Balancing Organizational Regulation and Agent Autonomy: An MDE-based Approach[⋆]

Loris Penserini[1], Virginia Dignum[1], Athanasios Staikopoulos[2], Huib Aldewereld[1], Frank Dignum[1]

[1] Universiteit Utrecht - Institute of Information and Computing Sciences
P.O.Box 80089, 3508 TB Utrecht, The Netherlands
email: {loris,dignum,virginia,huib}@cs.uu.nl
[2] Trinity College Dublin, Computer Science, Ireland
email: Athanasios Staikopoulos@cs.tcd.ie

Abstract. The deployment of agent societies —as complex systems— in dynamic and unpredictable settings brings forth critical issues concerning their design. Organizational models have been advocated to specify open systems in dynamic environments in order to accomplish the need to represent regulating structures explicitly and independently from acting components (or agents). Despite several frameworks have been proposed for the specification of organizational models, it is still a matter of design choice how to balance between regulative design and component flexibility.
We propose a design framework, discussing its advantages of having different degrees of abstraction at organizational level in the development of agent societies. That is, we illustrate how the design properties differently impact the flexibility of run-time systems to cope with context changes. To deliver on such an aim, we adopt the OperA software engineering methodology to deal with the organizational model specification, and the Model Driven Engineering (MDE) mechanisms to map concepts between different design models.

1 Introduction

To empower individuals and human organizations to achieve their goals and perform their tasks effectively, we need systems that are aware of dynamic and usually unpredictable physical and organizational/social contexts where tasks are performed. The current trend is that an increasing number of entities, from smart personal devices to legacy databases, are controlled by software agents. The deployment of agent societies in dynamic and unpredictable settings brings forth critical issues concerning the design, implementation and validation of their behavior [11, 7, 4]. Changes in the environment lead to alterations on the effectiveness of the organization and therefore in a need to reorganize, or at least, the need to consider the consequences of the change to the organizations effectiveness and possibly efficiency.

⋆ This work has been performed in the framework of the FP7 project ALIVE IST-215890, which is funded by the European Community. The author(s) would like to acknowledge the contributions of his (their) colleagues from ALIVE Consortium (http://www.ist-alive.eu).

Organizational models have been advocated to specify open systems in dynamic environments. By open we mean that components are not designed nor controlled by a common entity, and by dynamic we mean that unplanned and unspecified changes may occur at run time. That is, there is a need to represent the regulating structures explicitly and independently from the acting components (or agents). Organization models comprise structural and behavioral aspects [11, 7]. Structural aspects include the formal patterns of relationships between groups and individuals, and the norms that govern their interactions, while behavioral aspects include processes, rules, activities, and operational methods. Depending on the specific situation, a change may affect the structural aspects or the behavior aspects.

Several frameworks have been proposed for the specification of organizational models. However, it is still a matter of design choice how to balance between regulative design and component flexibility. In this paper, we present some initial considerations towards the formalization of design guidelines for organizational models and apply these to the case of modeling of crisis management systems in the Netherlands [12]. The work reported in this paper is part of the general development framework within the European FP7 ALIVE's project. In particular, with respect to the proposed ALIVE's design framework, we discuss possible advantages of having different levels of abstraction —i.e., *Organizational*, *Coordination* and *Service*— in the design process of agent-based systems. That is, we illustrate how the design properties differently impacts the flexibility of run-time systems to cope with context changes. To deliver on such an aim, we adopt and combine the OperA software engineering methodology [5] to deal with the organizational model specification, and the Model Driven Engineering (MDE) mechanisms [1] —i.e., MDA based transformations [8]— to map concepts between the different levels of abstraction.

The paper is organized as follows: Section 2 gives an overview of the ALIVE's design framework that includes the OperA methodology and the MDE techniques. Section 3 briefly recalls the ALIVE's case study about The Netherlands procedures to cope with a crisis management scenario. Section 4 provides some intuitions about the effects of having a more or a less abstract organizational model specification over the (run-time) agents autonomy. In Section 5, we illustrate how MDE techniques make possible to regulate the level of design abstraction. In Section 6, we apply our approach to a real case study of organizational models. Section 7 provides some preliminary remarks about advantages and disadvantages of our approach. Finally, Section 8 gives some conclusions.

2 Background

2.1 The ALIVE Approach

The ALIVE project aims to apply organizational concepts to the design and implementation of service-based software systems. The main focus of the project is to create complex systems based on the composition of (existing) services, through the addition of levels of abstraction. The project extends the current trend in service-oriented engineering by combining the latest in coordination and organization mechanisms and model driven design to create a framework for software and service engineering for "live" open systems of active services. An overview of the project's architecture is given in figure 1.

– The *Service Level* (SL) augments and extends existing service models with semantic descriptions to make components aware of their social context and of the rules of engagement with other services.
– The *Coordination Level* (CL) provides the means to specify, at a high level, the patterns (workflows) of interaction between services, using a variety of powerful coordination techniques from recent European research in the area.
– The *Organization Level* (OL) provides context for the other levels – specifying the organizational rules that govern interaction and using recent developments in organizational dynamics to allow the structural adaptation of distributed systems over time.

The advantage of added levels of abstraction to the design process of systems based on the composition of services is two-fold: 1) it is more intuitive to think in organizational structures and interactions while designing complex interactions for services, and the addition of these layers of abstraction allows for a gradual (fluent) transition from the system as foreseen to the actual implementation; 2) when changes happen in the environment (e.g., specific services become unavailable) the added levels of abstraction are an explicit representation of the conceptual steps made at design, thus giving additional information on why certain interactions (between the services) are as they are, which enables the system to dynamically cope with the changes.

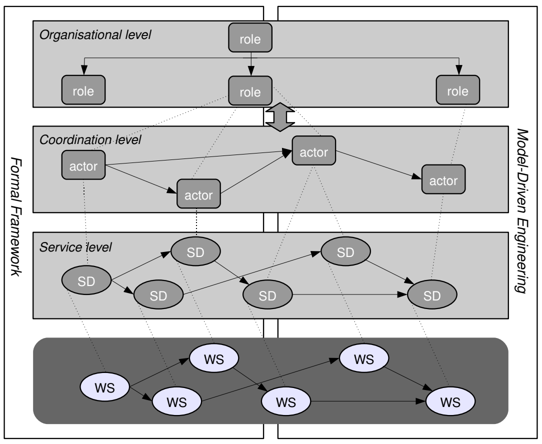


Fig. 1. The ALIVE framework for software and service engineering.

OL and CL have different philosophies: CL takes a bottom-up approach to modeling, by describing the behavior of the system as a plan workflow by describing the individual behavior of agents and specifying their activities. OL takes a top-down view, by describing the objectives of an organization. OL defines the desired result of the collective behavior, whereas CL describes the practice (i.e., the individual activities and interactions) that leads to that result. The difference in the viewpoints becomes clear when we compare the models used at OL and CL, as detailed below.

In the rest of this paper, we pay more attention on OL and CL. Specifically, we illustrate how different degrees of design abstractions at OL and CL may be differently affected by context changes that force the system to adapt.

The Organization Level of ALIVE. The *Organizational Level (OL)* defines the organizational structure of the society, describing roles and interactions, as intended by the organizational stakeholders. The OL specification is based on the OperA framework [5]. OperA enables the specification of abstractions, suitable both to model and study existing societies, as well as to develop new systems that participate in an organizational context. The main focus of OperA enable a suitable balance between global aims and requirements agent autonomy, their coordination needs, and environmental stakeholders' needs.

At the OL, social structures (describing roles and their relationships) as interaction structures (describing abstract patterns of interaction) and norms (describing organizational rules and requirements) are specified. These form the basis for the design of the coordination layer, by providing the organization information that agents will require to enact roles in the organization.

The Social Structure specifies objectives of the society, its roles and what kind of model governs coordination. The Social Structure is typically depicted in a role-dependency graph (e.g., Figure 2). Normative structures describe how to shape the freedom of role-enacting agents within the organization. While the Social Structure diagram deals with organizational objectives and social responsibilities between organizational roles, and the Normative Structure describes the regulations that hold in an organizations, the Interaction Structure specifies how to fulfill responsibilities and to achieve objectives, while still remaining compliant with a normative dimension (cf, Figure 6). Interaction Structure diagrams specify a partial ordering of the activities within the organization defined in terms of scenes and scene transitions. It abstracts over how the activities are done, but defines in which order what is supposed to happen such that organizational objectives are met without violating any organizational rules.

Interaction Scenes describe a scenario of activity, that is, how roles can interact and evolve. A scene is described by its players, results and the norms regulating the interaction (cf, Figure 6.(B)). Within a scene, *landmarks* define important states in the scene execution, that is, states that must be reached in any interaction between players to achieve a scene result. Landmark patterns provide a partial ordering of scene states (cf, Figure 6.(C)).

The Coordination Level of ALIVE. The OL provides the overall organization design that fulfills the stakeholders requirements. However, it does not specify how to structure groups of agents and constrain their behavior by social rules such that their combined activity will lead to the desired results. Coordination can be defined as the process of managing dependencies between activities [9]. That is, one way to coordinate is to manage functional dependencies. In this sense, coordination refers to task sharing and management, such that individual and shared goals are achieved. Another view if that of the system as an organization where dependencies are captured through supervision (e.g. influence, authority, etc.) and collaboration (e.g. teams) relationships between agents. In this sense, coordination refers to the specification of power and authority relations between agents.

2.2 Model Driven Engineering

Model Driven Engineering (MDE) [1] refers to the systematic use of models as primary artifacts throughout the Software Engineering (SE) development process. A model-driven approach to development is generally based on the Model Driven Architecture (MDA) [8] that is an initiative from OMG [3] specifying a framework of open standards and related technologies. The framework is built upon the metamodel foundation in order to enable a standard specification and inter-operability mechanism for tools. So systems and applications are formalized with metamodel descriptions and are visualized by models as metamodel instantiations. Actual code implementations are created automatically by applying predefined transformations from source models to target models and implementation languages.

In the context of this paper, MDE specifies the organization, coordination and services layers, upon which special purpose tools (editors) are created, allowing the modeling and instantiation of corresponding models. Moreover, constraint rules in the form of OCL expressions are attached on the metamodels to validate models during instantiation as well as transformations are applied to integrate ALIVE across its layers.

3 Motivating Scenario

Incident Management refers to the activities of an organization to identify, analyze and correct hazards. The Netherlands has an extensive crisis management structure to respond to incidents that affect public order. A layered model, the Coordinated Regional Incident-Management Procedure (GRIP), is a nationwide emergency management procedure based on the severity of the disaster, and allows local, regional and national authorities to take action where necessary. The aim of the procedure is to enable the adequate response to the situation, with a minimum level of disruption of the public life. The procedure is currently used by emergency services, different layers of government and government agencies. Table 1 gives an overview of reach and procedures of the different levels.

These GRIP procedures are a motivating use-case for ALIVE as a scenario in which changes in the environment affect organizations [12]. Following the ALIVE method, a distinction is made between organizational structures, specified in the OL (i.e. the structural aspects of the system), and coordination activities, specified in the CL as multi-agent systems (i.e. the behavioral aspects of the system). One aim of ALIVE is to

[3] Object Management Group, see http://www.omg.org/

Table 1. GRIP levels of incident management

Phase	**Affected area**	**Coordination**
GRIP 0	Day-to-day routine operations	-
GRIP 1	Incident of limited proportions	Duty officers
GRIP 2	Incident with a definite effect on the surrounding area	Local commander
GRIP 3	Threatened well-being of (large groups of) the population within a single municipality	Mayor
GRIP 4	Province or country level	Queen's Commissioner

support smooth transition between different disaster levels and maintain consistency of operation, by providing a flexible formal model for the GRIP procedures.

The organizational model describes the abstract purposes of the system. That is, the organizational model describes the objectives and requirements for the organization; it describes the reason for which the organization is created and/or laws or regulations imposed on the organization by external institutions. In the GRIP procedures these can be, for instance, the overall objective to "solve crises" (holds over all possible environments) or abstract maintenance goals like "minimize casualties", "minimize structural damage", and "minimize cost". There are obviously, many different ways in which to specify these issues.
Design decisions taken at OL have consequences for the possibilities at CL. For example, assume that at OL it is explicitly defined that *evacuation of casualties* is a task of the *medics* role and to be done using an *ambulance* by a team of exactly 3 players (the *driver*, the *nurse* and the *first-aid specialist*). This choice gives a very precise description to the CL on how to deploy agents for this task and how to define a concrete evacuation plan. On the other hand, it can be decided at the OL that *evacuation of casualties* is the objective of the crises-solvers. This leaves open the possibility, in case of need, to request a bystander to use her car to take some casualties to the hospital, which may be more efficient if no medics are around. However, such a design decision requires a more complex solution at CL, given that less handles are provide about which are the capabilities and requirements of specific agents, who must be endowed with the capabilities to reason about their position in the organization.

Abstract organization models are flexible and robust as they can accommodate many changes in the environment; however, these models may not give enough information to generate an efficient coordination model. This would be the case of the GRIP 'free' model depicted in Figure 2. Similarly, concrete organization models are more efficient as they describe capabilities and requirements in great detail but will need to be reorganized in face of environmental changes. This implies the specification of a detailed organizational model for each GRIP level, as the models for GRIP-2 and GRIP-3 in Figure 2 plus the reorganization rules to change between these models. The issue is thus to decide when to choose a flexible and robust organizational model, and when to choose a highly descriptive and efficient model.

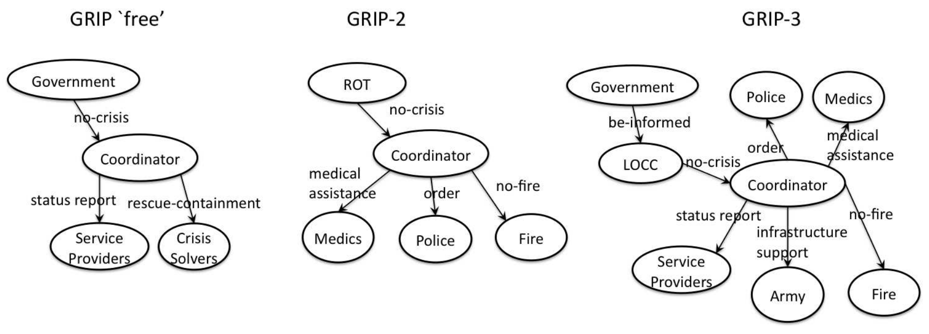


Fig. 2. Possible organizational models at different levels of abstraction, by OperA's Social Structure diagrams.

4 Balance Regulation and Flexibility

As discussed in the previous section, at the highest level of abstraction, organization models represent all institutional norms with the minimum constraints for agents. At the lowest level of abstraction, organization describe one, proven, process that guarantees fully compliance of organizational norms and objectives. In [7] and [11] authors propose a formal description in terms of graph theory for quantifying to what extent organizational structures enjoy specific characteristics such as robustness, flexibility and efficiency. These formal properties can be used to determined the level of specificity of the organizational model. That is, given different organizational models, it can be determined which one provides the highest flexibility, and which one the highest efficiency.

Basically, the spectrum of choices concerning the level of abstraction of the organization model ranges between the following two extremes:

Most Abstract The organization model consists of one only role, which has one objective corresponding to the global objective of the organization, and as norms all the organizational norms. This requires the Coordination Level (CL) to have the capabilities to interpret such description in the context, and decide on the best transformation into concrete plans and activities.

Most Concrete The organization model includes many roles with detailed objectives and sub-objectives. Interaction scenes and the partial ordering of scenes provide all the details needed to generate a complete plan that fixes activity such that there is no room for interpretation. Structures at CL can be obtained by a translation mechanism and therefore these is no need for decision making capabilities at CL.

As detailed by examples in the next sections, different degrees of abstraction reflect also into the single agent autonomy required to accomplish organizational objectives. Intuitively, more detailed specification given at OL, less alternative ways (autonomy) for agents at CL to achieve the organizational objectives. So, despite we often desire agents highly autonomous (e.g., see [4]), we also expect agents to adhere to institutional norms that regiment their activities and, hence, reducing their autonomy. The OperA modeling language adopted at OL does not have the goal of guaranteeing agent autonomy but rather of coping with the given autonomy. Nevertheless, by the separation of concerns long three levels (*organizational*, *coordination* and *service*), the ALIVE's design framework help designers to better regulate agent autonomy simply intervening in the specification of each level. Of course, the autonomy property is also often used as an indicator to establish the system flexibility to adapt to context changes, as detailed next.

5 MDE to guarantee flexibility-regulation balance

This Section describes a concrete MDA approach to support the design of organizational models with different degrees of abstraction as well as the modification of their underlying coordination model as a result of an organizational adaptation. As it is explained, the design property impacts differently the flexibility of run-time systems to cope with context changes. That is, the adaptation process of the run-time system may require changes either within models at CL —e.g., agents' workflows— or within both CL and OL —e.g., agents' workflows and organization's Social Structure. Figure 3 outlines in terms of MDA concepts, the two principal modeling scenarios discussed long the paper.

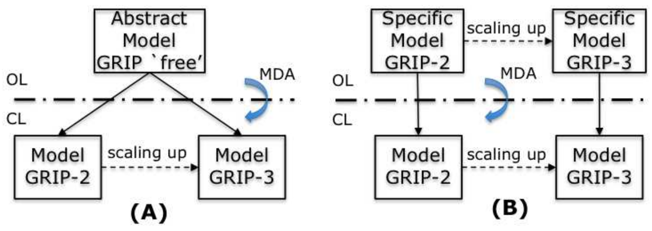


Fig. 3. Modeling alternatives at OL: (A) an abstract model and (B) a detailed model for every crisis situation.

As it is illustrated in the first case (Figure 3.(A)), the abstracted GRIP 'free' organizational model is capable to capture effectively both scenarios of GRIP-2 and GRIP-3 (at OL) with no modifications. As a result this design solution provides ultimate flexibility and dynamicity for the OL. While, the coordination models need to be modified to reflect changes on agents and workflows involved. In this case both GRIP-1 and GRIP-2 coordination models are associated with the same abstracted organizational model making the generation process rather generic, so agents playing the same role are treated similarly ignoring their specific/individual capabilities and requirements.

In the second case, the escalation of scenario from GRIP-2 to GRIP-3 level requires both the modification (with new organizational roles and regulations) of the organizational models and their corresponding coordination models. This more concrete design approach may compromises adaptation flexibility, however it provides more elaborate mappings among the organizational and coordination models, thus providing more elaborate workflows due to enhanced regulations.

In MDA, the OL and CL dependencies are formulated with model driven mappings (relations). The mappings are specified with a transformation language, among the corresponding elements of the OL and CL metamodels shown in Figure 4 and Figure 5 respectively. For example, below, the mappings among `sceneToAgents`, `playerToAgent`, `landmarkToCompositeTask` and `sceneResults` have been specified. So, the transformation process can be initiated from the `sceneToAgents` mapping that creates a number of agents from the players of a scene. The rule in turn applies a mapping among a player and an agent, which based on the interaction patterns identified in the scene for that player creates a number of ordered task lists. A particular ordered task list is derived from the *from* and *to* landmarks that are mapped to composite tasks, and a flow that connects tasks as a result of the partial order is created. Similarly the results (e.g., sub-objectives) of a scene are mapped to workflow tasks.

```
mapping sceneToAgents(in s: OL::Scene, inout wf:Workflow) {
 var agents:= s.players->collect(pl| map
 playerToAgent(agentNames, pl, s));
}
mapping playerToAgent(in agentName:String, in pl:OL::Player, in
 s:OL::Scene, inout wf:Workflow):Agent{
 name := agentName; play := pl.playerID;
 participate += object CL::Partition{
  agent := result;
  task += s.interactionPattern->collect(x|
  map createOrderedTaskLists(x, wf)); };
}
mapping createOrderedTaskLists(in lp: OL::LandmarkPattern,
 inout wf:Workflow):Sequence(CompositeTask){
 var partialOrders := lp.landmarkOrder;
 result := partialOrders->collect(x:OL::PartialOrder|
 map createOrderedTaskList(x, wf));
}
mapping createOrderedTaskList(in po:OL::PartialOrder,
 inout wf:Workflow):Sequence(CompositeTask){
 var ct1 := map landmarkToCompositeTask(po.from);
 var ct2 := map landmarkToCompositeTask(po.to);
 var flow := object Flow{};
 ct1.outgoing->append(flow); ct2.incoming->append(flow);
 wf.edge +=flow; result +=ct1; result +=ct2;
}
mapping landmarkToCompositeTask(in l:OL::Landmark):CompositeTask{
 result := object CL::CompositeTask{
 name := l.name;};
...}
mapping sceneResults(in s:OL::Scene, inout wf:Workflow):
 Sequence(SimpleTask){
 result+= s.results->collect(x| x.stateDescription->
 map toSimpleTask());
...}
mapping OL::PartialStateDescription:: toSimpleTask():SimpleTask{
 name:= self.stateDescription;
...}
```

When an abstracted organizational model is used as a source of the mapping definitions only few elements are resolved, associated with general capabilities. In the second case where the source model is more elaborate the mapping can produce more detailed `Agent` specifications with specific capabilities.

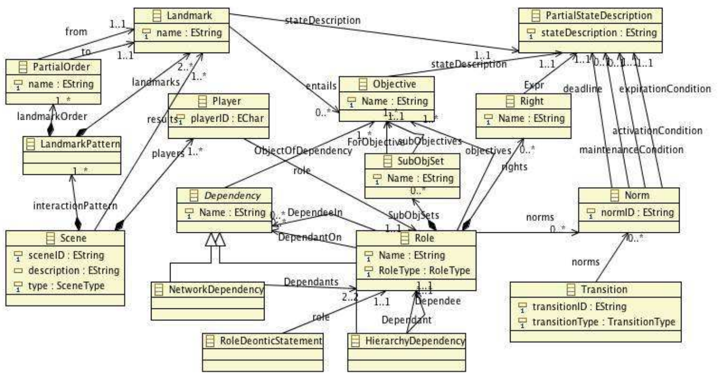


Fig. 4. OperA Meta-Model fragment.

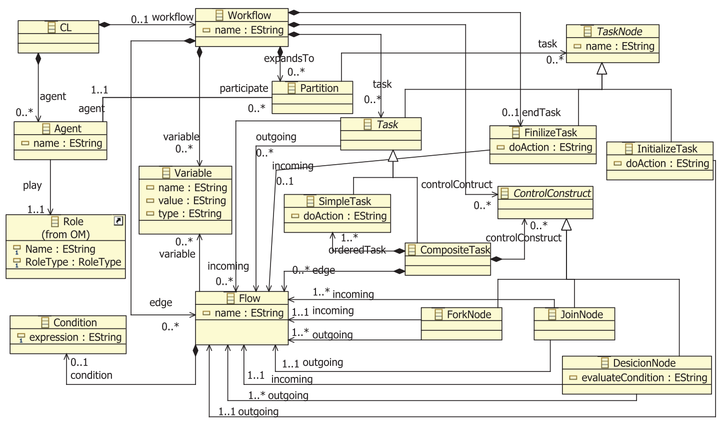
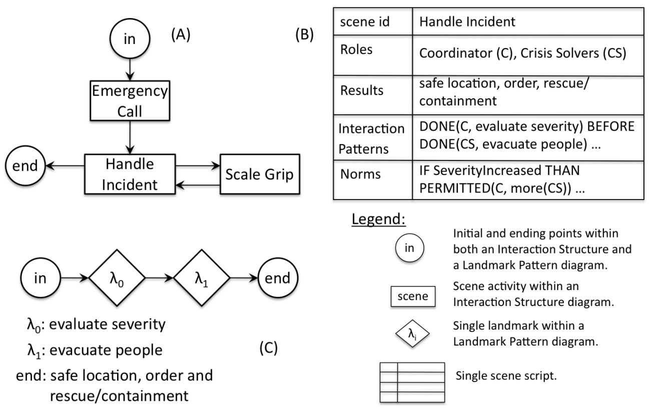


Fig. 5. Coordination Meta-Model fragment.

6 Organizational Structure Design

This section provides some details about how to use the modeling language of our ALIVE's design famework. According to Figure 3, the focus is given on how context changes may affect the whole system model, considering both a more and a less abstract organizational model at OL. The features of the ALIVE's design framework are illustrated over the crisis scenario by some examples of MDA transformation rules.

6.1 Abstract OL


in
(A)
Emergency Call
Handle Incident
Scale Grip
end
(B)
in
λ0
λ1
end
λ0: evaluate severity
λ1: evacuate people
end: safe location, order and rescue/containment
(C)
Legend:
in
Initial and ending points within both an Interaction Structure and a Landmark Pattern diagram.
scene
Scene activity within an Interaction Structure diagram.
λi
Single landmark within a Landmark Pattern diagram.
Single scene script.

scene id	Handle Incident
Roles	Coordinator (C), Crisis Solvers (CS)
Results	safe location, order, rescue/ containment
Interaction Patterns	DONE(C, evaluate severity) BEFORE DONE(CS, evacuate people) …
Norms	IF SeverityIncreased THAN PERMITTED(C, more(CS)) …

Fig. 6. Modeling fragments at OL: (A) Interaction Structure, (B) scene script for `Handle Incident` and its (C) landmark pattern, referred to the *GRIP 'free'* model of Figure 2.

This is the case where at OL the designer provides an organizational model that abstracts from the GRIP crisis situations, i.e., as showed by *GRIP 'free'* model of Figure 2. The abstraction level adopted for dealing with the social structure of the organization also reflects into its interaction structure, as pointed out by Figure 6. Figure 6.(A) defines a partial ordering of activities (scenes) within the organization to fulfill the main objective `no-crisis`. Moreover, within each scene, the designer may describe how roles interact and evolve, expected results and norms regulating the interactions, as showed in Figure 6.(B) and (C).

In the abstract OL, the landmark pattern provided for the scene `Handle Incident` only specifies two important objectives (landmark states) to be achieved by role enacting agents within the scene execution. In particular, each agent that enacts role `Coordinator` has at least to reach the state `evaluate severity` and each agent that enacts role `Crisis Solvers` has at least to reach the state `evacuate people` in order to correctly fulfill the scene results. This level of details are quite far from what actually agents require at CL to effectively deal with a `Handle Incident` activity (a complex task) within a crisis scenario. For example, Figure 7 gives an idea of how complex may be the workflow of agent tasks [4] at CL to deal with the expected results from scene `Handle Incident`, i.e., safe location, street order, rescue and containment.

[4] This design concept 'task' is used to model both an agent's complex plan and an agent's simple atomic action, as for the scope of this paper the distinction is not relevant.

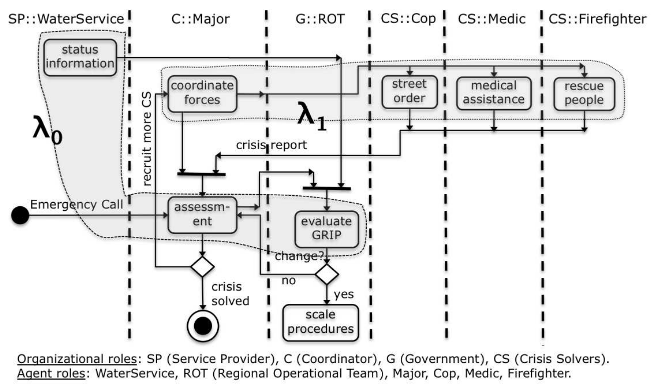


Fig. 7. Possible workflow model at CL to deal with the scene `Handle Incident` of Figure 6 when severity is at GRIP-2.

Thus if we apply the transformation description (`sceneToAgents`) upon the organizational model, an incomplete coordination model is derived. Therefore, even if some OL concepts can be straight mapped into CL, lot of still lacking information has to be captured within CL, e.g., by providing agents that embed more complex activities than required. That is, more abstract the OL specification, more freedom to the agents at CL to choose the way to fulfill organizational objectives. For coping with the crisis situation of GRIP-2, a possible workflow required at CL —mainly to achieve the results required by scene `Handle Incident`— is described by Figure 7. Worth noticing that the single landmark `evacuate people` —λ_0: specified at OL— is mapped into three main CL workflow tasks —`status information`, `assessment` and `evaluate GRIP`— belonging to roles `Service Providers`, `Coordinator` and `Government` respectively, reflecting the more complex level of details as well as the more freedom given to the agents in order to achieve an objective.

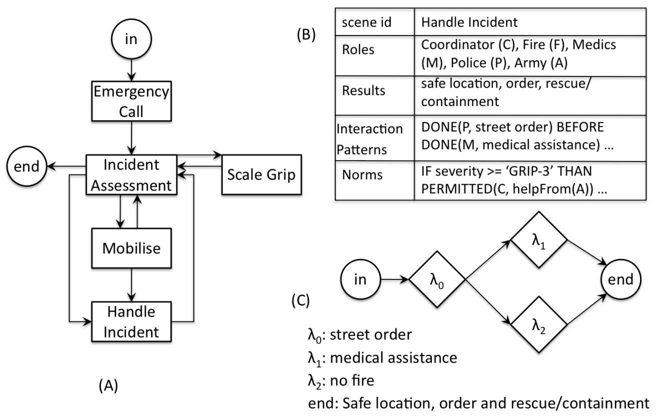
In the case of a transformation, the landmark `evacuate people` produces a composite task with corresponding agents, which are derived from the corresponding `players` of the `scene`. However, the composite task cannot be decomposed further into two simple tasks as such details are not provided yet. Moreover, in GRIP-2, several agents' expertise are needed to cope with the crisis, e.g., agents `Cop`, `Medic` and `Firefighter` comply with and enact the role `Crisis Solvers` and how them should coordinate each other is not detailed within the landmark patterns of scenes. So, workflow constructs such as `ForkNode` and `JoinNode` can not be created to impose the ordering of single tasks that are created from the result mapping (`sceneResults`). According to our crisis management scenario, we describe here a possible context change that forces the whole ALIVE's organizational setting to adapt. In particular, we discuss how these changes differently impact design models provided at CL and OL as illustrated in Figure 3.(A).

Scenario 1 *While the crisis organization is involved in a GRIP-2 level of a floodwater incident, the river that crosses a big city has breached its banks. This causes the roads into (or out of) the incident location to get blocked and cannot be used for wounded and* *people evacuation any longer. This context change also forces the Government to scale up to the GRIP-3 level.*

The regulation imposed by the organizational model *GRIP 'free'* —i.e., see Figure 2 and Figure 6— remains still valid to cope with Scenario 1. Nevertheless, as partially described by Figure 7, this workflow is not suitable anymore for the new crisis situation and a new workflow model is needed at CL. For example, new agents playing roles along with new rights come to play at GRIP-3 of incident handling, e.g., as the mayors of the affected towns are coordinated and supervised by a regional policy team (LOCC) that keeps informed the Minister of Internal Affairs (Government) about the actual crises situation, i.e., see the social structure diagram of Figure 2.(GRIP-3). In this case, the transformation process (`playerToAgent`) is parameterized from domain knowledge as follows: `playerToAgent('Major', C)` and `playerToAgent('LOCC', C)`.

Finally, with MDA many of the normative rules defined in OL models can be translated into OCL rules in CL models, so the generated coordination models can be validated after a transformation or an adaptation process.

6.2 Specific OL



scene id	Handle Incident
Roles	Coordinator (C), Fire (F), Medics (M), Police (P), Army (A)
Results	safe location, order, rescue/ containment
Interaction Patterns	DONE(P, street order) BEFORE DONE(M, medical assistance) ...
Norms	IF severity >= 'GRIP-3' THAN PERMITTED(C, helpFrom(A)) ...

Fig. 8. Modeling fragments at OL: (A) Interaction Structure, (B) scene script for `Handle Incident` and its (C) landmark pattern, referred to the *GRIP-3* model of Figure 2.

In this Section, we discuss the case where at OL the designer provides an organizational model specifically tailored to a crises situation, e.g., let us consider the case of GRIP-2. A possible resulting model is defined by the social structure of Figure 2.(GRIP-2) and by the interaction structure of Figure 8 but except for the scene `Mobilise` along with any reference to the role `Army`, as it comes to play within the crisis only at GRIP-3 and GRIP-4.

The more accurate level of details also reflects in landmark patterns of scenes, as partially showed in Figure 8.(C). That is, a MDA transformation can produce the

workflow of Figure 7 as almost a straight interpretation of the OL model. For example, the previous freedom at CL of choosing the three agents —`Cop`, `Medic` and `Firefighter`— to deal with `Handle Incident` activities at GRIP-2 is lost. In fact, now by the landmark pattern of Figure 8.(C) the need of such three agents along their required abilities is precisely defined by:
$DONE(P, street\_order) BEFORE$
$(DONE(M, medical\_assistance)\ AND\ DONE(F, no\_fire))$

So, the transformation of scene results (`sceneResults`) will produce a detailed workflow fragment of tasks with the imposed ordering and concurrency derived from the landmark pattern. Moreover, instead of the previous generic role of `Crisis Solvers` (CS) played by three different agents —`CS::Cop`, `CS::Medic` and `CS::Firefighter`— now the workflow is more elaborated with `P::Cop`, `M::Medic` and `F::Firefighter` where `P`, `M` and `F` correspond to well defined organizational roles `Police`, `Medics` and `Firefighter` respectively, along with their required abilities and rights.

Considering the previous Scenario 1, the context change forces the organization to adapt but this time the organizational model proposed in Figure 2.(GRIP-2) is not valid anymore to deal with GRIP-3. The change from GRIP-2 to GRIP-3 introduces some new organizational roles and changes in the hierarchical structure of the role interactions. For example, the National Operational Coordination Center (`LOCC`) of the Dutch Ministry of Interior and Kingdom Relations comes to play with the responsibility to monitor and support the activities of the pool of majors involved into the crisis. That is, the proposed model for GRIP-3 (see Figure 2) considers a local dimension for the `Coordinator` that is enacted by any involved major along with the required local handling forces. Moreover, within the organizational model of GRIP-2 there is not any role (e.g., `Crisis Solvers`) that may abstract on possible `Army` enacting agents needed at GRIP-3.

7 Consequences for Agent Design

Role description as provided by the OL modeling is a "position" to be filled by a player [10]. At CL, roles will be (temporarily) assigned to players. For example, in GRIP 1, the role of crisis coordinator is assigned to the first official to arrive on the scene. A role may be temporarily unassigned, without necessarily leading to a failure in the operation of the system. From the perspective of the organization it does not matter whether policewoman Alice or policeman Bob take the coordinator role, as long as they both have sufficient capability. The ability to dynamically bind different players to roles gives the organization a degree of adaptability in meeting changing goals and environments.

A role player who is bound to a role in an organizational structure needs an ability to perform the assigned role. This capabilities include [2]:

- Execution of function defined by the role or imposed by role relationships, including the ability to use resources available to the role.
- Ability to communicate, as a proxy for its role, with players of other roles.

– Ability to reason about what plans and activities can be used to achieve role objectives (landmark states).

In particular this last capability requires different skills, depending on the level of abstraction of the OL specification. From the perspective of a role player, the role description provides a, more or less abstract, definition of the organizational knowledge and skills required to adequately perform the role. That is, different levels of abstraction of OL models have consequences for the capabilities required from CL agents, i.e., in its operational knowledge or know-how. Role players need evaluate context and determine fulfillment of the role. This demands observation and reasoning skills for the agents in order to adapt to different contexts and determine plans and operational activities to enact role. Communication between agents is necessary, and robustness of the system is rather dependent on the monitoring process. Nevertheless, when agents are designed for a specific role, they cannot accommodate large changes in the context.

Taking the crisis management scenario as an example, we can identify the following characteristics of the different abstraction choices:

Case 1 Abstract emergency management description at OL.
In this situation, the crisis management model at OL is abstract enough to incorporate all GRIP situations. Specific models for each GRIP level need to be defined at CL, but they still have to adhere to those regulations inherited from OL by MDA transformations.
- Advantages: Only requires organizational verification for one model, MDA model transformations guarantee link between OL and CL. Furthermore, dynamic changes are dealt with at CL affecting only the holding GRIP model, enabling a stable OL model. Less regulations inherited from OL allows agents for a more freedom to search for optimized ways of achieving organizational objectives.
- Disadvantages: There is a need for CL-designers to be able to acquire the lacking OL specification, e.g., manually and/or by other design tools. Hence, the CL-designer has also to be a domain expert in order to correlate agent capabilities with organizational objectives and norms.

Case 2 Specific GRIP descriptions at OL.
- Advantages: The verification of each GRIP is done at OL level (e.g., making use of OperettA verification capabilities [6]). The CL model is explicitly generated for each GRIP, limiting the need of acquiring context information at CL.
- Disadvantages: Change dynamics are required at OL which implies a more complex change strategy, as changes at OL have consequences for all models at CL. Interpretation of GRIP at CL is fixed by the OL-CL transformation, which means that it is not possible to incorporate specific agent capabilities.

Another important issue is the evolution of situation knowledge. If we keep the knowledge coupled to a role, then it can be transferred to any agent enacting that role (e.g. when the role of coordinator moves from police officer to mayor). If we keep knowledge at agent level, then explicit ‘debriefing’ is needed when roles change. Comprehensive solutions for this issue require complex agents that are able to reason about their own objectives and desires and thus decide and negotiate their participation in an organization [3].

8 Conclusions

The proposed work shows features and discuss the applicability of a design framework for software and service engineering of systems deployed in highly dynamic contexts. We illustrated that the modularity (levels: OL, CL and SL) of our design framework helps designers in separation of concerns, making easier and more precise to regulate agent autonomy by simply intervening in the specification of each level. Moreover, we focused on how context changes may affect the system flexibility to adapt, considering both a more and a less abstract organizational model at OL. To deliver on this aim, we adopt and combine the OperA methodology to deal with the organizational model specification, and the MDA transformation techniques to map concepts between the different levels of abstraction at OL and CL. We motivate and actually apply our ideas within the case of modeling of crisis management systems in the Netherlands.

References

1. A. W. Brown, J. Conallen, and D. Tropeano. *Introduction: Models, Modelling and Model Driven Architecture (MDA).* Springer, 2005.
2. A. Colman and J. Han. Roles, players and adaptive organisations. *Applied Ontology: An Inter. Journal of Ontological Analysis and Conceptual Modeling*, (2):105–126, 2007.
3. M. Dastani, V. Dignum, and F. Dignum. Role assignment in open agent societies. In *AAMAS03*. ACM Press, July 2003.
4. M. Dignum and H. Weigand. I am autonomous, you are autonomous. *Agents and Computational Autonomy, Springer*, pages 227–236, 2004.
5. V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. PhD thesis, Universiteit Utrecht, 2004.
6. V. Dignum and D. Okouya. Operetta: A prototype tool for the design, analysis and development of multi-agent organizations. In *Proc. AAMAS-08 (Demo Track)*, 2008.
7. D. Grossi, F. Dignum, V. Dignum, M. Dastani, and L. Royakkers. Structural aspects of the evaluation of agent organizations. In *COIN@ECAI 2006*, 2006.
8. A. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture—Practice and Promise*. Addison-Wesley, 2003.
9. T. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1), March 1994.
10. J. Odell, M. Nodine, and R. Levy. A metamodel for agents, roles, and groups. In J. Odell, P. Giorgini, and J. Mller, editors, *AOSE IV*, LNCS, forthcoming. Springer, 2005.
11. L. Penserini, D. Grossi, F. Dignum, V. Dignum, and H. Aldewereld. Evaluating organizational configurations. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2009)*, 2009.
12. T. B. Quillinan, F. Brazier, H. Aldewereld, F. Dignum, V. Dignum, L. Penserini, and N. Wijngaards. Developing Agent-based Organizational Models for Crisis Management. In *Proc. of the 8th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2009)*, pages 45–51. ACM Press, 2009.