Redesign of Organizations as a Basis for Organizational Change

Mark Hoogendoorn[1], Catholijn M. Jonker[2], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen,treur}@cs.vu.nl
[2] Radboud University Nijmegen, Nijmegen Institute for Cognition and Information
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract. Artificial Intelligence has contributed (formal) design models and software support tools to application areas such as architecture, engineering and software design This paper explores the effectiveness of applying design models to the area of organization (re)design. To that purpose a component-based model for (re)design of organizations is presented as a specialization of an existing generic design model. Using recently developed formalizations within Organization Theory organization models are described as design object descriptions, and organization goals as design requirements. A design process specification is presented that models the redesign process for an organization that adapts to changes in the environment. The formally specified and implemented approach to organization redesign thus obtained has been tested for a well-known historical case study from the Organization Theory literature.

1 Introduction

Organizations are created to smoothen processes in all aspects of society, even in the artificial societies of software agents. From a design perspective organizations have goals to be achieved or maintained that serve as requirements for their functioning. The behavior of the elements or parts of the organization and their interaction together should result in overall organization behavior that fulfills the goals of the organization. Environmental circumstances impose constraints on the organization with respect to the way its goals can be fulfilled. As the environment changes over time, so do these constraints. To adapt to such changes in constraints, the organization might have to change itself. From a design perspective the changing constraints can be interpreted as changing requirements for a redesign problem.

Within the area of AI and Design, in the last decade formally specified generic models for (re)design processes have been developed; e.g., [1, 4]. Application of a generic redesign model to the area of organizations requires specialized knowledge on: (1) organization goals; (2) how to derive refined requirements from such goals given a variable environment; (3) the current design object description, and (4) what components for a design object satisfy which requirements. A redesign process results in a new design object description as a modification of the existing one and a specification of changed (new) design requirements.

P. Noriega et al. (Eds.): COIN 2006 Workshops, LNAI 4386, pp. 48–64, 2007.

A redesign process as formally modeled in [4] involves generation and modification steps both for the specification of the requirement set and for the design object description. A formal model of a redesign process thus requires formalizations of design objects, design requirements, and of the dynamics of redesign processes. This paper proposes such formalizations for the area of organizational (re)design, in the context of a component based model for (re)design of organizations. Formalized organization models [5,10,11,14,18] serve as design object descriptions. Formalizations of organizational behavior are used for design requirements specifications [10,11,14,18]. Finally, for design process dynamics a formalization is used as put forward in [1]. The resulting approach contributes to the organization redesign domain in that it facilitates formal modeling, simulation and verification of the redesign process, supported by modeling and analysis tools.

Section 2 gives the component-based model for the design and redesign process and describes the types of domain specific knowledge needed in such a process. Section 3 addresses the formalization of design object descriptions by means of an organization model format in which different components and aggregation levels are distinguished. In Section 4 the relation between goals, changing environment and requirements is described, illustrated for cases described in Organization Theory. Section 5 presents the method of requirements refinement and shows a specific example. Thereafter, Section 6 presents examples of design object that are known to satisfy certain design requirements, and Section 7 presents generic properties which enable an evaluation of the successfulness of the whole (re)design process. Section 8 presents simulation results of the model, and finally Section 9 is a discussion.

2 A Component-Based Model for (Re)design of Organizations

This Section presents a component-based generic model for design of organizations based on requirements manipulation and design object description manipulation. The component-based model presented draws inspiration from [4] and was specified within the DESIRE [3] framework. It is composed of three components, see Figure 1:

- RQSM, for Requirement Qualification Set Manipulation, acquires requirements, for example, by elicitation from managers within a company. Within RQSM the appropriate requirements are determined in relation to the goals set for the organization and the current environmental conditions. After having selected a set of requirements, these are refined to more specific ones.
- DODM, for Design Object Description Manipulation, creates a design object description based on the (specific) requirements received from RQSM. In order to do this, a number of alternative solutions known to satisfy the requirements are generated and according to certain strategic knowledge one of those is selected.
- Design Process Coordination (DPC) is the coordinating component for the design process. The component determines the global design strategy (e.g., [4]) and can evaluate whether the design process is proceeding according to plan.

Information exchange possibilities are represented by the links between input and output of the components and the input and output of the model. Input and output are represented by the small boxes left and right of components.

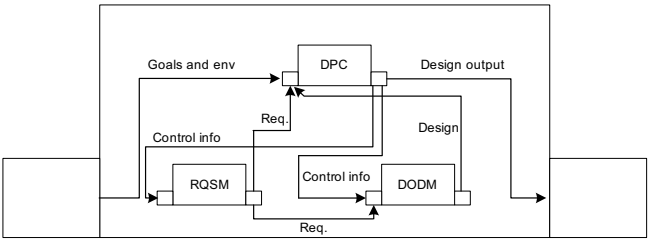

Fig. 1. Top level of the design model

The next sections describe the three components in more detail. The model as described here, is a generic design model for organizational design without application- or domain-specific knowledge. In later sections such knowledge is specified for a case study.

RQSM. This component is composed of two sub-components, namely Requirements Sets Generation and Requirements Set Selection, see Figure 2.

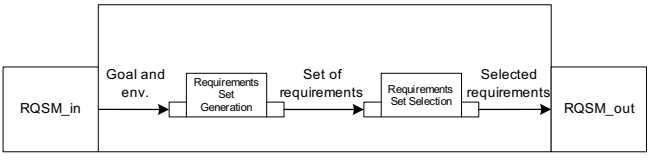

Fig. 2. Components within RQSM

The component Requirements Sets Generation receives as an input the current environmental conditions and the organizational goals. The sub-component contains knowledge on what requirements entail fulfillment of organizational goals given the environmental conditions. Such knowledge can be depicted in the form of AND/OR trees as shown in Figure 3.

If for example E1 is observed, requirement R1 is an example of a requirement that, when fulfilled, guarantees to satisfy goal G under environmental conditions E1. If the environment changes to situation E2, the requirement has to change as well; the example tree shows how R1 can be changed to requirement R2 that guarantees G under the new environmental conditions E2. Note that these environmental conditions can be defined as an abstraction of the potentially infinite actual environment. This resembles how a manager would define such requirements, for instance by just looking at a few specific aspect of the environment, and basing his/her requirement for the organization on those. After a requirement is determined, it can be refined in order to obtain requirements on a more specific level. Making such a requirement more specific can result in several options being generated. For example, it might be possible to establish a certain market share by having the best quality products but also by having the lowest priced products. After having refined each of the requirements, all possible sets of refined requirements are forwarded to the component Requirements Set Selection.

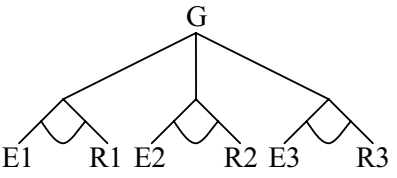

Fig. 3. Example AND/OR tree relating environmental conditions and requirements to a goal

After the component Requirements Set Selection has received the alternative sets of requirements its task is to select one of those alternatives, and to forward it to the component DODM which will in turn find a suitable organization design for such a requirement set. Different selection methods exist, e.g., explicit ranking, on the basis of strategic knowledge. Such strategic knowledge can for example be based on the source of requirements: requirements that originate from users can for example be preferred over those derived by default rules which are in turn preferred over requirements derived from previous requirements (see [12]).

DODM. This component receives a set of refined requirements from RQSM, which is handled by two sub-components, Design Object Description Generation and Design Object Description Selection. The design object descriptions are descriptions of designs of the organization, including both structural aspects as behavioral aspects.

Design Object Description Generation receives the requirements and delivers descriptions of possible alternative design objects (i.e., organization design descriptions), such that the (specific) requirements as received from RQSM are satisfied. To establish satisfaction, knowledge is needed that specifies what part of a design object contributes to fulfillment of a specific requirement. If, for example, the requirement is to produce products of the highest quality, then a satisfactory design is an organization having a department dedicated to checking quality and repairing of production errors. Again, there can be many possibilities available that satisfy the requirements. All alternatives found are forwarded to the component Design Object Description Selection.

The component Design Object Description Selection can use several criteria to choose the optimal design, such as operational costs effectiveness, and production time effectiveness. In order to make such a selection, the component has (strategic) knowledge concerning these aspects. It might for example know the typical price for hiring an agent for a particular role. Eventually, the component outputs a new design for the organization.

DPC. The component DPC is the component which determines the global design strategy and oversees whether the design process proceeds according to plan. Two different tasks are distinguished. DPC checks whether a design object description determined by DODM satisfies the refined requirements. It might for example be the case that the combination of two suitable design object parts causes a conflict. In case the refined requirements are not satisfied control information is passed to DODM stating that an alternative should be found (e.g., taking a different branch of an OR tree). In case these refined requirements are satisfied whereas the high-level requirements are not, the requirements refining process has failed, therefore control

information is given to RQSM to refine the requirements in another way (again by for example taking another OR branch).

3 Organization Models as Design Objects

An organizational structure defines different elements in an organization and relations between them. The dynamics of these different elements can be characterized by sets of dynamic properties. An organizational structure has the aim to keep the overall dynamics of the organization manageable; therefore the structural relations between the different elements within the organizational structure have to impose relationships or dependencies between their dynamics; cf. [18]. In the introduction to their book Lomi and Larsen [20] emphasize the importance of such relationships:

- 'given a set of assumptions about (different forms of) individual behavior, how can the aggregate properties of a system be determined (or predicted) that are generated by the repeated interaction among those individual units?'
- 'given observable regularities in the behavior of a composite system, which rules and procedures - if adopted by the individual units- induce and sustain these regularities?'

Both views and problems require means to express relationships between dynamics of different elements and different levels of aggregation within an organization. In [20] two levels are mentioned: the level of the organization as a whole versus the level of the units. Also in the development of MOISE [11,12,14] an emphasis is put on relating dynamics to structure. Within MOISE dynamics is described at the level of units by the goals, actions, plans and resources allocated to roles to obtain the organization's task as a whole. Specification of the task as a whole may involve achieving a final (goal) state, or an ongoing process (maintenance goals) and an associated plan specification.

The approach in this paper is illustrated for the AGR [9] organization modeling approach. Figure 4 shows an example organization modeled using AGR. Within AGR organization models three aggregation levels are distinguished: (1) the organization as a whole; the highest aggregation level, denoted by the big oval, (2) the level of a group denoted by the middle size ovals, and (3) the level of a role within a group denoted by the smallest ovals. Solid arrows denote transfer between roles within a group; dashed lines denote inter-group interactions. This format is adopted to formalize organization models as design object descriptions. In addition, behavioral properties of elements of an organization are part of a design object description. TTL [17] is used to express such behavioral properties.

In TTL state ontology is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} 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). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow STATES(Ont)$, i.e., a sequence of states γ_t $(t \in T)$ in STATES(Ont). The set of all traces over state ontology Ont is denoted

by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Σ) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

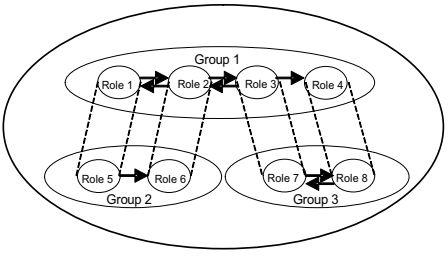

Fig. 4. An AGR Organization Structure

Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) |= p denotes that state property p holds in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as ¬, ∧, ∨, ⇒, ∀, ∃. A special software environment has been developed for TTL, featuring both a Property Editor for building and editing TTL properties and a Checking Tool that enables formal verification of such properties against a set of (simulated or empirical) traces.

4 RQSM: Changing Requirements Upon Environmental Change

Organizational requirements change due to changing environmental circumstances. The circumstances are input to RQSM. The general pattern is follows. A certain organizational goal G (e.g. sufficient demand) is no longer reached, due to an environmental change, say from E1 to E2. In the old situation requirement R1 was sufficient to guarantee G under environmental condition E1: E1 & R1 ⇒ G. Here R1 is a requirement expressing a relation which states that under the condition E1 the organization is able to achieve G. The change from E1 to E2 makes that requirement R1, which is still fulfilled but has become insufficient, is to be replaced by a new, stronger requirement R2 which expresses that under environment E2 goal G can be achieved; therefore: E2 & R2 ⇒ G. Thus, the organization is triggered to change to fulfill R2 and as a consequence fulfill goal G again.

Jaffee [16] distinguishes several classes of external triggers for organizational change: triggers in the organization's *input*, (e.g., changes in the resources or suppliers), and triggers in *enabling / constraining factors* such as government/labor rules and (new) technology. Government regulations for workers might affect human

resource practices and composition of the workforce. Concerning labor aspects, the union might demand a reduction from 40 to 36 hours a week, which naturally causes organizational change. Examples of input triggers are *resources* that run out, becoming a lot more expensive, c*ustomers* whose demands decrease for the good being produced, and *competitors* changing their production methods causing more efficient production for products within the same product group. Another example of an input-base external trigger is the case that at time t *suppliers* increase their price of a product P, which is used by the organization for the production, from M_1 to M_2. A formal form of this environmental condition is specified in E1 using the Temporal Trace Language (TTL) as explained in Section 3.

E1(P, M, t): Supplier Price
∃R:REAL state(γ, t) |= environmental_condition(price(P, R), pos) & R ≤ M

Before the environmental change, E1(P1, M_1, t) specifies the relevant property of the environment. After the change of supplier price however, this property no longer holds whereas E1(P1, M_2, t) does hold. The overall goal to be maintained within the organization is to keep the demand of product P above a threshold D. A formal specification of the goal is presented in OP1.

OP1(P, D, t): Sufficient demand
∃I:INTEGER state(γ, t) |= environmental_condition(customer_demand(P, I), pos) & I ≥ D

The requirement imposed for the organization is to maintain the goal of keeping demand for product P2 above D, in the new situation given the environmental condition of the price M for product P1 which is needed for the production of P2. This requirement is specified below in property R.

R(P1, P2, M, D): Maintain demand
∀t :TIME [state(γ, t) |= needed_for_production_of(P1, P2) & E1(P1, M, t)] ⇒ OP1(P2, D, t)

Before the change in the environment, requirement R1 which is R(P1, P2, M_1, D) was sufficient to ensure the goal being reached. After the change however, this requirement is still satisfied but might be insufficient to ensure the goal. This is due to the fact that the environmental condition E1 in the antecedent of E1 & R1 ⇒ G does not hold, and hence, cannot be used to entail G (although the requirement R1 is fulfilled all the time). The requirement is therefore withdrawn and replaced by the requirement R2 which is R(P1, P2, M_2, D). This R2, however, is not necessarily satisfied and may require an organizational change to enable fulfillment.

5 RQSM: Refining Requirements Based on Interlevel Relations

To fulfill requirements at the level of the organization as a whole as discussed in Section 4, parts of the organization need to behave adequately (see also the central challenges put forward by Lomi and Larsen [20] as discussed in Section 2). Based on this idea, in this paper dynamics of an organization are characterized by sets of dynamic properties for the respective elements and aggregation levels of the organization. An important issue is how organizational structure (the design object description determined in DODM) relates to (mathematically defined) relationships

between these sets of dynamic properties for the different elements and aggregation levels within an organization (cf. [18]). Preferably such relations between sets of dynamic properties would be of a logical nature; this would allow the use of logical methods to analyze, verify and validate organization behavior in relation to organization structure. Indeed, following [18], in the approach presented below, logical relationships between sets of dynamic properties of elements in an organization turn out an adequate manner to (mathematically) express such dynamic cross-element or cross-level relationships.

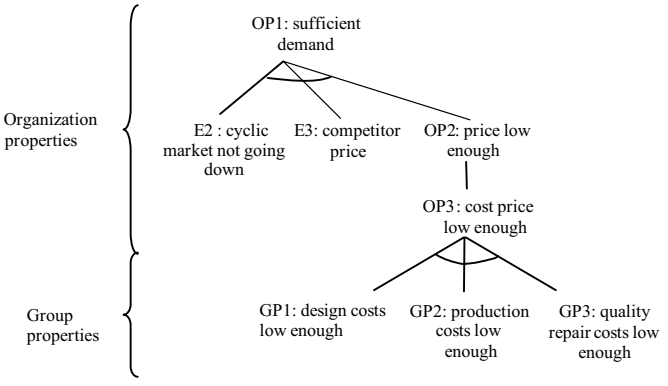

Fig. 5. Hierarchy of Organizational and Group properties

Figure 5 shows an example of a hierarchy of dynamic properties for an organization producing certain products, the properties follow field observations at the Ford Motor Company in 1980 described in [25]. The overall organizational goal is to maintain sufficient demand for the goods being produced, as was also the case in OP1 in Section 4. The organization has separate departments for design, production and quality control, which are modeled as groups in the organization. The highest levels represent organizational properties or goals at the aggregation level of the organization as a whole, whereas the lowest level shown here represents properties at the aggregation level of the groups. Note that the fact that these are group properties already restricts the design of the object in DODM, which makes the process less complex.

A definition for each of the properties in Figure 5 is presented below. Notice that this hierarchy could easily be extended by other aspects (e.g., of quality of the products as a reason for the demand decreasing or not). Property OP1 is described in Section 4. One of the environmental conditions is that the cyclic market is not going down for a product P at time t in case the demand for the product group as a whole (i.e., all goods produced by different companies in this particular category) is not going down.

E2(P, t): Cyclic market not going down

$\forall$G:PRODUCT_GROUP, I1,I2:INTEGER
[state(γ, t) |= belongs_to_product_group(P, G) &
state(γ, (t-1)) |= environmental_condition(customer_demand(G, I1), pos) &
state(γ, t) |= environmental_condition(customer_demand(G, I2), pos)]
$\Rightarrow$ I2 $\geq$ I1

Furthermore, an environmental condition E3 poses a requirement on the price of competitors in the form of the average price of products within the product group to which product P belongs. These prices should not be higher than V:

E3(P, V, t): Competitor Price
∀G:PRODUCT_GROUP, V1:REAL [[state(γ, t) |= belongs_to_product_group(P, G) &
state(γ, t) |= environmental_condition(average_price(G,V1), pos)] ⇒ V1 ≥ V]

To achieve goal OP1 given environmental conditions E2 and E3, the price of the products being produced by the organization should be low enough, which in turn is the requirement posed on the organization. Prices are considered low enough for a product P at time t in case the price for the product is equal or below the average price level within the product group (i.e. prices are ≤ V as set above).

OP2(P, V, t): Price low enough
∀G:PRODUCT_GROUP, V1:REAL [state(γ, t) |= price(P, V1)] ⇒ V1 ≤ V

Whether the price is low enough depends on the cost price for the particular product P at time t, which purely depends on the costs for the different groups within the organization, as expressed in the group properties (GP's).

OP3(P, V, t): Cost price low enough
∀V1,V2,V3:REAL [state(γ, t) |= design_cost(P, V1) & state(γ, t) |= production_cost(P, V2) &
state(γ, t) |= quality_repair_cost(P, V3)] ⇒ V1+V2+V3 ≤ V

Finally, the individual group properties can be specified such that the costs of each group are below a certain value. The division of such costs over groups is a refinement choice. An example decision could be to allow only a small percentage of the costs for quality repair and to divide the brunt of the costs equally over production and design. Each group should meet their individual requirements. First of all, design costs should be low enough:

GP1(P, V1, t): Design costs low enough
∀Q:REAL [state(γ, t) |= design_cost(P, Q)] ⇒ Q ≤ V1

Also, the production costs for product P should be low enough:

GP2(P, V2, t): Production costs low enough
∀Q:REAL [state(γ, t) |= production_cost(P, Q)] ⇒ Q ≤ V2

Finally, quality repair costs should be low enough for product P:

GP3(P, V3, t): Quality repair costs low enough
∀Q:REAL [state(γ, t) |= quality_repair_cost(P, Q)] ⇒ Q ≤ V3

After having generated all options in RQSM, selection knowledge is used to select one of the available options. In this paper, such selection knowledge is not further addressed. The output of RQSM is, however, of the form selected_basic_refinement_set(RS) where RS is a name for a requirements set. The elements within this set are defined as follows: in_selected_basic_refinement_set(R, RS) where R is a requirement, as the ones shown above, and RS is the selected basic refinement set.

6 DODM: Constructing Design Objects

As stated in Section 2, DODM contains a library of templates for (parts of) design objects which are known to satisfy certain requirements (of the form as specified in the last paragraph of the previous section). For the case study, the DODM library contains two templates. One of those is a template in which a mass production system is used to produce goods. Such a system produces goods at reasonable production costs but at high quality repair costs. The template for mass production includes a group of production workers (e.g. a production worker for attaching a wheel to a car). The mass production template also contains a quality repair department of considerable size with quality repair worker roles.

The second template in the library is a lean production organization. Lean production has no quality repair costs, since there is no separate quality repair department. The production costs are at the same level as the production costs for mass production organizations. In the lean production method (see e.g. [25]), multi-task production workers are present which perform several tasks, and also handle errors in case they are observed. As a result of such immediate error detection and correction, a quality repair department is not present within a lean production model.

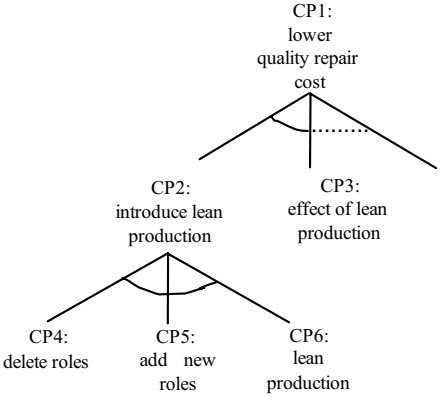

Fig. 6. Redesign options specified in the form of an AND/OR tree

Figure 6 shows an example AND/OR tree for DODM (focusing at lean production as a solution) in which options for changes in a design object not satisfying the requirement that design costs are low enough. The specific changes in the design object are presented below. First of all, the highest level property states that design costs will at least at the required level within a duration d:

CP1(P, D, t):Lower Quality Repair Costs
∀V1,V2:REAL [state(γ, t) |= selected_basic_requirement_in(GP3(P, V1, t), RS) &
state(γ, t) |= DOD_includes(D, quality_repair_cost(P, V2)) & V1 < V2]
⇒ ∃t2:TIME > t, V3:REAL [t2 < t+d & state(γ, t2) |=DOD_includes(D,quality_repair_cost(P, V3)) & V3 ≤ V1]

On a lower level, property CP2(P, D, t) specifies the introduction of lean production into an organization. This reduces the quality repair costs to 0 as shown by CP3(P, D, t).

Although more options are possible for reducing quality repair costs, shown by the dots in Figure 6, these are not addressed in this paper.

CP2(P, D, t): Introduce Lean Production
∀V1,V2:REAL [state(γ, t) |= selected_basic_requirement_in(GP3(P, V1, t), RS) &
state(γ, t) |= DOD_includes(D, design_cost(P, R2)) & V1 < V2]
⇒ ∃t2:TIME > t [t2 < t + d & state(γ, t2) |= DOD_includes(D, lean_production_method(P))]

CP3(P, D, t): Effect of Lean Production
[state(γ, t) |= DOD_includes(D, lean_production_method(P)) ⇒
state(γ, t) |= DOD_includes(D, quality_repair_cost(P, 0))]

Introducing a lean production system entails that within the production process the specialized roles for mass-production and quality repair department are deleted.

CP4(P, D, t): Delete Roles
∀R1,R2:REAL [state(γ, t) |= DOD_includes(D, lean_production_method(P))
⇒ ∃t2:TIME > t [t2 < t + d &
state(γ,t2)|= ¬DOD_includes(D,exists_role(spec_production_worker)) &
state(γ, t2)|= ¬DOD_includes(D,exists_group(quality_repair_group))]]

Moreover, roles are created that perform multiple tasks, and teams are created such that the roles combined in the team have all the abilities to make a car.

CP5(P, D, t): Add New Roles
∀R1,R2:REAL [state(γ, t) |= DOD_includes(D, lean_production_method(P))
⇒ ∃t2:TIME > t, ∀A:AGENT
[t2 < t + d & state(γ, t2) |= DOD_includes(D, exists_role(multi_task_production_worker)) &
state(γ, t2) |= DOD_includes(D, previously_allocated_to(A, spec_production_worker, production_group)) &
state(γ, t2) |= DOD_includes(D, allocated_to(A, multi_task_production_worker, production_group))]]

Agents that were allocated to the deleted roles in the production process are allocated to the newly formed roles. Agents formerly allocated to a role in quality repair are fired. Once the system is organized in this fashion, quality repair in a separate department becomes obsolete, and quality repair costs are down to 0 as the production workers are now performing the task. CP6 expresses that the measures as described in CP4 and CP5 results in a lean production method for the product P:

CP6(P, D, t): Lean Production
∀A:AGENT
[state(γ, t) |= ¬ DOD_includes(D, exists_role(spec_production_worker)) &
state(γ, t) |= ¬ DOD_includes(D, exists_group(quality_repair_group)) &
state(γ, t) |= DOD_includes(D, exists_role(multi_task_production_worker)) &
state(γ, t) |= DOD_includes(D, previously_allocated_to(A, spec_production_worker, production_group))
state(γ, t) |= DOD_includes(D, allocated_to(A, multi_task_production_worker, production_group))]
⇒ ∃t2:TIME < t + d state(γ, t2) |= DOD_includes(D,lean_production_method(P))

After such options for (re)design of the object have been generated based on the requirements, selection knowledge is used to select one of the options that have been generated. This knowledge is not addressed in this paper. Eventually, DODM outputs a design object description of the form selected_DOD_output(D) where D is the design object description. Furthermore to identify properties of the DOD or its parts, output of the form in_selected_DOD_output(P,D) is generated where P is a property of (a part of) the DOD and D is the selected DOD. This is based on the internal information represented in the form of DOD_includes(D, P).

7 (Re)design Process Evaluation

This section addresses the evaluation of the whole design process. The overall design process is successful when both RQSM and DODM show the proper behavior.

RQSM shows the proper behavior in case it generates requirements, and these requirements indeed result in the goal set for the organization being met. Such properties are formulated in a formal form below.

RQSM_generate
If RQSM receives new environmental conditions on its input, then RQSM eventually generates a set of requirements

∀t:TIME, γ:TRACE, E:ENV_COND [[state(γ, t, input(RQSM)) |= environment_property(E) &
¬∃t':TIME < t [state(γ, t', input(RQSM)) |= environment_property(E)]]
⇒ ∃t2:TIME > t, G:GOAL, RS:REQUIREMENT_SET [state(γ, t2, output(RQSM)) |= main_requirement(G) &
state(γ, t2, output(RQSM)) |= selected_basic_refinement_set(RS)]]

RQSM_successful
If RQSM generates requirements, then the combination of these requirements entail the goal set for the organization.

∀t:TIME, γ:TRACE, RS :REQUIREMENT_SET, G :GOAL
[[state(γ, t, output(RQSM)) |= main_requirement(G) &
state(γ, t, output(RQSM)) |= selected_basic_refinement_set(RS)] ⇒ entails_goal(RS, G)]

DODM shows the proper behavior in case it first of all generates a design object description in case a new requirement set is received. Besides simply generating such a design object description, the object also needs to satisfy the requirements received on its input.

DODM_generate
If DODM receives a new requirements set on its input, then DODM eventually generates a design object description as output.

∀t:TIME, γ:TRACE, RS :REQUIREMENTS_SET
[state(γ, t, input(DODM)) |= selected_basic_refinement_set(RS) &
¬∃t':TIME < t [state(γ, t', input(DODM)) |= selected_basic_refinement_set(RS)]
⇒ ∃t2:TIME, D:DESIGN_OBJECT_DESCRIPTION
state(γ, t2, output(DODM)) |= selected_DOD_output(D)]

DODM_successful
If DODM generates a design object description as output, then the design object description satisfies the requirements set on the input of DODM.

∀t:TIME, γ:TRACE, R :REQUIREMENT_SET,
D:DESIGN_OBJECT_DESCRIPTION
[state(γ, t, input(DODM)) |= selected_basic_refinement_set(R) &
state(γ, t, output(DODM)) |= selected_DOD_output(D)]
⇒ fulfills_requirements(D, R)

8 Simulation Results

In order to show the functioning of the model presented above, simulation runs have been performed based on the properties as identified in Sections 4-6 using the component-based design presented in Section 2. As a scenario for the case study, a sudden decrease of competitor price is inserted as an event into the simulation

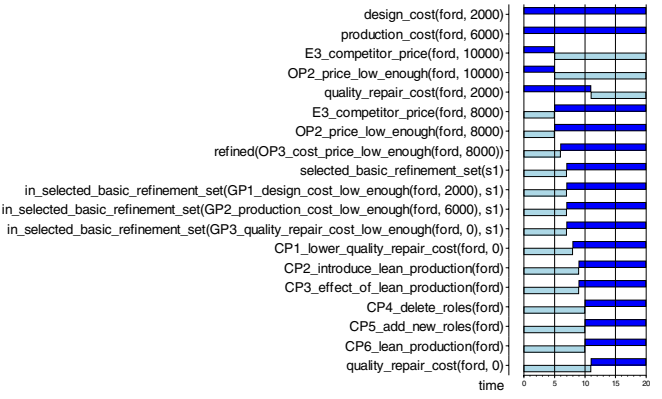

Fig. 7. Case study simulation results

(following [25]). Figure 7 shows a partial trace of the simulation results. In the figure, the left side shows the atoms that occur during the simulation whereas the right side shows a timeline where a dark gray box indicates an atom being true at that particular time point and a light gray box indicates the atom is false.

The figure shows the following. Initially, the different cost factors for the ford design object are the following: design_cost(ford, 2000); production_cost(ford, 6000); quality_repair_cost(ford, 2000). This perfectly fulfills the requirement that price is considered to be low enough in case it is at most 10000 as expressed in OP2 at that time point: OP2_price_low_enough(ford, 10000) . This requirement is sufficient to guarantee the goal OP1 (as expressed in Figure 5) due to the environmental condition E3 that competitor price for products within the same product group as ford are at that exact same level: E3_competitor_price(ford, 10000). Furthermore, the cyclic market should not be going down (E2) which is left constant during this simulation. Suddenly however, the environment changes, the price of competing cars drops to 8000: E3_competitor_price(ford, 8000). The current property OP2 is now insufficient to guarantee the overall goal OP1 being satisfied, therefore, a redesign process is activated. RQSM determines a new requirement for the design object, namely that prices should be below 8000, the competitor car price:OP2_price_low_enough(ford, 8000). Other options might be possible as well, but are not addressed in the simulation. The requirement is refined, first of all by expressing that the cost price should be low enough: refined(OP3_cost_price_low_enough(ford, 8000)). This results in a selected basic refinement that quality repair costs should become 0 whereas design and production costs can remain 2000 and 6000 respectively, as shown in the requirements part of the selected refinement s1:

in_selected_basic_refinement_set(GP1_design_cost_low_enough(ford, 2000), s1);
in_selected_basic_refinement_set(GP2_production_cost_low_enough(ford, 6000), s1);
in_selected_basic_refinement_set(GP1_quality_repair_cost_low_enough(ford,0),s1).

Since these are basic refinements, they are passed to DODM in order to find templates appropriate for these basic requirements. DODM observes that quality repair costs for the current design object are too high, and therefore starts to use the tree as expressed in Figure 6, refining the exact changes to be performed on the design object more and more. First the introduction of the lean production system is

chosen, as expressed in CP2. Thereafter, the more concrete changes are determined, namely the deletion of the specialized production worker roles, the addition of new multi-task roles, and the insertion of the new behavior of those roles: CP4_delete_roles(ford); CP5_add_new_roles(ford); CP6_lean_production(ford). Note that in the simulation the actual contents of such properties are more concrete (in the form of current DOD descriptions), however, these are not presented here for the sake of brevity. Finally, after the actual changes have been performed for the design object, quality repair costs drop to 0, and the goal is therefore satisfied again: quality_repair_cost(ford, 0). To see whether the properties as expressed in Section 7 hold for the simulation trace, first of all, the RQSM_generate and DODM_generate properties have been checked against the trace shown in Figure 7 using a software tool called the TTL Checker [17]. Both properties were shown to hold for the trace. In order to see whether the refinement process within RQSM is properly performed, the tree used for the simulation as presented before in Section 5 has been formally proven by means of the SMV model checker [22]. The results indeed show that the lowest level properties entail the goal given the environmental conditions. Furthermore, to prove the successfulness of DODM, the property hierarchy shown in Figure 6 has also been proven by the SMV model checker which shows that introducing lean production in a design object indeed results in canceling the quality repair costs, which satisfied the property DODM_successful. As a result, the DODM_successful property is satisfied as well as the RQSM_successful property in case the components indeed generate the output based on these property hierarchies.

9 Discussion

Organizations aim to meet their organizational goals. Monitoring whether events occur that endanger fulfillment of these goals enables organizations to consciously adapt and survive. Adaptation is essential once an organizational goal becomes unreachable. This paper views such a change as a (re)design process. A component-based formal generic model for design developed within the area of AI and Design is specialized into a model for organization (re)design.

Formalizations developed within Organization Theory and AI (or computational organization theory), have proved suitable for the description of organization models as design object descriptions, and organization goals as design requirements. Furthermore, different types of specialized knowledge have been identified: (1) about main organization goals and their relation for given environmental conditions to organization requirements, (2) about refinement of organization requirements, (3) about design object descriptions, and (4) which components for a design object description satisfy which requirements. The generic design model was instantiated with such types of knowledge to constitute a specialized component-based model for (re)design of organizations. Example properties have been taken from a well known case in Organization Theory on the introduction of lean production [25].

This paper focuses on external triggers for organizational change. Triggers are related to specific goals that play the role of design requirements which the organizational change should comply to. These requirements tend to be high-level goals and lack the detail needed for specifying how an organization should change.

Therefore, design requirement refinement is used based on requirements hierarchies. Such hierarchies relate objectives of the organization (e.g., high demand for cars) to organizational change properties at different organizational levels (e.g., change in some departments). Thus, they relate triggers at the level of the organization to properties at the level of parts (groups) within the organization. For example, that a certain type of car is not selling according to the goals set is related to the costs of quality repair. Requirements hierarchies help to localize where to change the organization. High-level goals for an organization as well as goals for organizational redesign have been related to low-level executable properties. Formal verification has been performed, showing satisfaction of the non-leaf properties in the property tree.

When comparing the approach to previous work in redesign of organizations a main strength is the formal description of the whole redesign process in terms of a generic redesign model for organizations. In the field of management for example (e.g., [7]), only informal descriptions are given of redesign processes. Systems Theory (e.g., [23]), addresses goal oriented behavior. The gap observed between actual and desired state of a system causes redesign, which corresponds with the approach taken in this paper. Formalizations by means of property hierarchies are, however, not present, therefore formal verification as done in this paper cannot be performed.

In [13] a general diagnosis engine is presented which drives adaptation processes within multi-agent organizations using the TAEMS modeling language as the primary representation of organizational information. In the design of the diagnostic engine three distinct layers are identified: symptoms, diagnosis, and reactions which in the approach presented in this paper roughly correspond to Section 4, 5, and 6 respectively. The implementation of these elements differs in both approaches. The goals and requirements in this paper are explicitly connected to each other. Once an organizational goal is observed not to be fulfilled, such a dissatisfaction is related directly to a goal for change. In the approach presented in [13] lacks such an explicit relation between goals and error diagnosis. Furthermore, this paper also introduces an approach to diagnose whether the whole reorganization process was successful, which is not the case in [13]. [6] explores dynamic reorganization of agent societies and focuses on changes to the structure of an organization, this paper presents an approach that enables such a dynamic reorganization.

[15] presents an approach which aims to archive adaptive real-time performance through reorganizations of the society. As a domain of application, production systems are used throughout that paper. Whereas that paper focuses on adaptive agents, this paper concentrates on adaptation of an organizational model that abstracts from agents and specifies elements on the level of roles the agents can fulfill.

The work presented in this paper can also be compared with the work on institutions as a way to describe multi-agent organizations. In [8] an institution is said to structure interactions and enforce individual and social behavior by obliging everybody to act according to norms, and a formalization language is introduced for such an institution. The approach to use dynamic expressions as a restriction of the behavior of agents allocated to that role used in this paper is also expressive enough to describe such norms. For example, in [21] an example of a norms is the following: “Students are prohibited from sitting the exam if they have not completed the assignment”. Such a norm can easily be formulated in terms of a dynamic property for the student role. The approach presented in this paper could therefore also be applied

to institutions and normative organizations. In [2] an adaptation mechanism of norms is proposed using an evolutionary approach contrary to the pre-specified knowledge assumed in this paper. Such an evolutionary approach can be incorporated in RQSM and DODM, allowing them to derive requirements and design objects for certain environmental conditions and goals without using pre-specified knowledge.

Finally, in the field of coalition formation (see e.g. [19, 24]), the main purpose of forming a coalition is to perform a task that cannot be performed by a single agent. That work can be combined with our approach by addressing the problem of the allocation of agents to roles, after the change of the organizational model by the approach presented in this paper.

References

1. Bosse, T., Jonker, C.M., Treur, J.: Analysis of Design Process Dynamics. In: de Mantaras, R.L., Saitta, L. (eds.) Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'04, pp. 293–297 (2004)
2. Bou, E., Lopez-Sanchez, M., Rodriguez-Aguilar, J.A.: Towards Self-Configuration in Automatic Electronic Institutions. LNCS, vol. 4386, pp. 247–263. Springer, Heidelberg (2007)
3. Brazier, F.M.T., Jonker, C.M., Treur, J.: Principles of Component-Based Design of Intelligent Agents. Data and Knowledge Engineering 41, 1–28 (2002)
4. Brazier, F.M.T., van Langen, P.H.G., Treur, J.: Strategic knowledge in design: a compositional approach. Knowledge-Based Systems 11, 405–415 (1998)
5. Ciancarini, P., Wooldridge, M.J. (eds.): AOSE 2000. LNCS, vol. 1957. Springer, Heidelberg (2001)
6. Dignum, V., Sonenberg, L., Dignum, F.: Dynamic Reorganization of Agent Societies. In: Proc. of CEAS: Workshop on Coordination in Emergent Agent Societies at ECAI 2004 (2004)
7. Douglas, C.: Organization redesign: the current state and projected trends. Management Decision 37(8) (1999)
8. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 348–366. Springer, Heidelberg (2002)
9. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organisations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), pp. 128–135. IEEE Computer Society Press, Los Alamitos (1998)
10. Hannoun, M., Sichman, J.S., Boissier, O., Sayettat, C.: Dependence Relations between Roles in a Multi-Agent System: Towards the Detection of Inconsistencies in Organization. In: Sichman, J.S., Conte, R., Gilbert, N. (eds.) Multi-Agent Systems and Agent-Based Simulation. LNCS (LNAI), vol. 1534, pp. 169–182. Springer, Heidelberg (1998)
11. Hannoun, M., Boissier, O., Sichman, J.S., Sayettat, C.: MOISE: An organizational model for multi-agent systems. In: Monard, M.C., Sichman, J.S. (eds.) SBIA 2000 and IBERAMIA 2000. LNCS (LNAI), vol. 1952, pp. 152–161. Springer, Heidelberg (2000)
12. Haroud, D., Boulanger, S., Gelle, E., Smith, I.F.C.: Strategies for conflict management in preliminary engineering design. In: Proceeding of the AID 1994 Workshop Conflict Management in Design (1994)

13. Horling, B., Benyo, B., Lesser, V.: Using Self-Diagnosis to Adapt Organizational Structures. In: Muller, J.P., Ander, E., Sen, S., Frasson, C. (eds.) Proceedings of the Fifth International Conference on Autonomous Agents, pp. 529–536. ACM Press, New York (2001)
14. Hubner, J.F., Sichman, J.S., Boissier, O.: A Model for the Structural, Functional and Deontic Specification of Organizations in Multiagent Systems. In: Proc. 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), Porto de Galinhas, Brasil (2002) Extended abstract in: Castelfranchi, C., Johnson, W.L. (eds.) Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'02. pp. 501–502, ACM Press, New York (2002)
15. Ishida, T., Yokoo, M., Gasser, L.: An Organizational Approach to Adaptive Production System. In: Proceedings of the 8th National Conference on Artificial Intelligence, Boston, USA, pp. 52–58 (1990)
16. Jaffee, D.: Organization Theory: Tension and Change. McGraw-Hill Publishers, New York (2001)
17. Jonker, C.M., Treur, J.: Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. Int. J. of Cooperative Information Systems 11, 51–92 (2002)
18. Jonker, C.M., Treur, J.: Relating Structure and Dynamics in an Organisation Model. In: Sichman, J.S., Bousquet, F., Davidsson, P. (eds.) MABS 2002. LNCS (LNAI), vol. 2581, pp. 50–69. Springer, Heidelberg (2003)
19. Klusch, M., Gerber, A.: Dynamic Coalition Formation among Rational Agents. IEEE Intelligent Systems 17(3), 42–47 (2002)
20. Lomi, A., Larsen, E.R.: Dynamics of Organizations: Computational Modeling and Organization Theories. AAAI Press, Menlo Park (2001)
21. McCallum, M., Vasconcelos, W.W., Norman, T.J.: Verification and Analysis of Organisational Change. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.) Proc. 1st OOOP Workshop, pp. 91–106 (2005)
22. McMillan, K.: Symbolic Model Checking: An approach to the state explosion problem. Kluwer Academic Publishers, Dordrecht (1993)
23. Rapoport, A.: General System Theory. Abacus Press (1986)
24. Shehory, O., Kraus, S.: Task allocation via coalition formation among autonomous agents. In: Proceedings of IJCAI 1995, pp. 655–661 (1995)
25. Womack, J.P., Jones, D.T., Roos, D.: The Machine That Changed The World: The Story of Lean Production. HarperCollins Publishers, New York (1991)