \models order\_state(O, delivered, final\_report) \wedge order\_details(O, deadline, d\_value) \wedge d\_value \geq t$

This property does not hold with respect to the trace. The next logical step in analysis of the causes for property failing would be to check if some incident occurred in transit. In case that a severe incident happened with the truck and the agent (a truck driver) was incapable of performing his role any more, the next step would be to verify whether or not enough time is available for a role reallocation. Subsequently, analysis of organization functioning can be continued until all inquiries about delivery are satisfied.

If an agent allocated to a role possesses individual attitudes and behavioral characteristics that are not explicitly identified in role requirements, however which may influence the execution of functions associated with the role, then dedicated analysis techniques for determining consequences of different agent architectures for role performance can be applied. These techniques are not considered in this paper and will be described elsewhere.

6 Discussion

Both in human society and for software agents, organizational structure provides the means to make complex, composite dynamics manageable. To understand and formalize how exactly organization structure constrains composite dynamics is a fundamental challenge in the area of organizational modeling. The modeling approach presented in this paper addresses this challenge. It concerns a method for formal specification of organizations, which can capture both structural and dynamic aspects of organizations and provides the means for (i) representation of organization structure, (ii) simulations of different scenarios, (iii) analysis of organization, verifying static and dynamic properties against (formalized) empirical data or simulated scenarios, (iv) diagnosis of inconsistencies, redundancies, and errors in structure and functioning. Additionally, the environment is integrated as a special component within the organization model.

Specification of organization structure usually takes the form of pictorial descriptions, in a graph-like framework. These descriptions often abstract from detailed dynamics within an organization. Specification of the dynamic properties of organizations, on the other hand, usually takes place in a completely different conceptual framework; these dynamic properties are often specified in the form of a set of logical formulae in some temporal language. The logical relationships express the kind of relations between dynamics of parts of an organization, their interaction, and dynamic

properties of the organization as a whole, which were indicated as crucial by Lomi and Larsen [25] in their introduction.

This paper shows how pictorial descriptions, in a graph-like framework, and a set of logical formulae in some temporal language can be combined in one organization modeling approach. Inspection can be done on the abstraction level preferred and both the pictorial and formal specifications of the dynamic properties can be inspected. Five essential types of dynamic properties characterizing behavior of main structural components of an organization model (including environment) are identified.

Due to the high expressivity of the introduced modeling (structural and behavioral) languages, the proposed framework creates the formal fundament for developing more specific types of models that describe certain particular aspects of organizations (e.g., goals and tasks). Such models can be built by introducing new particular specifications for these aspects in terms of sorts, predicates, and properties, which represent instantiations of general types of static and dynamic properties described in this paper. In future work different particular perspectives on organizations (e.g., performance-orients, goal-oriented, process-oriented) will be elaborated.

Furthermore, the approach proposed here supports formal specification and verification for both static and dynamic properties. This possibility is especially useful for diagnosis of inconsistencies, redundancies, and errors in structure and functioning of real organizations and providing recommendations for their improvement (e.g., by way of evaluating of performance indicators). Compared to most organization-oriented, multi-agent system, design approaches [1, 10, 11, 42], our model allows any number of aggregation levels in the organization model, which makes it more suitable for modeling and analyzing real organizations. While a role aggregation relation is considered to be crucial for representing an organizational model, other types of relations between roles should also be taken into account. For example, a role specified in a template model and its corresponding role instances defined in a deployed model are related by means of a generalization relation. Furthermore, even more general role templates (or classes), which possess essential characteristics of roles of a certain type (e.g., seller, vendor, customer), independent of any application domain, can be created. Different types of relations between such roles can be identified (e.g., aggregation, generalization, interaction). Then, based on roles classes and their relations libraries can be created that can be used for the specification of a template organizational model. Moreover, such libraries may be employed for constructing templates of different types of organizations. Both structural and dynamic aspects of different types of organizations should be reflected in such templates; for this formal languages introduced in this paper can be used. To identify the distinctive features of different organization types, agent-based models identified in [36] and the literature from organization theory [28, 30] are useful to consider.

Let us now consider a case in which agents show autonomous behavior, independent of (or sometimes conflicting to) organizational rules and goals. To tackle the forthcoming problems from such settings, further investigation of the relationships between formally predefined organizational model and agent autonomous behavior in settings of different types of organizations will be undertaken. The work on holonic structures [36, 37] may be relevant for further investigations on this question. By applying the approach introduced in this paper the specifications of a (hierarchical)

structure and dynamics can be developed, which describe a certain holon, or are imposed on agents within a holon. The specification of autonomous agent behavior takes place in a different conceptual framework, which, nevertheless, can be related (at least in ontological sense) to the modeling framework introduced in this paper. Then, by varying the types and flexibility of the (imposed) structures and behaviors (using for example the types described in [36]), and the level of agent autonomy, different types of organizations represented by multi-agent systems can be investigated. Furthermore, by applying analysis methods described in this paper the behavior of holons can be checked for compliance with the prescribed norms and other (global) properties of an organization.

In the case of highly dynamic organizations (e.g., self-organizing and organic organizations), organizational change is a crucial and frequent process. Due to their high complexity, such organizations are difficult to investigate. However, different simulation techniques can help in providing further insights into mechanisms of functioning of such organizations. For the latter purpose, research has been conducted based on the introduced formal model [18].

In conclusion, this paper introduced a new, formal, fully traceable method on modeling and analyzing (multi-agent) organizations. It comprises both static and dynamic aspects as well as environment representation. Hence, it provides the basis of a formal framework, which provides the means for both the design and for the automatic validation and verification of organizations.

Acknowledgments

This research was partially supported by the Netherlands Organization for Scientific Research (NWO) under project number 612.062.006. SenterNovem is gratefully acknowledged for funding the projects Cybernetic Incident Management (CIM) and Distributed Engine for Advanced Logistics (DEAL) that also funded this research partially. Further, we thank the reviewers for their detailed comments on the original manuscript.

References

1. M. Amiguet, J.-P. Mueller, J.-A. Baez-Barranco, and A. Nagy, “The MOCA Platform”, in Proc. of MABS, 2002, pp. 70-88.
2. T. Bosse, C.M. Jonker, L. Meij, and J. Treur, “LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn”, in Proc. of the Third German Conference on Multi-Agent System Technologies, MATES'05, edited by T. Eymann, F. Kluegl, W. Lamersdorf, M. Klusch, and M.N. Huhns, Lecture Notes in Artificial Intelligence, vol. 3550, Springer Verlag, 2005, pp. 165-178.
3. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, “Tropos: An Agent-Oriented Software Development Methodology”, Journal of Autonomous Agent and Multi-Agent Systems, vol. 8(3), pp. 203-236, 2004.

4. R. M. Burton and B. Obel, Strategic Organizational Diagnosis and Design: Developing Theory for Application, Kluwer Academic Publishers: Dordrecht, 2004.
5. K. Carley and J.-S. Lee, “Dynamic Organizations: Organizational Adaptation in a Changing Environment”, Disciplinary Roots of Strategic Management Research, vol. 15, pp. 267-295, 1998.
6. E.M. Clarke, O. Grumberg, and D.A. Peled, Model Checking. MIT Press, 2000.
7. M. Dastani, J. Hulstijn, F. Dignum, and J-J. Meyer, “Issues in Multiagent System Development”, in Proc. of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems AAMAS'04, 2004, pp. 922-929.
8. R. David and H. Alla, Discrete, Continuous, and Hybrid Petri Nets. Springer-Verlag, 2005.
9. M. Esteva, D. Cruz, and C. Sierra, “ISLANDER: an electronic institutions editor”, in Proc. of the 1st International Conference on Autonomous Agents and Multiagent systems, 2002, pp. 1045-1052.
10. J. Ferber and O.Gutknecht, “A meta-model for the analysis and design of organizations in multi-agent systems”, in Proc. of Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society, 1998, pp. 128-135.
11. J. Ferber, O. Gutknecht, and F.Michel, “From Agents to Organizations: an Organizational View of Multi-Agent Systems”, in Proc. of 4th International Workshop AOSE, 2003, pp. 214-230.
12. J.W. Forrester, Industrial dynamics, Waltham, MA: Pegasus Communications, 1961.
13. M. Fox, M. Barbuceanu, M. Gruninger, and J. Lin, “An Organization Ontology for Enterprise Modelling”, in Simulating Organizations: Computational Models of Institutions and Groups, edited by M. Prietula, K. Carley and L. Gasser, Menlo Park CA: AAAI/MIT Press pp. 131-152, 1997.
14. A. Galton, Temporal Logic, in Stanford Encyclopedia of Philosophy, 2003. URL: http://plato.stanford.edu/entries/logic-temporal/#2.
15. A. Galton, “Operators vs Arguments: The Ins and Outs of Reification”, Synthese, vol. 150, pp. 415-441, 2006.
16. M. Hannoun, J.S. Sichman, O. Boissier, and C. Sayettat, “Dependence Relations between Roles in a Multi-Agent System: Towards the Detection of Inconsistencies in Organization”, in Proc. of MABS, 1998, pp. 169-182.
17. A. Hodgson, R. Roennquist, P. Busetta, and N. Howden, “Team Oriented Programming with SimpleTeam”, in Proc. of SimTecT 2000, Sydney, Australia, 2000, pp. 115-122.
18. M. Hoogendoorn, C.M. Jonker, M. Schut, and J. Treur, “Modelling the Organisation of Organisational Change”, in Proc. of the Sixth International Workshop on Agent-Oriented Information Systems, 2004, pp. 29-46. Extended version: Jounal of Computational and Mathematical Organisation Theory. In press, 2006.
19. B. Horling, V. Lesser, “A Survey of multi-agent organizational paradigms”, The Knowledge Engineering Review, Vol. 19(4), pp. 281-316, 2005.
20. J.F. Hubner, J.S. Sichman, O. Boissier, “A Model for the Structural, Functional and Deontic Specification of Organizations in Multiagent Systems”, in Proc. of SBIA, 2002, pp. 118-128.
21. C.M. Jonker, A. Sharpanskykh, J. Treur, and P. Yolum, “Verifying Interlevel Relations within Organizational Models”, Technical Report #TR-061909AI, Vrije Universiteit Amsterdam, 2006. URL: http://hdl.handle.net/1871/10210
22. C.M. Jonker and J. Treur, “A temporal-interactivist perspective on the dynamics of mental states”, Cognitive Systems Research Journal, 4(3), pp. 137-155, 2003.
23. C.M. Jonker, J. Treur, and W.C.A. Wijngaards, “A temporal-modelling environment for internally grounded beliefs, desires, and intentions”, Cognitive Systems Research Journal, 4(3), pp. 191-210, 2003.

24. R. Kowalski and M. Sergot, “A logic-based calculus of events”, New Generation Computing, 4, pp. 67-95, 1986.
25. A. Lomi and E.R. Larsen, Dynamics of Organizations: Computational Modeling and Organization Theories, AAAI Press, Menlo Park, 2001.
26. F. Lopez y Lopez, M. Luck, and M. d'Inverno, “A Normative Framework for Agent-Based Systems”, Computational and Mathematical Organization Theory, 12(2-3), pp. 227-250, 2005.
27. M. Manzano, Extensions of First Order Logic, Cambridge University Press, 1996.
28. H. Mintzberg, The Structuring of Organizations, Prentice Hall, Englewood Cliffs, 1979.
29. R.E. Miles, C.C. Snow, J.A. Mathews, and H.J. Coleman, “Organizing in the knowledge age: Anticipating the cellular form”, Academy of Management Executive, 11(4), pp. 7-20, 1997.
30. G. Morgan, Images of organizations, SAGE Publications, Thousand Oaks London New Delhi, 1996.
31. J. Odell, H.V.D. Parunak, and B. Bauer, “Extending UML for Agents”, in Proc. of Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence, 2000, pp. 3-17.
32. A. Omicini, “SODA: Societies and infrastructures in the analysis and design of agent-based systems”, in Proc. of AOSE, 2000, pp. 185–193.
33. H.V.D. Parunak and J. Odell, “Representing Social Structures in UML”, in Proc. of Agent-Oriented Software Engineering II Workshop, edited by M. Wooldridge, G. Weiss, and P. Ciancarini, Lecture Notes on Computer Science, vol. 2222, Springer-Verlag, Berlin, pp. 1-16, 2002.
34. M. Prietula, L. Gasser, K. Carley, Simulating Organizations, MIT Press, 1997.
35. R. Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical System, Cambridge MA: MIT Press, 2001.
36. M. Schillo, “Self-Organization and Adjustable Autonomy: Two Sides of the Same Coin?”, Connection Science, vol. 14 (4), 2003, pp. 345-360.
37. M. Schillo and D. Spresny, “Organization: The Central Concept for Qualitative and Quantitative Scalability”, in Socionics: Contributions to the Scalability of Complex Social Systems edited by K. Fischer and M. Florian, Lecture Notes in Artificial Intelligence, Berlin, vol. 3413, Springer, 2005, pp. 84-103.
38. J. Scott, Social Network Analysis: A Handbook, 2nd Ed. Newberry Park, CA: Sage, 2000.
39. W.R. Scott, Institutions and organizations, SAGE Publications, Thousand Oaks London New Delhi, 2001.
40. A. Sharpanskykh and J. Treur, “Verifying Interlevel Relations within Multi-Agent Systems”, in Proc. of the 17th European Conference on Artificial Intelligence, ECAI'06. IOS Press, 2006, pp. 290-294.
41. J. Vázquez-Salceda, H.M. Aldewereld, and F.P.M. Dignum, “Norms in multiagent systems: From theory to practice”, International Journal of Computer Systems Science & Engineering, vol. 20(4), pp. 225-236, 2005.
42. F. Zambonelli, N. R. Jennings, M. Wooldridge, “Developing multiagent systems: the Gaia Methodology”, ACM Transactions on Software Engineering and Methodology, vol. 12 (3), 2003, pp. 317-370.

Appendix A. The complete specification of dynamic properties from the running example

EP1(Env, T, severe_incident) Incident occurrence

Informal description:

In the environment a severe incident with the truck T occurs

Formalization:

∃t1:TIME state(γ, t1, environment)|= truck_state(T, incident, severe_incident)

EP2(Env, T) Stable information about the environnent

Informal description:

Role D (a driver) operates the truck T and is assigned to deliver the order A20; role D is assigned to the fleet manager FM, and FM is in the region of the load manager LM

Formalization:

∀t1:TIME state(γ, t1, environment) |= [truck_property(T, operated_by, D) ∧ order_property(A20, assigned_to, D) ∧ assigned_to(D, FM) ∧ in_region(FM, LM)]

EIP1(Env, D) Incident observation

Informal description:

If an incident happens with a truck, then a driver responsible for this truck will observe this incident

Formalization:

∀t1:TIME ∀T:TRUCK_TYPE ∀D:DRIVER ∀ins:INCIDENT state(γ , t1, environment))|= [truck_state(T, incident, ins) ∧ truck_property(T, operated_by, D)] ⇒ ∃t2 t2>t1 state(γ , t2, input(D)) |= observation_result(truck_state(T, incident, ins), true)

RP1(D) Request for incident solution

Informal description:

If a driver observes an incident with his truck, then s/he will react by generating a request for advice to his fleet manager

Formalization:

∀t1:TIME ∀T:TRUCK_TYPE ∀D:DRIVER ∀ins:INCIDENT ∀FM: FLEET_MANAGER state(γ , t1, input(D)) |= observation_result(truck_state(T, incident, ins), true) & state(γ, t1, environment) |= assigned_to(D, FM)
⇒ ∃t2 t2>t1 state(γ , t2, output(D)) |= to_be_performed(communicate_from_to(D, FM, ask, solution_for_problem(ins, T)))

TP1(D, FM) Request transfer to Fleet Manager

Informal description:

If a driver sends a request to his fleet manager, the fleet manager will receive this request

Formalization:

∀t1:TIME ∀D:DRIVER ∀FM: FLEET_MANAGER ∀req: REQUEST state(γ, t1, output(D)) |= to_be_performed(communicate_from_to(D, FM, ask, req)) & state(γ, t1, environment) |= assigned_to(D, FM)
⇒ ∃t2 t2>t1 state(γ , t2, input(FM)) |= observation_result(communicate_from_to(D, FM, ask, req))

RP2(FM) Request for solution propagation

Informal description:

If a fleet manager receives a request from a driver for advice to solve a severe problem, then s/he will propagate this request further to the regional load manager

Formalization:

∀t1:TIME ∀D:DRIVER ∀T:TRUCK_TYPE ∀FM: FLEET_MANAGER ∀LM: LOAD_MANAGER state(γ , t1, input(FM)) |= observation_result(communicate_from_to(D, FM, ask, solution_for_problem(severe_incident, T))) & state(γ, t1, environment) |= in_region(FM, LM)

⇒ ∃t2 t2>t1 state(γ , t2, output(FM)) |= to_be_performed(communicate_from_to(FM, LM, ask, solution_for_problem(severe_incident, T)))

TP2(FM, LM) Request transfer to Load Manager

Informal description:

If a fleet manager sends a request to a regional load manager, the regional load manager will receive this request

Formalization:

∀t1:TIME ∀FM: FLEET_MANAGER ∀LM: LOAD_MANAGER ∀req: REQUEST state(γ ,t1, output(FM)) |= to_be_performed(communicate_from_to(FM, LM, ask, req))
⇒ ∃t2 t2>t1 state(γ , t2, input(LM)) |= observation_result(communicate_from_to(FM, LM, ask, req))

RP3(LM) Change of a delivery status

Informal description:

If a load manager receives a request from a fleet manager for advice to solve a severe problem, then s/he officially identifies the incident as severe and changes into "delay" the state of the corresponding delivery order in the information system.

Formalization:

∀D:DRIVER ∀t1:TIME ∀FM: FLEET_MANAGER ∀T:TRUCK_TYPE ∀LM: LOAD_MANAGER ∀ON:ORDER_NUM state(γ , t1, input(LM)) |= observation_result(communicate_from_to(FM, LM, ask, solution_for_problem(severe_incident, T))) & state(γ, t1, environment) |= [order_property(ON, assigned_to, D) ∧ truck_property(T, operated_by, D)]
⇒ ∃t2 t2>t1 state(γ , t2, output(LM)) |= to_be_performed(change(order_state(ON, delay, severe_incident)))

RP4(LM) Informing CR about a delivery status

Informal description:

If a load manager changes a state of a delivery order object, then the information about this change is generated at the output of the load manager role for the customer relation role.

Formalization:

∀t1:TIME ∀LM: LOAD_MANAGER ∀ON:ORDER_NUM ∀st: STATE_TYPE ∀r: REASON state(γ , t1, output(LM)) |= to_be_performed(change(order_state(ON, st, r)))
⇒ ∃t2 t2>t1 state(γ, t2, output(LM))|= to_be_performed(communicate_from_to(LM, CR, inform, order_state(ON, st, r)))

ILP1(LM, OP) Generation of information about the state change of a delivery order object

Informal description:

If a load manager communicates information about the change of a delivery status to the customer relation role, then the operational department role transmits this information to the customer relation department role.

Formalization:

∀t1:TIME ∀LM: LOAD_MANAGER ∀ON:ORDER_NUM ∀st: STATE_TYPE ∀r: REASON state(γ, t1, output(LM))|= to_be_performed(communicate_from_to(LM, CR, inform, order_state(ON, st, r)))
⇒ ∃t2 t2>t1 state(γ, t2, output(OP))|= to_be_performed(communicate_from_to(OP, CR, inform, order_state(ON, st, r)))

Appendix B. The complete specification of the transition system from the running example

present_time(t) &
¬performing_action(preparation(output_LM_communicate_from_to_LM_CR_inform_order_state_A20_delay_severe_incident)) ⟶ present_time(t+1)

truck_state_T_incident_severe_incident & truck_property_T_operated_by_D ⇾
observed(input_D_truck_state_T_incident_severe_incident)

present_time(t) & observed(input_D_truck_state_T_incident_severe_incident) ⟶ memory(t,
observed(input_D_truck_state_T_incident_severe_incident))

memory(t, observed(input_D_truck_state_T_incident_severe_incident)) ⟶ memory(t,
observed(input_D_truck_state_T_incident_severe_incident))

present_time(t) & memory(t, observed(input_D_truck_state_T_incident_severe_incident)) & assigned_to_D_FM
⟶ qcprep1

present_time(t) & qcprep1 ⇾
preparation(output_D_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T)

preparation(output_D_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T) ⇾
performing_action(output_D_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T)

performing_action(output_D_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T) ⇾
observed(input_FM_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T)

present_time(t) &
observed(input_FM_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T) ⟶ memory(t,
observed(input_FM_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T))

memory(t, observed(input_FM_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T))
⟶ memory(t,
observed(input_FM_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T))

present_time(t) & memory(t,
observed(input_FM_communicate_from_to_D_FM_ask_solution_for_problem_severe_incident_T)) &
in_region_FM_LM ⟶ qcprep2

present_time(t) & qcprep2 ⇾
preparation(output_FM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T)

preparation(output_FM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T) ⇾
performing_action(output_FM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T)

performing_action(output_FM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T)
⟶ observed (input_LM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T)

present_time(t) & observed
(input_LM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T) ⟶ memory(t,
observed(input_LM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T))

memory(t, observed(input_LM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T))
⇾
memory(t, observed(input_LM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T))

present_time(t) & memory(t,
observed(input_LM_communicate_from_to_FM_LM_ask_solution_for_problem_severe_incident_T)) &
order_property_A20_assigned_to_D & truck_property_T_operated_by_D ⟶ qcprep3

present_time(t) & qcprep3 ⟶ preparation(change_order_state_A20_delay_severe_incident)

preparation(change_order_state_A20_delay_severe_incident) ⇾
performing_action(preparation(change_order_state_A20_delay_severe_incident))

performing_action(preparation(change_order_state_A20_delay_severe_incident)) ⇾
performing_action(preparation(output_LM_communicate_from_to_LM_CR_inform_order_state_A20_delay_severe_incident))

performing_action(preparation(output_LM_communicate_from_to_LM_CR_inform_order_state_A20_delay_severe_incident)) ⇾
performing_action(preparation(output_OP_communicate_from_to_OP_CR_inform_order_state_A20_delay_severe_incident))