VU Research Portal

Constraint-based Modelling of Organisations

Viara, Popova; Sharpanskykh, A.

2008

document version
Publisher's PDF, also known as Version of record

Link to publication in VU Research Portal

citation for published version (APA)
Viara, P., & Sharpanskykh, A. (2008). *Constraint-based Modelling of Organisations*.

Constraint-based Modelling of Organisations

Technical Report AI91008

Viara Popova[1,2] and Alexei Sharpanskykh[2]

[1] De Montfort University, Centre for Manufacturing, Leicester, UK
[2] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

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 correctness of organisational specifications. To verify the satisfaction of constraints, efficient and scalable algorithms have been developed and implemented. The application of the proposed approach is illustrated by a case study from the air traffic domain.

1 Introduction

Organisations that exist in modern society and nature are characterised by a great variety of structures and dynamics. The complexity of an organisation is interdependent on the complexity of the environment, in which it is situated [16]. To achieve high productivity, internal structures of an organisation should be organised effectively and efficiently, so that the fit with the environment is achieved. Due to a high structural and behavioural complexity of many modern organisations, this goal is difficult to achieve, to a great extent because of mistakes and performance bottlenecks. With the growth of complexity, the possibility of rapid and reliable analysis of organisations becomes increasingly important. Manual analysis of specifications 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 many automated processes (e.g., computer integrated manufacturing [2]), and for inter-enterprise cooperation and organisational change.

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 [7]. In particular, in the framework from [11] used for defining the specifications in this paper four interrelated views with their dedicated predicate logic-based languages are defined: performance-oriented, process-oriented, organisation-oriented and agent-oriented. The internal consistency and validity of organisational specifications can be ensured by defining a set of constraints that should be satisfied by them. An approach for modelling and verification of constraints is the main contribution of this paper. *A constraint* is a restriction imposed on some aspect(s) of the organisational structure and/or behaviour. Syntactically, a constraint is an expression over the language of a view(s). 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. Any specification of the view should satisfy such constraints. Other constraints reflect domain-specific regulations of a particular organisation(s) and are required to be satisfied by the specifications of this (these) organisation(s). The paper proposes a classification framework for constraints.

To ensure satisfaction of constraints by organisational specifications, efficient verification techniques are required. Formally, a set of constraints is represented by a *logical theory* that consists of formulae constructed from the terms of the languages of the views. A specification is *correct* if this theory is satisfied by the specification - all sentences in the theory are true in the logical structure(s) corresponding to all possible executions of the specification. The algorithms for verification of correctness of specifications w.r.t. different types of constraints were developed and implemented.

The correctness of a specification does not guarantee its flawless execution by agents performing organisational roles. Currently many organisations use information systems controlling the execution of their processes. Such systems may also be used to check the correspondence of actual executions of organisational scenarios to the organisational specification (e.g., by the method from [13]). To enable such verification, a special class of constraints is required, which is also addressed here. Diverging behaviour of agents may have positive or negative influence on organisational performance. To identify such influences, automated analysis methods are required, which are considered in this paper. Based on the results of such analysis, changes in the constraints and/or the specification of an organisation may follow.

The paper is organized as follows. Section 2 briefly describes the framework for organisation modelling. Section 3 introduces the classification framework for constraints. Examples of constraints and checking algorithms are given in Section 4. Methodological guidelines for redesign of constraints are described in Section 5. Section 6 discusses the related literature. Section 7 concludes the paper.

2 The Organisation Modelling Framework

In this Section the general organisation modelling framework is briefly described via the four interrelated views. 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) [20], 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 views of the framework and of different types of constraints, a case study from the air traffic management domain is used. In this case [19] an Air Traffic Control Organisation (ATCO) has been modelled and analysed. An ATCO ensures a safe and efficient flow of aircrafts both at airports and in the air. Nowadays, an ATCO represents a complex organisation that involves many parties with diverse goals (e.g., airports, air navigation service providers (ANSP), airlines, regulators, and the government) performing a wide range of tasks. In this paper two tasks performed by the ATCO are considered: movement of an aircraft on the ground, and incident reporting and investigation. The ground movement task includes the taxiing of an aircraft to the designated runway and the subsequent take off from this runway. During taxiing an aircraft moves from one sector of the airport to another. The monitoring and the traffic control in a sector or runway are performed by a dedicated air traffic controller from the ANSP. During taxiing the control over an aircraft is handed over from one controller to another, depending on the physical position of the aircraft. Before crossing an active runway the crew of an aircraft should request the controller responsible for the runway for clearance. Only when clearance is provided, the aircraft is allowed to cross. In case an incident/accident occurs during taxiing or take off, this should be reported and investigated.

The ***process-oriented view*** of the framework contains information about the organisational functions, how they are related, ordered and synchronized and the resources they use or produce. *Tasks* represent organisational functions characterized by name, maximal and minimal duration. Tasks can be decomposed into more specific ones using AND- and OR-relations, forming hierarchies, e.g., the task "Taxiing the aircraft to the designated runway" has a subtask "Inquiry for the clearance for crossing an active runway" with minimal duration 2 seconds and maximal duration 5 seconds.

A *workflow* is defined by a set of (partially) temporally ordered *processes*. A process uses a task as a template and inherits all characteristics of the task. Decisions are treated as processes associated with decision variables. The order of process execution is defined by sequencing, branching, synchronization and cycle relations which can express the most commonly used workflow templates. Figure 1 gives an example of the workflow for formal incident reporting initiated by a crew. The duration of each process in a workflow may vary in actual executions and each process may have different starting points in different executions. The earliest and the latest starting time point for each process p can be identified (denoted as est_p and lst_p).

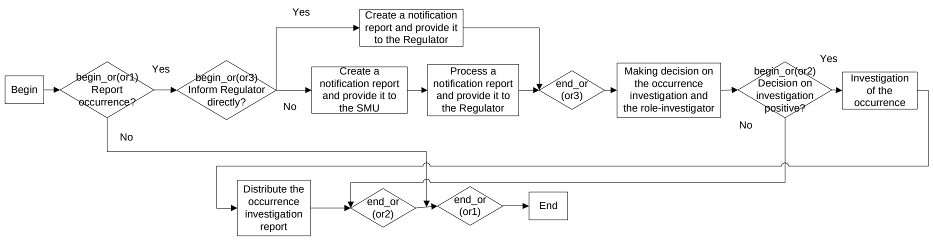
Begin
begin_or(or1) Report occurrence?
Yes
No
begin_or(or3) Inform Regulator directly?
Yes
No
Create a notification report and provide it to the Regulator
Create a notification report and provide it to the SMU
Process a notification report and provide it to the Regulator
end_or (or3)
Making decision on the occurrence investigation and the role-investigator
begin_or(or2) Decision on investigation positive?
Yes
No
Investigation of the occurrence
Distribute the occurrence investigation report
end_or (or2)
end_or (or1)
End

Fig. 1. The workflow defining the execution of the incident reporting task initiated by a crew.

The value of est_p (lst_p) is calculated under the assumption that all processes influencing the starting point of p have the minimum (resp. maximum) duration. The earliest (latest) ending time of a process p (eet_p (let_p)) is calculated as est_p + p.min_duration (lst_p + p.max_duration). More details on the calculation are given in [12].

Tasks use, consume and produce resources of different types describing tools, supplies, components or data. Among the characteristics of resource types are location and expiration_duration (the time duration for which a resource type can be used). Resources are instances of resource types, inherit their characteristics and have, in addition, name and amount. Some resources can be shared by processes (e.g., storage facilities, transportation vehicles). The process-oriented view is discussed in [12, 13].

Central notions in the ***performance-oriented view*** are *goal* and *performance indicator (PI)*. A PI is a quantitative or qualitative indicator reflecting the state/ progress of a company, unit or individual. PIs can be hard, i.e., quantitative (e.g., PI1 "time for incident investigation") or soft, i.e., not directly measurable (e.g., PI2 "level of quality of incident investigation"). PIs can be connected by various relationships, such as (strongly) positive/negative causal influence of a PI on another, positive/negative correlation between PIs, aggregation (PIs express the same measure at different aggregation levels). For example, PI1 and PI2 are related by a negative causality relation meaning that positive change of one PI causes negative change of the other.

Goals are objectives that describe a desired state or development and are defined as expressions over PIs. The characteristics of a goal include, among others: priority; horizon – for which time point/interval should the goal be satisfied; hardness – hard or soft (for which instead of satisfaction, degrees of *satisficing* are defined). Based on the PIs defined above the following goals can be formulated: G1: "It is required to achieve high level of quality of the investigation of an incident", G2: "It is required to minimize the investigation time for an incident". A goal can be refined in sub-goals forming a hierarchy. Information about the satisfaction of lower-level goals is propagated to determine the satisfaction of high-level goals. Detailed description of the view can be found in [11].

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.

Each role has an input and an output interface, which facilitate the interaction (in particular, communication) with other roles and the environment. Role interfaces are described in terms of interaction (input and output) ontologies: a signature specified in order-sorted logic. Generally speaking, 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 the aggregation level 1 of the air traffic organisation from the case study are depicted in Figure 2a. Figure 2b zooms into role Air Navigator Service Provider and presents its subroles at aggregation level 2.

For more details on interaction relations in organisations we refer to [8].

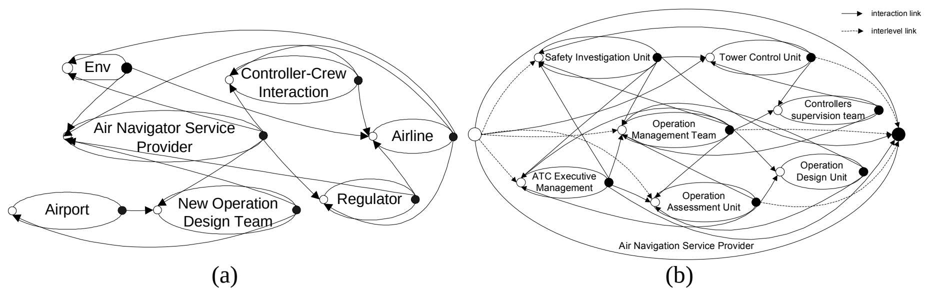
Env
Controller-Crew Interaction
Air Navigator Service Provider
Airline
Airport
New Operation Design Team
Regulator
(a)
interaction link
interlevel link
Safety Investigation Unit
Tower Control Unit
Controllers supervision team
Operation Management Team
ATC Executive Management
Operation Design Unit
Operation Assessment Unit
Air Navigation Service Provider
(b)

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

Besides interaction relations, 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 or rights w.r.t. specific aspects of tasks: e.g., monitoring, execution, consulting, making decisions. Roles with managerial rights may authorize or make other roles responsible for some aspects of task execution. For instance, for the task "Investigation of occurrence based on the notification report": the Safety Investigator is responsible for execution and technological decisions and the Head Safety Investigation Unit - for monitoring, consulting and managerial decisions. Details on authority relations are given in [17].

In the ***agent-oriented view*** an agent is defined as an autonomous entity able to interact (e.g., by observations and actions) with other agents and the environment. An agent is characterised by a set of capabilities (knowledge and skills) and personal traits. In general, an agent can be allocated to a role if s/he possesses the necessary capabilities and traits required for the role. Depending on the type of the organisation, allocation of agents to roles may be prescribed by a specification or may be determined at runtime. The framework allows representing both intentional (e.g., goals) and motivational aspects of agent behaviour. Other internal states of agents are represented as beliefs. Using TTL any particular type of agent behaviour can be specified. More detailed description of the agent-oriented view can be found in [18].

3 The Classification Framework for Constraints

The role of constraints may differ in organisation modelling which influences their format, purpose and way of use. In this Section 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.

Two main groups of constraints were identified: specification constraints and execution constraints. ***Specification constraints*** are embedded in the specification of an organisation and represent statements that must be true for the current specification. They are expressed as formulae that are constructed from terms of the formal languages of one or more views using Boolean connectives and quantifiers. Specification constraints can be checked at every step of the (re-)design process in order to ensure the correctness of the current specification. They can be classified in two dimensions based on their scope and on their origin. Based on their scope, specification constraints are divided into:

- *Constraints within one structure* to ensure its consistency and validity: Several structures can be defined in the specification, e.g.: goal and performance indicators structures in the performance-oriented view, workflow, task and resource structures in the process-oriented view, 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, matching of durations, uniqueness of an object in a structure, etc. Examples of hierarchical structures are the goals, task, resource and role decomposition structures.
- *Constraints within one view*: The constraints of this type are expressed using the language of the corresponding view. Some take care of the consistency between related structures within the view such as: consistency between the goals and the performance indicators structures, between the task structures and the workflow, etc. Note that constraints on the agent-oriented view alone cannot be formulated due to the descriptive character of this view unlike the other three views which have prescriptive character. However it is possible to define constraints of the remaining two types.
- *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 organisation*: These constraints ensure that organisations related by some kind of contract/cooperation are aligned so that successful interaction can be achieved. Such constraints are very specific and depend on the types of the involved organisations and on the contract/cooperation between them. An example is a supply chain that needs to coordinate its operations by making sure the involved parties have the necessary goals, processes, resources, etc.

Based on their origin, the 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 constraints* used to ensure correctness of the structure and views; *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).

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 such as the society or government and contain rules about working hours, safety procedures, emissions, etc. Sources for such constraints can be laws, 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. This is

in contrast to the generic physical constraints which should be satisfied by any specification irrespective of the application domain.

To reduce 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 constraints, examples of which are given in the following Section 4. The rest of interlevel consistency constraints 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 [20] is used.

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. Every specification can be used to formulate such constraints. The set of execution constraints needs to be sufficiently complete in order to provide meaningful results, i.e. a guarantee that the actual execution of an organisational scenario is correct w.r.t. the specification. Execution constraints are defined using a dedicated formal language for expressing execution traces from [13]. Aspects monitored and recorded in traces include started/finished processes, produced/used resources, measured values of performance indicators, roles assigned to and agents performing processes, 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 more that one organisation 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 play a role of shortcuts for complex logical expressions, and can be easily used by non-professionals in logics. Such templates can be selected and customized by the designer by assigning specific values to the parameters of the template. Examples of such templates are given in Section 4.

4 Examples of Constraints

In this Section, examples of constraints are given following the classification from Section 3. All constraints considered in this paper can be formalized using the dedicated languages of the views and TTL. Due to the length limitations, formal representation is only provided for some constraints.

4.1 Constraints within one structure or within one view

Within the ***process-oriented view*** three types of structures are defined: a task structure, a resource structure and a flow of control (or workflow structure).

A task structure is a hierarchy and to ensure its consistency several constraints have been identified, among which is this interlevel consistency constraint:

CS1: For every and-decomposition of a task, the minimal duration of the task is at least the maximal of all minimal durations of its subtasks.

The following constraint is required for the consistency of a resource structure:

CS2: If data type dt2 is a functional part of data type dt1, then the expiration duration of dt1 is at most the expiration duration of dt2.

Structural constraints for a workflow are defined as the workflow correctness properties in [12]. The following generic constraints are expressed over multiple structures of the process-oriented view:

CV1: If a task produces certain resource type as output then there exists at least one subtask in at least one and-decomposition of this task that produces this resource type (*interlevel consistency constraint*)

CV2: For every process that uses certain amount of a resource of some type as input, without consuming it, either at least that amount of resource of this type is available or can be shared with another process at every time point during the possible execution of the process.

CV3: Non-sharable resources cannot be used by more than one task at the same time.

An example of the domain-specific process-oriented constraint from the case:

CV4: A clearance to cross/takeoff is provided to some aircraft becomes invalid either when the aircraft has crossed the runways 2 minutes after the time point of its generation.

In the ***performance-oriented view*** constraints are defined for performance indicators and goal structures. To ensure the consistency of a PI structure the following constraints are specified:

CS3: PIs related by an aggregation relation should have the same unit of measurement.
∀pi1, pi2:PI aggregation_of(pi1, pi2) ⇒ pi1.unit = pi2.unit

CS4: Aggregation is an antisymmetric relation.
∀pi1, pi2:PI aggregation_of(pi1, pi2) ⇒ ¬aggregation_of(pi2, pi1)

CS5: The causality relation is transitive.
∀pi1, pi2, pi3:PI, s1, s2, s3:EFFECT causing(pi1, pi2, s1) & causing(pi2, pi3, s2) ⇒ causing(pi1, pi3, s3)

Consider several constraints ensuring the interlevel consistency of a goal structure:

CS6: A goal cannot be a subgoal of itself.

CS7: In each refinement of a soft goal, should be at least one subgoal that satisfices this goal.

The following constraints are expressed over both PI and goal structures:

CV5: If goals are related by a refinement, then the PIs on which these goals are based should be related by some kind of a causality relation.
∀ g1, g2: GOAL, ∀ l: GOAL_LIST ∀ gp1, gp2: GOAL_PATTERN ∀pi1, pi2: PI
is_in_goal_list(g1, l) & is_refined_to(g2, l) & is_based_on(gp1, pi1) & is_formulated_over(g1, gp1) & is_based_on(gp2, pi2) & is_formulated_over(g2, gp2) ⇒ ∃pn: EFFECT causing(pi1, pi2, pn)
(EFFECT = {very_negative, negative, positive, very_positive})

CV6: The hardness of a goal and the PI, on which this goal is based, should be the same.

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 constraints:

CS8: No role can be a subrole of itself at any aggregation level.

CS9: A role can be a subrole of one role at most.

In the structure Γ ∀r, r1, r2: ROLE subrole_of_in(r, r1, Γ) & subrole_of_in(r, r2, Γ) ⇒ r2=r1

CS10: Each subrole of a composite role r should interact with at least one other subrole of r.

In the structure Γ ∀r1: ROLE subrole_of_in(r1, r, Γ) ⇒ ∃r2:ROLE ∃e:INTERACTION_LINK subrole_of_in(r2, r, Γ) & (connects_to(e, r2, r1, Γ) | connects_to(e, r1, r2, Γ))

CS11: Information provided to the input of a composite role should be further transmitted to one or more its subroles.

CS12: Any subrole of a composite role is not allowed to interact directly with any other role outside of this composite role.

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

CS13: Particular information should be always transferred between some roles.

An instantiated version of this constraint with the corresponding template CS13(role1, role2, information_type) is applied in the air traffic domain:

CS13(first_pilot, second_pilot, controller_instructions): The pilots of a crew should always share with each other information about the controllers' instructions.

CS14: An interaction path should exist between two particular roles.

For checking this constraint a specification of role interaction I is represented as a directed graph G = (V, E), in which each vertex $v \in V$ corresponds to a role from I and each edge $e \in E$ with the initial vertex representing role r1 from I and the terminal vertex representing role r2 from I corresponds to either interaction link e specified by connects_to(e, r1, r2, I) or to interlevel link il specified by interlevel_connection(il, r1, r2, I). Then, using the simple algorithms for establishing the existence of a path between two vertices from the graph theory the satisfaction of the constraint is established. The time complexity of such a transformation procedure is linear in the size of I.

An instantiated version of this constraint with the template CS14(role1, role2) from the air traffic domain is the following:

CS14(Ground Controller, Safety Investigator): An interaction path should exist between Ground Controller and Safety Investigator roles.

Among the generic constraints for the authority structure of an organisation are:

CS15: Roles that are responsible for a certain aspect related to some process should be necessarily authorized for this.

∀r ROLE ∀a:TASK ∀aspect:ASPECT responsible_for(r,aspect,a) ⇒ authorized_for(r,aspect,a)

CS16: The relation is_subordinate_of_for: ROLE x ROLE x PROCESS is transitive

CS17: Only roles that have the responsibility to make managerial decision w.r.t. some process are allowed to authorize other roles for some aspect of this process.

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

CS18: The taxiing of an aircraft should be supervised by the controller of a sector, in which the aircraft is situated.

A number of constraints that ensure the consistency between the structures of the organisation-oriented view have been identified, among which:

CV7: Roles related by a superior-subordinate relation should interact.

∀r1, r2:ROLE ∀a1:PROCESS is_subordinate_of_for(r2, r1, a1) ⇒ ∃e1, e2:INTERACTION_LINK connects_to(e1, r2, r1, Γ) & connects_to(e2, r1, r2, Γ)

CV8: The role that supervises the execution of some process, should interact with the role performing the process.

An example of the domain-specific constraint over the view from the case study is:

CV9: 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.

4.2 Constraints over multiple views

Since the views are interrelated, also constraints may be expressed over combined specifications of two or more views, using the relations defined 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 identified.

Over the performance-oriented and process-oriented views:

CMV1: Each task should contribute to the satisfaction of one or more goals (*generic*).

CMV2: Each goal should be realized by execution of one or more tasks (*generic*).

Over the process-oriented and organisation-oriented views:

CMV3: At the beginning of each process for each of the basic aspects for this process (execution, tech_des, and manage_des) a responsible role should be assigned (*generic*).

CMV4: For any time point of the taxiing of an aircraft at most one supervising controller is assigned (*domain -specific*).

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

CMV5: 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-oriented, organisation-oriented and agent-oriented views:

CMV6(ag_list, dr): The amount of working hours of each agent from the ag_list should not exceed dr (*domain-specific*)

This constraint is instantiated for the air traffic domain:

CMV6(CONTROLLER, 5): Each controller should work not more that 5 hours per day.

Another constraint for the air traffic organisation:

CMV7: After each work session of a controller there is 1 hour break before the next session (*domain-specific*)

CMV8: A controller may guide maximum five aircraft at the same time (*domain-specific*)

The taxiing of an aircraft is supervised by the Aircraft Controller role. Generally, the agent assignment to this role changes, when the aircraft moves to another airport's sector. In some cases a re-assignment may occur in the same sector. All cases of assignment change should be taken into account, when constraint CMV8 is checked. Consider the algorithm for checking constraint CMV8 for an agent-controller ag1:

1. In the organisational specification identify the set of roles of type AIRCRAFT_CONTROLLER, to which agent ag1 is allocated: ROLES_REL = {r: AIRCRAFT_CONTROLLER | agent_plays_role(ag1, r)}
2. Identify the set of taxiing processes supervised by the roles from ROLES_REL: PROC_REL={p:PROCESS| $\exists r \in$ ROLES_REL is_responsible_for(r, supervision, p)}
3. For each process $p \in$ PROC_REL identify the execution interval $[est_p, let_p]$ and write the values est_p and let_p in a new row of the allocation matrix M of ag1. If an agent has an allocation for a part of the execution interval of the process performed by a role, then the time points of the beginning and end of this allocation should be written in M.
4. Process the obtained allocation matrix M row by row. For each row identify the existence of non-empty intersections with intervals represented by other rows of M. An intersection of the intervals represented by rows i and j is nonempty if $\neg((m_{i2} < m_{j1}) \vee (m_{j2} < m_{i1}))$ is true. When a row is processed, it is not taken into account in any other evaluations. If for a row the amount of non-empty intersections is greater than 4, then **CMV8 is not satisfied**, **exit**.
5. When all rows are processed, **CMV8 is satisfied.**

The presented algorithm proceeds under the assumption that any organisational process p may be executed at any time point during the interval $[est_p, let_p]$. Thus, the satisfaction of constraint CMV8 is checked for all possible executions (combinations of intervals) of the processes allocated to an agent. To this end, instead of the calculation of combinations of processes at each time step (as in state-based methods [3]), the algorithm establishes the existence of non-empty intersections of the complete execution intervals of processes in a more efficient way. As shown in [12] the time complexity for the calculation of the execution bounds for all processes of a workflow for the worst case is not greater than $O(|P|^2 Cw)$, where P is the set of processes of the workflow, and Cw is the set of constraints on the workflow. When the execution bounds for processes are known, the time complexity of the algorithm for CMV8 is estimated as follows: the execution of the steps 1-3 takes O(|C|), where C is the set of constraints of an organisation; the execution of the step 4 requires $O(|C|^2)$; thus the time complexity of the complete algorithm is $O(|C|^2)$, which is much better than the exponential time complexity of state-based algorithms.

4.3 Execution Constraints

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 [13] the specification from the case study was translated into execution constraints, some of which are:

EC1: Taxiing process of aircraft AC20 taxiing2_AC20 should start and finish in the workflow.
∃t1 holds(state(γ, t1), process_started(taxiing2_AC20)) ⇒
∃t2 holds(state(γ, t2), process_finished(taxiing2_AC20)

EC2: The duration of crossing process of aircraft AC20 crossing1_AC20 should be as specified by the formal organisation:
∃t1, t2 holds(state(γ, t1), process_started(crossing1_AC20)) &
holds(state(γ, t2), process_finished(crossing1_AC20))
⇒ crossing1_AC20.min_duration ≤ t2-t1 & t2-t1 ≤ crossing1_AC20.max_duration

EC3: The delay between the time point, when the clearance to cross is provided to aircraft taxiing2_AC20 and the time point when the actual crossing starts should be less than 2 seconds

EC4: Active_runway1 is allowed to be used by one process at most at any time point:
∀L:PROCESS_LIST_EX ∀p1:PROCESS_EX ∀t: TIME holds(state(γ, t), [resource_used_by(active_runway1, L) ∧ is_in_list(p1, L)] ⇒ ¬∃p2 p2 ≠ p1 is_in_list(p2, L)])

EC5: Each instruction of controller CONTR_A guiding aircraft AC20 should be read back by either PILOT_A or by PILOT_B of the crew of AC20 within 5 minutes.

EC6: When aircraft AC20 has moved from sector_A to sector_B, the responsibility over AC20 should be transferred from controller CONTR_A to controller CONTR_B within 1 minute.

5 Methodological Guidelines for Redesign of Constraints

This Section considers the problem of organisational change and redesign from the point of view of formulating constraints. Organisational change can be necessary for various reasons internal (e.g., the performance decrease) or external which are out of the scope of this paper. Often the analysis leading to the decision to change starts by observing the behaviour of the organisation in its environment, i.e. its actual execution traces. Checking the execution constraints can give insights when it is observed that the traces do not conform to the specification. However, even when the traces are correct, analyzing them is important. Execution traces can be very long and automated support can greatly reduce the effort. TTL and the dedicated execution language allow formulating properties that can be checked automatically over one or more traces at once. These properties can, for example, express hypotheses of the analyst over recurrent patterns of behaviour, bottlenecks or other relations between events in the traces and how they are linked to the observed performance of the organisation. The results of such an analysis may have high significance (i.e., safety-related issues in safety-critical organisations) and require a change of the organisational structure and/or behaviour. Furthermore, sometimes informal patterns of agent behaviour may be observed regularly. If such patterns do not create a conflict with the major organisational goals and contribute positively to the satisfaction of some goals of agents, the organisation may consider institutionalizing them. The type of change required is based on the identified relationships, which do not form a part of the current organisation and which may play the role of constraints over the new, future organisation that should result from the change process. Since the identified properties are expressed in the execution language, a translation into the format of a formal organisation is required. Since both formats are closely related, such a translation can easily be automated although sometimes a direct translation from properties to constraints might not be enough and reformulation might be necessary.

Furthermore, together with the constraints the designer might start changing the specification by adding/deleting structures, relations or objects. Constraints that have become irrelevant should be removed or reformulated. The resulting new set of constraints can be checked over the new specification at every step of the design process, if necessary, in order to ensure consistency.

The described approach has been applied to the case study. The formal air traffic organisation describes the incident reporting path, initiated by a controller and/or a crew, who creates a report based on an observed incident. Based on such report(s), the decision about the investigation necessity is made by the Safety Investigation Unit (SIU) role of ANSP, and in the case of a crew initiation - by the Regulator role. Thus, the following property can be checked on actual execution traces to ensure that each incident investigation is after a positive decision by either the SIU or the Regulator:

∀γ:TRACE ∀t:TIME ∀p: PROCESS_INVESTIGATION_EX holds(state(γ, t),process_started(p)) ⇒
∃t1:TIME ∃r:ROLE t1< t holds(state(γ, t1), decision_taken(initiate_investigation_p, positive) ∧ decision_maker(initiate_investigation_p, r) ∧ (r = SIU ∨ r = REGULATOR))

The automatic verification of this property using the checking tool [13] showed that for some traces it did not hold. The further analysis identified that in some cases the incident investigation started based on the positive decision of the Operation Management Team (OMT). The informal aspects of the organisational dynamics are not registered electronically and were made explicit by performing interviews. The analysis of the informal behaviour of the organisation showed that potential safety-related problems have sometimes been reported by controllers based on their informal discussions during breaks. Information about these problems was propagated along the informal incident reporting path to the OMT, who made the decision to initiate incident investigation. The analysis based on the empirical data and stochastic agent-based simulations showed that the informal incident reporting always resulted in faster identification of safety-related problems than the formal incident reporting [19]. At least two reasons can be identified that may explain this finding: (1) the informal incident reporting path is shorter and does not require inter-organisational cooperation that may involve delays; (2) the identification of a problem in the informal incident reporting case is based on the combined knowledge and experience of the controllers involved in the discussion, which may expedite the problem identification. Although evident patterns of interaction can be identified in the informal incident reporting, still it occurs sporadically, without proper organisation. Since the safety-related issues have first priority in air traffic management, the organisation may consider institutionalizing (or at least providing a certain structure to) this process.

6 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. In [22] some advantages of a rule-based approach are identified, e.g. externalization of rules, clarity and traceability of rules, possibility of rapid change. Several classification schemata for business rules have been proposed [4, 14, 22]. Usually rules are classified along the functional dimension, based on how they are used in applications. In [4] 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 and do not capture different conceptual aspects related to organisational structures and behaviour (e.g., related to interactions, power, goals). In contrast, the classification framework introduced in [10] distinguishes a number of dimensions that capture some aspects of organisations (e.g., related to goals and to task execution). However the concepts are treated at a high abstraction level only and are not elaborated (e.g., by giving precise definitions, introducing relations and structures). The framework allows specifying rules across multiple categories, but it does not address consistency and correctness of such specifications in a precise manner.

The approach presented in [9] focuses on temporal constraints defined over workflow structures. Although the approach provides precise definitions of concepts and relations, and describes possibilities for automated analysis, it does not consider influence of different important organisational factors (power structure, interaction relations) on execution of processes.

Many enterprise architectures and methodologies provide means to capture diverse types of structural relations and dynamics of organisations using dedicated specification languages (e.g., ARIS [15], CIMOSA [2], TOVE [6], BPML [1]) but only simple (or no) verification or validation tools are provided for the analysis of organisational models.

In the theoretical work on institutions [16] the notion of a *norm* plays an important role. Norms are statements that regulate behaviour of institutional actors and the interactions between them. Some norms form a part of an organisational specification (e.g., define interactions between roles, ordering relations on processes), while other may be considered as 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 absence of formal foundations of institutional models. Many algorithms for checking consistency of norms have been proposed in the area of Artificial Intelligence [5] for electronic institutions. However, electronic institutions are created with the primary aim to improve computational properties of distributed algorithms based on agent systems - many key structural and behavioural aspects of human organisations are not captured by the models of artificial institutions.

The constraint-based approach considered in this paper differs from another technique from the area of Artificial Intelligence - constraint satisfaction [21]. 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. The designer can vary both the specification and the set of constraints, supported by the automated analysis tool.

7 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 correctness of organisational structures and behaviour specified using the framework from [11]. Constraints are divided into specification and execution constraints. Specification constraints are defined using the expressive languages of the views over all key organisational aspects (performance, power, interaction, etc.). Execution constraints ensure that the actual executions of organisational scenarios correspond to the specifications of formal organisations.

Often, for an organisation to achieve its primary goals in changing environmental conditions, it needs to change its structure and behaviour, which usually results in a change of the set of constraints. Methodological guidelines for redesign of constraints are discussed in this paper. Different types of constraints and the application of the proposed guidelines are illustrated using the case study from the air traffic domain.

To ensure satisfaction of constraints, efficient algorithms should be developed that can also be automated. Most of the static structures of the views can easily be translated into a graph representation, on which structural constraints can be checked using efficient algorithms from graph theory. For checking dynamic constraints related to processes execution, an approach has been developed, which differs from standard state-based approaches. For each process the bounds of its execution interval (i.e., the earliest starting and the latest ending time points) are calculated. Then verification of constraints is performed based on operations on the obtained temporal intervals. This type of verification is computationally much cheaper (has polynomial time complexity) than the general-purpose state-based analysis (e.g., by model checking [3], which has exponential time complexity).

The developed approach allows scalability by compositional modelling and analysis. Depending on the purpose and level of details of analysis, constraints may be specified at different aggregation levels. To ensure consistency between constraints of different levels and to guarantee integrity of an organisational specification, a set of interlevel consistency constraints is introduced, which is also addressed in the paper.

The proposed framework for specification of constraints together with the developed scalable and efficient analysis techniques provide useful means for improvement of performance and flexibility of modern organisations that operate in a competitive, constantly changing environment.

References

1. Business Process Modeling Language (BPML).http://www.bpmi.org.
2. CIMOSA – Open System Architecture for CIM; ESPRIT Consortium AMICE, Springer-Verlag, Berlin (1993)
3. Clarke, E.M., Grumberg, O., and Peled, D.A.: Model Checking. MIT Press (2000)
4. Date, C.: What Not How: The Business Rule Approach to Application Development, Addison-Wesley Longman Inc. (2000)
5. Esteva, M., Rodriguez-Aguilar, J. A., Sierra, C., Garcia, P., Arcos, J. L.: On the Formal Specification of Electronic Institutions. In the book Agent-mediated Electronic Commerce: the European AgentLink Perspective, LNAI 1991, Springer-Verlag, (2001) 126-147
6. Fox, M, Barbuceanu M, Gruninger, M, Lin, J.: An Organization Ontology for Enterprise Modelling. In: M. Prietula, K. Carley and L. Gasser (eds.), Simulating Organizations: Computational Models of Institutions and Groups, AAAI/MIT Press, 131-152 (1997)
7. GERAM: The Generalized Enterprise Reference Architecture and Methodology. IFIP-IFAC Task Force on Architectures for Enterprise Integration, In: Bernus P, Nemes L, Schmidt G. (eds): Handbook on Enterprise Architectures, Springer-Verlag, 21-63 (2003)
8. Jonker, C.M., Sharpanskykh, A., Treur, J., Yolum, P.: A Framework for Formal Modeling and Analysis of Organizations, Applied Intelligence, 27(1), 49-66 (2007)
9. Lu, R., Sadiq, S., Padmanabhan, V., Governatori, G.: Using a temporal constraint network for business process execution. In Proceedings of the 17th Australasian Database Conference, pp.157-166 (2006)
10. Orriens, B., Yang, J., Papazoglou, M. A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaboration. In Proceedings of the 3rd International Conference on Service Oriented Computing (2005)
11. Popova, V., Sharpanskykh, A.: Formal Modelling of Goals in Agent Organizations. In: V. Dignum, F. Dignum, E. Matson, B. Edmonds (eds.), Proceedings of the Workshop on Agent Organizations: Models, and Simulation at IJCAI'07, 74-86 (2007)
12. Popova, V., Sharpanskykh, A.: Process-Oriented Organization Modeling and Analysis. In: Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, INSTICC Press, pp. 114-126 (2007)
13. Popova, V., Sharpanskykh, A.: Formal Analysis Of Executions Of Organizational Scenarios Based On Process-Oriented Models. In: I. Zelinka et al. (eds.), Proceedings of 21st European Conference on Modelling and Simulation, SCS Press, 36-44 (2007)
14. Ross, R.: Principles of the Business Rule Approach, Addison-Wesley (2003)
15. Scheer, A-W., Nuettgens, M. ARIS Architecture and Reference Models for Business Process Management. LNCS 1806, Berlin et al. 366-389 (2000)
16. Scott, W.R.: Institutions and organisations. SAGE Publications, Thousand Oaks (2001)
17. Sharpanskykh, A.: Authority and its Implementation in Enterprise Information Systems. In: Proceeding of the 1st International Workshop on Management of Enterprise Information Systems, MEIS 2007, INSTICC Press, 33-43 (2007)
18. Sharpanskykh, A.: Modeling of Agents in Organizational Context. In: H-D. Burkhard, G. Lindeman, L. Varga, R. Verbrugge (eds.), Proceedings of the 5th International Central and Eastern European Conference on Multi-Agent Systems, LNAI 4696, Springer Verlag (2007)
19. Sharpanskykh, A.: Modelling and Analysis of Air Traffic Organizations. Technical Report 072510AI, Vrije Universiteit Amsterdam, http://www.few.vu.nl/~sharp/tr072510AI.pdf
20. Sharpanskykh, A., Treur, J.: Verifying Interlevel Relations within Multi-Agent Systems. In: Brewka, G. et al. (eds.), Proceedings of the 17th European Conference on Artificial Intelligence, IOS Press, 290-294 (2006)
21. Tsang, E.: Foundations of Constraint Satisfaction, Academic Press (1993).
22. von Halle, B.: Business Rules Applied: Building Better Systems Using the Business Rule Approach. John Wiley & Sons Ltd (2002)