A Framework for Constraint-based Modelling and Analysis of Organisations

Viara Popova*, Alexei Sharpanskykh **

** De Montfort University, Centre for Manufacturing, Leicester, UK (e-mail: vpopova@dmu.ac.uk)*
*** Vrije Universiteit Amsterdam, Department of Artificial Intelligence, The Netherlands (e-mail: sharp@cs.vu.nl)*

Abstract: Modern organisations are characterised by a great variety of forms and often involve many actors with diverse goals, performing a wide range of tasks in changing environmental conditions. Due to high complexity, mistakes and inconsistencies are not rare in organisations. To provide better insights into the organisational operation and to identify different types of organisational problems explicit specification of relations and rules, on which the structure and behaviour of an organisation are based, is required. Before it is used, the specification of an organisation should be checked for internal consistency and validity w.r.t. the domain. To this end, the paper introduces a framework for formal specification of constraints that ensure the consistency and validity of organisational specifications. To verify the satisfaction of constraints efficient algorithms have been developed and implemented. The application of the proposed approach is illustrated by a case study from the air traffic domain.

Keywords: Organisation modelling, constraints, organisation design, consistency of a specification, validity of a specification.

1. INTRODUCTION

Due to a high structural and behavioural complexity of many modern organisations, mistakes and performance inefficiencies often occur. With the growth of complexity, the possibility of rapid and reliable analysis of organisations becomes increasingly important. Manual analysis can be cumbersome, error-prone and slow. Automated, formally grounded analysis is devoid of these issues. To enable automated analysis, a formal specification of an organisation reflecting its structural and behavioural dependencies and rules, is required. Moreover, a formally defined organisational specification provides a basis for automated processes (Bernus, 1998) and for inter-enterprise cooperation.

To represent diverse aspects of organisations, highly expressive formal modelling languages are required. To decrease the complexity of modelling, several dedicated perspectives (or views) on organisations are often distinguished. For example the meta-framework GERAM (a template for development of frameworks for enterprise modelling based on common features of several architectures e.g. CIMOSA, TOVE, ARIS) defines four views: function, information, resource and organisation; see (Bernus, 1998) for an overview on these frameworks. The framework in (Sharpanskykh, 2008), used for defining the specifications in this paper, four interrelated views (that can be mapped to those in GERAM) with dedicated predicate logic-based languages are defined: performance-, process-, organisation- and agent-oriented. The internal consistency and validity of organisational specifications is ensured by a set of constraints that should be satisfied by them. A framework for modelling and verification of constraints is the main contribution of this paper. *A constraint* is a restriction imposed on aspect(s) of the organisational structure and/or behaviour. Syntactically, a constraint is an expression over the language of a view(s), i.e. a formula constructed from terms of one or more of these languages using Boolean connectives and quantifiers. A constraint is *satisfied* by a specification, iff in all models of the specification the constraint is evaluated to true.

The languages of the views allow specifying a great variety of constraints. Some constraints define meta-rules for the correct use of language concepts in a view based on its semantics. If a set of all such constraints defined for the view is satisfied on a specification, then the specification is called *consistent*. Other constraints reflect domain-specific regulations of a particular organisation(s) and are required to be satisfied by the specifications of the organisation(s). If a specification satisfies all domain-specific constraints, and the constraints imposed by the environment of the organisation (physical world), then the specification is called *valid*. The algorithms for verification of consistency and validity of specifications were developed and implemented.

The consistency and validity of a specification do not guarantee its flawless execution by agents. Currently many organisations use information systems controlling the process execution. Such systems may also be used to check the correspondence of actual executions of organisational scenarios to the organisational specification. To enable such verification, a special class of constraints is required, which is addressed here. Diverging behaviour of agents may influence organisational performance. Automated analysis methods to identify such influences are considered in this paper.

Section 2 outlines the approach for organisation modelling focusing on the organisation-oriented view. Section 3 introduces the framework for constraints. Constraints from the case study and checking algorithms are given in Section 4. Section 5 discusses the related literature.

2. THE ORGANISATION MODELLING FRAMEWORK

The organisation modelling framework (as presented in (Sharpanskykh, 2008)) has the following characteristics. It allows the representation and analysis of organisation models at *different levels of abstraction* in order to handle complexity and increase scalability. It enables *formal verification and validation* of models of different perspectives on organisations. It also enables *simulation* of the organisational behaviour under different circumstances. The framework proposes computational *analysis methods across multiple perspectives* on organisations as well as supports and controls the *execution of organisational scenarios* and the *evaluation of organisational performance*.

The framework consists of four interrelated views (Sharpanskykh, 2008) describing different aspects of the organisation. The ***process-oriented view*** of the framework specifies the organisational functions, how they are related, synchronized, ordered and the resources they use or produce. The ***performance-oriented view*** specifies performance-related concepts such as goals and performance indicators and how they are related. The ***agent-oriented view*** describes the agents, their capabilities, how they are assigned to organisational roles and allows representing both intentional (e.g., goals) and motivational aspects of agent behaviour. For the purpose of this paper, the discussion will focus on the ***organisation-oriented view***. For each view a dedicated predicate logic-based language has been developed. To express temporal relations, the dedicated languages of the views are embedded into the Temporal Trace Language (TTL) (Sharpanskykh, 2008), which is a variant of the order-sorted predicate logic. In TTL the organisational dynamics are represented by a trace, i.e. a temporally ordered sequence of states. Each state is characterized by a unique time point and a set of state properties that hold (are true) and are expressed using the dedicated languages of a view(s). Temporal (dynamic) properties are defined in TTL as transition relations between state properties.

For the further illustration of the organisation-oriented view and of different types of constraints, a case study from the air traffic domain is used (Sharpanskykh, 2008). Two tasks of an Air Traffic Control Organisation (ATCO) have been modelled and analysed: movement of aircraft on the ground, and incident reporting. The ground movement task includes taxiing of aircraft to the designated runway across sectors of the airport, and the subsequent take off from this runway. The monitoring and the traffic control in a sector or runway are performed by a dedicated air traffic controller. During taxiing the control over an aircraft is handed over from one controller to another, depending on the physical position of the aircraft. In case an incident/accident occurs during taxiing or take off, this should be reported and investigated.

In the ***organisation-oriented view*** organisations are modelled as composite roles that can be refined iteratively into composite or simple roles, representing as many aggregation levels as needed. The refined role structures correspond to different types of organisation constructs (e.g., groups, units, departments). The view provides means to structure roles by defining interaction and power relations on them.

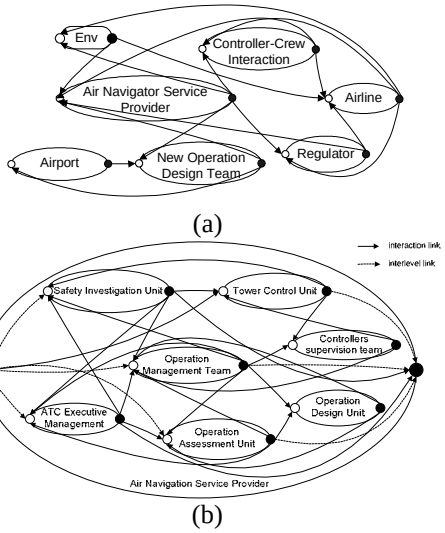

Fig. 1. Interaction relations in the air traffic organisation shown at aggregation level 1 (a) and in the composite role Air Navigator Service Provider at aggregation level 2 (b).

Each role has an input and an output interface, which facilitate interaction (e.g., communication) with other roles and the environment. Role interfaces are described in terms of input and output ontologies: a signature specified in order-sorted logic. In general, an input (resp. output) ontology determines what types of information are allowed to be transferred to (generated at) the input (output) of a role or the environment. Roles that are allowed to interact are connected by an interaction link that indicates the direction of interaction. Interaction between adjacent aggregation levels is enabled by interlevel links. For example, the interaction relations between the roles at aggregation level 1 of the organisation from the case study are depicted in Figure 1a. Figure 1b zooms into role Air Navigator Service Provider (ANSP) and presents its subroles at aggregation level 2. Consider an interaction example. Information generated at the output of role Operation Management Team (OMT) at aggregation level 2 for role Regulator is transmitted first to the output of role ANSP via an interlevel link. Then, at aggregation level 1 it is transferred from the ANSP's output to the input of role Regulator via an interaction link.

Besides interaction, also power relations on roles are a part of the formal organisational structure. Formal power (authority) establishes and regulates normative superior-subordinate relations on roles w.r.t. tasks. Roles may have responsibilities and rights w.r.t. specific aspects of tasks: e.g., monitoring, execution, making decisions. Roles with managerial rights may authorize (make other roles responsible) for some aspects of tasks, e.g., for the task "Investigation of occurrence based on the notification report" the Safety Investigator is responsible for execution and the Head Safety Investigation Unit for monitoring, consulting and managerial decisions.

3. THE FRAMEWORK FOR ORGANISATIONAL CONSTRAINTS

To ensure consistency and validity of specifications, constraints are integrated in the general framework for

organisation modelling and analysis (Sharpanskykh, 2008). This Section describes the constraints, how they relate to the specification and how they are used for analysis.

Two main groups of constraints were identified: specification and execution constraints. *Specification constraints* are used to ensure *consistency* and *validity* of an organisational specification. If an organisation is designed from scratch, a methodology analogous to concurrent engineering may be followed: based on requirements a (partial) specification of the organisation is defined, which should satisfy specification constraints. *Execution constraints* are used to ensure that the actual execution of organisational scenarios conforms to the specification. Figure 2 represents at a high abstraction level the mechanisms of design, execution and evaluation of an organisational specification w.r.t. constraints. During the design phase, the specification constraints are checked on the specification to verify the consistency and validity of a specification. A consistent and valid specification can be translated to execution constraints. The specification can then be executed with the help of the organisation's automated managements system. The system records the necessary data related to the execution which forms execution traces. Aspects monitored and recorded in traces include started/finished processes, produced/used resources, measured values of performance indicators, roles assigned to agents and agents performing processes, etc. The set of execution constraints can then be checked on the execution traces to ensure that they conform to the specification.

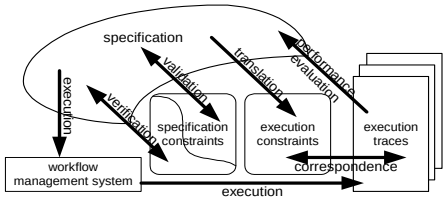

Fig. 2. Design, execution and evaluation of a specification

The role of constraints may differ in organisation modelling which influences their format, purpose and way of use. Here we present a classification framework for constraints covering a range of perspectives on organisations from very detailed to global, from specification to actual execution and from internal to external point of view, connecting the organisation with its environment.

Specification constraints can be checked at every step of the (re)design process in order to ensure the consistency and validity of the current specification. They can be classified in two dimensions based on their scope and on their origin.

Based on their origin, the specification constraints can be classified into: *generic constraints* that need to be satisfied by any specification built using the framework; *domain-specific constraints* dictated by the application domain of the specification. Two types of generic constraints are considered: *structural consistency constraints* used to ensure consistency of the specification; *constraints imposed by the physical world* - the laws of the physical world render certain events, relationships between concepts, etc. impossible (e.g. a resource cannot be at two locations at the same time).

The *consistency* of a specification is checked w.r.t. the set of structural consistency constraints. These constraints are axioms of the specification language and their logical consequences formulated based on definitions of the language and reflecting the rules of correct and consistent use of the elements of the language in modelling. These constraints ensure internal integrity of the structures defined using the language. A specification S is consistent w.r.t. a set of structural consistency constraints SCC iff in each of its models each formula from SCC is true: $\langle I, v \rangle \models SCC$, where I is an interpretation of the sorts, functions and predicates of the language of S and v is a valuation of variables in S.

Domain-specific constraints are imposed by the application domain in which the particular specification will be used and can be classified according to their sources:

Constraints imposed by the organisation have been chosen (e.g. by the management of the company) as necessary and need to be satisfied by any specification for the particular organisation. Such constraints can often be found in company policy documents, internal procedures descriptions, etc. *Constraints coming from external parties* are enforced by external parties (e.g. the society or government) and contain rules about working hours, safety procedures, emissions, etc. Sources for such constraints are regulations, agreements, etc. *Constraints of the physical world* come from the physical world w.r.t. the specific application domain and should be satisfied by any specification in this domain (in contrast to the generic physical constraints which should be satisfied by any specification irrespective of the application domain).

The *validity* of a specification is checked w.r.t. a set of physical-world and domain-specific constraints. An organisational specification S is *valid* w.r.t. a set of physical world and domain-specific constraints C iff in each of its models each formula from C is true: $\langle I,v \rangle \models C$.

To reduce the complexity of modelling and analysis, organisational specifications can be considered at different aggregation levels (e.g., to investigate certain organisational aspects, while abstracting from irrelevant details). To ensure consistency of specifications and sets of constraints of different aggregation levels, and integrity of a complete organisational specification, a set of *interlevel consistency constraints* is defined. A part of these constraints belong to the class of generic structural consistency constraints, examples of which are given in Section 4; the rest are domain-specific and should be identified and checked for a particular organisation. To this end, the automated method for verifying relations between properties of different aggregation levels described in (Sharpanskykh, 2008) is used.

Based on the scope, specification constraints are divided as follows. *Constraints within one structure* ensure its consistency and validity. Several structures can be defined in the specification, e.g.: role and authority structures in the organisation-oriented view. Each structure can have specific constraints expressed using the language of the view. When the structure is hierarchical a specific type of constraints is defined to preserve the consistency between the aggregation levels of the structure. The relevant aspects vary depending

on the structure, e.g. inheritance of characteristics, uniqueness of an object in a structure, etc. Example of a hierarchical structure is the role decomposition structures. *Constraints within one view* are expressed using the language of the corresponding view. Some take care of the consistency between related structures within the view e.g. the interaction and the authority structures of the organisation-oriented view. *Constraints between views* are expressed using the union of the languages of the corresponding views. Some of them ensure the consistency of the specification. *Constraints between the specifications of different organisations* ensure that organisations related by contract/cooperation are aligned to achieve successful interaction, e.g. a supply chain that coordinates its operations by making sure the involved parties have the necessary goals, processes, resources, etc.

The second main group of constraints are the ***execution constraints***. They are based on the specification but need to be satisfied by the actual execution traces ensuring that the execution traces conform to the specification. Execution constraints are defined using a dedicated formal language for expressing execution traces from (Popova and Sharpanskykh, 2007). An execution trace is a time-indexed sequence of states, where each state is represented by a conjunction of state properties that hold at that time point. A trace model is defined as $T = ((0, P_0), (1, P_1), \dots (n, P_n))$, for P_i – conjunctions of state properties that hold. An execution trace is correct if the set of execution constraints EC is true in trace model T: T|=EC.

Traces can be incorrect even when they are based on correct specifications due to deviations caused by unforeseen circumstances, agent mistakes, etc. Execution constraints can be checked at runtime at every step or after the trace is completed. It is also possible to formulate execution constraints for multiple organisations so that alignment of the organisations is ensured in all their actual execution traces.

To facilitate the specification of complex constraints, parameterized templates are introduced, which act as shortcuts for complex logical expressions, and can easily be used by non-professionals in logics. Such templates can be selected and customized by the designer by assigning specific values to the parameters. Examples are given in Section 4.

4. CONSTRAINTS IDENTIFIED FOR THE CASE STUDY

This Section gives examples of constraints identified for the *organisation-oriented view* of the case study following the classification in Section 3. All constraints are formalized using the dedicated languages of the views and TTL. Here formal representation is only provided for some constraints.

The ***organisation-oriented view*** describes two types of structures: an interaction structure and an authority structure. First, constraints over an interaction structure are considered. Examples of generic constraints that ensure the interlevel consistency of an interaction structure are the following:

CS1: A role can be a subrole of one role at most.
In the structure Γ $\forall$r, r1, r2: ROLE subrole_of_in(r, r1, Γ) & subrole_of_in(r, r2, Γ) $\Rightarrow$ r2=r1

CS2: Each subrole of a composite role r should interact with at least one other subrole of r.
In the structure Γ $\forall$r1: ROLE subrole_of_in(r1, r, Γ) $\Rightarrow$ $\exists$r2:ROLE $\exists$e:INTERACTION_LINK subrole_of_in(r2, r, Γ) & (connects_to(e, r2, r1, Γ) | connects_to(e, r1, r2, Γ))

CS3: Information provided to the input of a composite role should be further transmitted to one or more its subroles.

Several examples of domain-specific constraints over an interaction structure are:

CS4: Information of a type inf produced by a role r1 for a role r2 should be able to reach r2.

For checking if a path exists for communicating inf from r1 to r2 the following algorithm is proposed.

Algorithm 1: CHECK-EXISTENSE-OF-INTERACTION-PATH

```
1  i ← max (AGR_LEVEL(r1), AGR_LEVEL(r2)), rt1 ← r1, rt2 ← r2
2  if  ∃rh1, rh2 ∃Γ  subrole_of_in(rt1, rh1, Γ) and
3      subrole_of_in(rt2, rh2, Γ) and rh1 = rh2,
4  then if  IS_PATH_FROM_TO_FOR(rt1, rt2, inf) = true,
5              then return true, else return false.
6  if rt1 = r2 or rt2 = r1, then return true.
7   if AGR_LEVEL(r1) >= i and  ∃rh1 ∃Γ  subrole_of_in(rt1, rh1, Γ)
8   then if IS_PATH_FROM_TO_FOR(rt1, rh1, inf) = false
9         then return false.
10        rt1 ← rh1
11   if AGR_LEVEL(r2) >= i and ∃rh2 ∃Γ  subrole_of_in(rt2, rh2, Γ)
12   then if IS_PATH_FROM_TO_FOR(rh2, rt2, inf) = false
13         then return false.
14         rt2 ← rh2
15  i ← i – 1
16  until i > 0 perform steps 2-15.
```

Function AGR_LEVEL(r)

Output: returns the aggregation level number for role r

```
1  l ← 1, rt ← r
2  until ∃rh rh≠ORG and ∃Γ  subrole_of_in(rt, rh, Γ), perform step 3
3  rt ← rh, l ← l + 1
4  return l
```

Function IS_PATH_FROM_TO_FOR(src, dest, inf)

Output: returns true if a communication path exists from role src to role dest for information type inf, or returns false otherwise.

```
1  R ← {src}, RT← ∅
2  R ← R ∪ RT
3  RT ← { r2 | ∃e ∃r1∈R ∃Γ connects_to(e, r1, r2, Γ)
4           and has_onto_mapping(e, inf, inf)}
5  if dest ∈ RT, then return true.
6  until RT ⊄ R, perform 2-5.
7  return false.
```

The general idea of the algorithm is to check if a communication path exists at every aggregation level, through which information is transferred on its way from the role-source to the role-destination. The algorithm begins from the maximum aggregation level of both roles (algorithm 1:1). Then the information flow is reconstructed gradually by proceeding from both ends (the source and the destination) simultaneously (from the source- algorithm 1: 7-10, from the destination- algorithm 1: 11-14) until the point is reached, at which the source-part flows directly into the destination part (algorithm 1: 2-5 the parts connect at one level; 6: the parts connect across two levels). The worst case time complexity of the algorithm is estimated as $O(|LEVEL| \cdot |LINK|^2)$, where |LEVEL| is the number of aggregation levels, and |LINK| is the number of links between roles in the specification.

An instantiated version of this constraint with the template CS4(role1,role2,inf) from the air traffic domain is the following:

CS4(Ground Controller, Safety Investigator, occurrence report): An occurrence report produced by Ground Controller for Safety Investigator should be able to reach Safety Investigator.

Among the generic constraints for the authority structure are:

CS5: Roles that are responsible for a certain aspect related to some process should be necessarily authorized for this.
$\forall$r:ROLE $\forall$a:TASK $\forall$aspect:ASPECT responsible_for(r,aspect,a) $\Rightarrow$ authorized_for(r,aspect,a)

CS6: The relation is_subordinate_of_for: ROLE x ROLE x PROCESS is transitive w.r.t. a process.

CS7: At least one role is authorized for the execution of each task.

In the air traffic domain the following domain-specific constraint is defined:

CS8: The taxiing of an aircraft should be supervised by the controller of a sector, in which the aircraft is situated.

A number of constraints ensuring the consistency between the structures of the view were also identified, among which:

CV1: Roles related by a superior-subordinate relation should interact.
$\forall$r1, r2:ROLE $\forall$a1:PROCESS is_subordinate_of_for(r2,r1,a1) $\Rightarrow$ $\exists$e1, e2:INTERACTION_LINK connects_to(e1,r2,r1,Γ) & connects_to(e2,r1,r2, Γ)

CV2: The role that supervises the execution of some process, should interact with the role performing the process.

An example of a domain-specific constraint over the view is:

CV3: As soon as a runway is vacated and some aircraft is (are) waiting for clearance for this runway, the controller responsible for the runway should provide clearance to one of the aircraft.

Since the organisation-oriented view is related to three other views, constraints may be defined over the specifications of two or more views, using the relations between the views. Explicit identification of such constraints allows to specify and investigate (often latent) dependencies that may exert a significant influence on the organisational functioning and performance. In the following a number of generic and domain-specific constrains over multiple views are given.

Over the process-oriented and organisation-oriented views:

CMV1: For any time point of the taxiing of an aircraft at most one supervising controller is assigned (*domain -specific*).

Over performance-, process- and organisation-oriented views:

CMV2: If a role is committed to a goal, this role should be responsible for some aspect(s) of a task(s) that realizes this goal (*generic*).

Over the process-, organisation- and agent-oriented views:

CMV3: The amount of working hours of each agent from the ag_list should not exceed dr (*domain-specific*)
CMV4: A controller may guide maximum five aircraft at the same time (*domain-specific*)

In the air traffic organisation for safety reasons most of the events observed by human agents and technical systems are registered and stored electronically in form of traces. Using the approach of (Popova and Sharpanskykh, 2007) the specification from the case study was translated into execution constraints, two of which are:

EC1: Taxiing process of aircraft AC20 taxiing2_AC20 should start and finish in the workflow.

EC2: Each incident investigation process should start after a positive decision by either the Safety Investigation Unit (SIU) or the Regulator:
$\forall\gamma$:TRACE $\forall$t:TIME $\forall$p: PROCESS_INVESTIGATION_EX
holds(state(γ, t),process_started(p)) $\Rightarrow$ $\exists$t1:TIME $\exists$r:ROLE t1<t holds(state(γ,t1), decision_taken(initiate_investigation_p,positive) $\wedge$ decision_maker(initiate_investigation_p, r) $\wedge$ (r = SIU $\vee$ r = REGULATOR))

Checking the execution constraints using TTL Checker Tool (Sharpanskykh, 2008) can indicate when the traces do not conform to the specification. In particular, automatic verification of EC2 showed that for some traces it did not hold. Further formal analysis identified that in these traces the incident investigation started based on the decision of OMT. The analysis of the informal organisational behaviour by interviews showed that potential safety-related problems are sometimes reported by controllers to OMT informally, who initiates incident investigation.

5. RELATED LITERATURE

Studies relevant for the focus of this paper have been performed in Enterprise Modelling, Social Science and Artificial Intelligence. Some are discussed here.

The idea of defining rules (often called *business rules*) that determine structural relations and regulate the execution of organisational processes is becoming increasingly popular in the area of Enterprise Modelling. Several classification schemata for business rules have been proposed (Date, 2000; Ross, 2003; von Halle, 2002). Usually rules are classified along the functional dimension, based on how they are used in applications. In (Date, 2000) several rule types are identified, e.g. presentation rules describing the interaction with users, database rules defining operations on databases, logical inference rules allowing inference of truth values of statements. Constraints are often distinguished as a separate category of business rules defining integrity conditions on specifications of business rules. In the mentioned approaches business rules and constraints are defined at a low (machine) level. In contrast, the constraint modelling framework introduced here captures different conceptual aspects related to organisational structures and behaviour (e.g., related to interactions, power, goals). Also the classification framework in (Orriens et al, 2005) distinguishes a number of dimensions that capture some aspects of organisations (e.g., related to goals and tasks). However the concepts are treated at a high abstraction level only and are not elaborated. The framework allows specifying rules across multiple categories, but it does not address consistency and correctness of specifications in a precise manner, as in our framework. The approach of (Lu et al, 2006) focuses on temporal constraints over workflow structures. It provides precise definitions of concepts and relations and describes possibilities for automated analysis. However, in contrast to our framework it does not consider the influence of some key factors (power and interaction) on process execution. In (Goldratt, 1999) constraints are defined as impediments in a system (e.g., production). To identify such constraints dedicated techniques are developed based on cause-effect logics. In our framework inefficiencies may be identified by checking constraints. TTL language is more expressive than cause-effect logics and allows more sophisticated temporal reasoning.

Many enterprise architectures and methodologies allow to capture diverse types of structural relations and dynamics of organisations using dedicated specification languages (e.g.,

ARIS, CIMOSA, TOVE, BPML; see (Bernus, 1998) for an overview) but only simple (or no) verification or validation tools are provided for the analysis of organisational models.

In the theoretical work on institutions (Scott, 2001), a *norm* is a statement that regulates the behaviour and interactions of institutional actors. Some norms form a part of the specification (e.g., define interactions between roles, ordering relations on processes), other are domain-specific constraints by the classification of this paper. Consistency, correctness and integrity are considered as meta-properties on sets of norms. Checking properties is not addressed in the theory of institutions in Social Science due to the lack of formal foundations of the models. Many algorithms for checking consistency of norms for electronic institutions have been proposed in the area of Artificial Intelligence (Esteva et al, 2001). However, electronic institutions are created with the main aim to improve computational properties of distributed algorithms based on agent systems – key structural and behavioural aspects of human organisations are not captured.

The constraint-based approach proposed here differs from constraint satisfaction (Fruhwirth and Abdennadher, 2003). While the focus of the latter is on finding (optimal) solutions given a consistent and stable set of constraints, the proposed approach addresses both design of a specification and constraints that should be satisfied by the specification.

6. CONCLUSIONS

Explicit identification of relations, rules and norms, regulating the structure and behaviour of an organisation, provides better insight in the organisational operation, allows various forms of analysis and facilitates organisational change. All organisational specifications should be checked for internal consistency and validity w.r.t. the domain. To this end, the paper introduces a classification framework for constraints that ensure the consistency and validity of organisational structures and behaviour specified using the framework from (Sharpanskykh, 2008). Constraints are divided into specification and execution constraints. Specification constraints are defined using the expressive languages of the views. The completeness of a set of generic specification constraints depends on the completeness of the set of axioms of the specification language. In general, completeness and validity of domain-specific specification constraints are difficult to ensure. However, some degree of validity and completeness of such constraints may be ensured by involving experienced experts and elaborated analysis of organisational documents. Execution constraints ensure that the actual executions of organisational scenarios correspond to the specifications of formal organisations. Using an approach from (Popova and Sharpanskykh, 2007) a complete set of execution constraints based on a specification can be determined. Checking execution constraints facilitates operations management and decision making by providing information on deviations from the specification identifying recurring events or problems that need to be addressed. If organisational change becomes necessary, the proposed approach allows the requirements for the structure and behaviour of the new organisation to be incorporated as specification constraints early in the design process, ensuring that they will be satisfied by the final specification. Also, expressing requirements as formal constraints helps make assumptions and conflicts explicit and remove ambiguity.

A set of constraints may be inconsistent, i.e., no model exists in which all constraints are true (satisfied). A number of efficient constraint solvers for predicate logic specifications exist (Fruhwirth and Abdennadher, 2003), which allow identifying inconsistencies in a (semi-)automated way.

For checking dynamic constraints related to execution of processes an approach has been developed, which differs from standard state-based approaches. For each process the bounds of its execution interval are calculated. Then verification of constraints is performed based on the obtained temporal intervals. This type of verification is computationally much cheaper (has polynomial time complexity) than the general-purpose state-based analysis with exponential time complexity.

Future research includes the application of the developed framework for large-scale case studies.

REFERENCES

Bernus, P. et al. (eds.) (1998). *Handbook on Architectures of Information Systems*, Springer-Verlag, Heidelberg.

Date, C. (2000). *What Not How: The Business Rule Approach to Application Development*, Addison-Wesley Longman.

Esteva, M., Rodriguez-Aguilar, J. A., Sierra, C., Garcia, P., and Arcos, J. L. (2001). On the Formal Specification of Electronic Institutions. In *Agent-mediated Electronic Commerce: the European AgentLink Perspective, LNAI 1991*, Springer-Verlag, 126-147.

Fruhwirth, T. and Abdennadher, S. (2003) Essentials of Constraint Programming, Springer Verlag.

Goldratt, E. (1999) *Theory of Constraints*. Great Barrington, MA, North River Press.

Lu, R., Sadiq, S., Padmanabhan, V., and Governatori, G. (2006). Using a temporal constraint network for business process execution. In *Proc of the 17th Australasian Database Conference*, pp.157-166.

Orriens, B., Yang, J., and Papazoglou, M. (2005). A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaboration. In *Proc. of the 3rd Intl Conference on Service Oriented Computing*.

Popova, V., and Sharpanskykh, A. (2007). Formal Analysis of Executions of Organizational Scenarios Based on Process-Oriented Models. In *Proc. of 21st European Conference on Modelling and Simulation*, 36-44.

Ross, R. (2003). *Principles of the Business Rule Approach*, Addison-Wesley.

Scott, W.R. (2001). *Institutions and organisations*. SAGE Publications, Thousand Oaks.

Sharpanskykh, A. (2008). *On Computer-Aided Methods for Modeling and Analysis of Organisations*, PhD thesis, Vrije Universiteit Amsterdam.

von Halle, B. (2002). *Business Rules Applied: Building Better Systems Using the Business Rule Approach*. John Wiley & Sons Ltd.