Dynamics Within an Organisation:
Temporal Specification, Simulation and Evaluation

Catholijn M. Jonker[1], Jan Treur[1] and Wouter C.A. Wijngaards[1]

[1]*Department of Artificial Intelligence, Vrije Universiteit Amsterdam,*
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: <{jonker,treur,wouterw}@cs.vu.nl> URL: http://www.cs.vu.nl/{~jonker,~treur,~wouterw}

A crucial aspect of a multi-agent organisation is its dynamics. In this paper different types of specifications of properties of the dynamics within an organisation are introduced. Supporting tools for specification, simulation and analysis of dynamics within multi-agent organisations have been implemented.

1. INTRODUCTION

Multi-agent systems often have complex dynamics, both in human society and in the non-human case. Organisational structure is used as a means to handle these complex dynamics. It provides a structuring of the processes in such a manner that an agent involved can function in a more appropriate manner. For example, at least partly the behavioural alternatives for the other agents are known. Put differently, the flow of dynamics within a given organisational structure is much more predictable than in an entirely unstructured situation. This assumes that the organisational structure itself is relatively stable, i.e., the structure may change, but the frequency and scale of change are assumed low compared to the more standard flow of dynamics through the structure. Both types of dynamics, dynamics *within* an organisational structure, and dynamics *of* an organisational structure are quite relevant to the area of organisation modelling. In this paper, for reasons of focussing the former type of dynamics is addressed.

Models for the dynamics within an organisation can be specified to be used as a basis for simulation, also called *executable models*. These types of models can be used to perform (pseudo-)experiments. A language for executable organisation models should be formal, and as simple as possible, to avoid computational complexity. Expressivity can be limited. Software tools to support such a language serve as *simulation environment*.

A language to specify and analyse dynamic properties of the flow within an organisation, on the other hand, should be sufficiently expressive; executability, however, is not required for such a language. What is important, though, is that properties specified in such a language can be checked for a given sample behaviour (e.g., a simulation run) without much work, preferably in an automated manner. Moreover, it is useful if a language to specify properties provides possibilities for further analysis of logical relationships between properties, and to generate theories about organisation dynamics. For these reasons also a language to specify properties of the dynamics within an organisation should be formal, and at least partly supported by software tools (*analysis environment*).

In this paper, for the Agent-Group-Role (AGR) organisation modelling approach introduced in (Ferber and Gutknecht, 1998), two temporal specification languages are put forward and illustrated for an example organisation. In Section 2 the static

and dynamic view of the AGR modelling approach is briefly introduced. In Section 3 it is shown how languages to model dynamics within such a model can be defined. Section 4 discusses the example and the simulation environment.

2. ORGANISATION MODELLING AND BEHAVIOUR

To model an organisation, the Agent/Group/Role (AGR) approach, adopted from (Ferber and Gutknecht, 1998) is used. The *organisational structure* is the specification of a specific multi-agent organisation based on a definition of groups, roles and their relationships within the organisation. A central notion is the *group structure*. It identifies the roles and (intragroup) interaction between roles within a group. The group structure is defined by the set of roles, interaction schemes and an (intragroup) transfer or communication model within the group. In addition, (intergroup) role relations between roles of different groups specify the connectivity of groups within an organisation.

To be able to simulate or analyse dynamics within an organisation, in addition to the static organisation structure specification discussed above, as part of an organisation model also a specification of *dynamics within the organisation* is needed. To this end, in the specification of an organisation model the following types of specifications of *dynamic properties* are distinguished: *single role behaviour properties, intragroup interaction properties, intragroup transfer properties, intergroup interaction properties, global group properties* and *global organisation properties*. These properties serve as constraints on the dynamics of the respective roles and interactions.

Within this paper examples are taken from a case study that has been performed in the context of the Rabobank, one of the largest banks in the Netherlands, see (Brazier et al., 1999). The case study addressed design and analysis of a multi-agent approach for a bank service provision problem using a Call Centre.

3. TEMPORAL SPECIFICATION OF DYNAMICS WITHIN AN ORGANISATION

During the development of an organisation model, means to model dynamics should also play an important role. In Section 3.1 a language, TTL, is presented that is suitable for the specification and analysis of dynamical organisational properties. In order to simulate dynamics, in addition an executable temporal language is introduced in Section 3.2.

3.1. The Language to Specify Dynamic Properties

To specify properties on the dynamics within the organisation, the temporal trace language used in (Jonker and Treur, 1998; Herlea, Jonker, Treur, and Wijngaards, 1999) is adopted. States of a trace can be related to state properties via the formally defined satisfaction relation $\models$ between states and formulae. Comparable to the approach in situation calculus, the order sorted predicate logic *temporal trace language* TTL is built on atoms referring to traces, time and state properties, such as state(M , t, output(R)) $\models$ p.

As an example, a dynamic property for the dynamics within the organisation as a whole is shown. The organisation in question consists of an open group (for interaction) with clients and distributor groups. The distributor groups divide the workload over the organisation. To be able to specify ongoing interaction between two roles for which multiple appearances exist, the notion of *role instance* is used. This notion abstracts from the agent realising the role as actor, but enables to distinguish between appearances of roles. The example property concerns the open group and its two roles: the client role and the receptionist role.

GR1 All requests answered

This global organisation property specifies that at any point in time, if a client communicates a request to the receptionist, then at some later time point the receptionist will communicate either an acceptance or a rejection of the request to this client.

$\forall$ M : TRACES $\forall$ tid : TaskId, $\forall$ t1, tf : T $\forall$ C: CLIENT:open_group
$\forall$ R: RECEPTIONIST: open_group: [state(M , t1, output(C)) $\models$
comm_from_to(requested(tid, tf), C, R) $\Rightarrow$ $\exists$ t2 : T [t2 $\geq$ t1 &
[state(M , t2, input(C)) $\models$ comm_from_to(rejected(tid), R, C) $\vee$
state(M , t2, input(C)) $\models$ comm_from_to(accepted(tid), R, C)]]

3.2. The Executable Language to Specify Simulation Models

To obtain an executable language, in comparison with the very expressive temporal trace language discussed above strong constraints are imposed on what can be expressed. These constraints define a temporal language within the paradigm of executable temporal logic; cf. (Barringer et al., 1996). Roughly spoken, in this executable language it can only be expressed that

if a certain state property α on some part of the organisation holds for a certain time interval, then after some delay another state property β on a part of the organisation should hold for a certain time interval.

This specific temporal relationship •—» (*leads to*) is definable within the temporal trace language TTL. Only role behaviour, intergroup role interaction and transfer properties need to be specified in the executable language. The other types of properties are emergent in the simulation process.

Input for the simulation environment is a set of executable temporal formulae expressed in terms of the leads to relation •—», i.e., in the format defined above. Thus, the example executable organisation model is expressed in terms of such formulae. Examples taken from this example specification are:

IrRI1 Receptionist-Distributor Intergroup Role Interaction

This property expresses that if the receptionist of the open (or Client Service) group instance receives a request from a client, then the distributor role instance of cc the group instance of the distribution group forwards this request to the participants in his group.

$\forall$ tid : TASKID, $\forall$ tf : COMPLETIONTIME, $\forall$ P : PARTICIPANT:cc:DISTRIBUTION
$\forall$ R : RECEPTIONIST:open_group:OPEN_GROUP, $\forall$ C : CLIENT:open_group:OPEN_GROUP,
$\forall$ D : DISTRIBUTOR:cc:DISTRIBUTION, $\forall$ r : REGION,
[INTERGROUP_ROLE_RELATION(R, D) & CLIENT_REGION_RELATION(C, r) &
REGION_BANK_RELATION(r, P)] $\Rightarrow$ [input(R):comm_from_to(requested(tid, tf), C, R)
        •—»$_{5,5,10,10}$ output(D):comm_from_to(requested(tid, tf), D, P)]

IrRI2 Distributor-Receptionist Intergroup Role Interaction

Property IrRI2 expresses that any information regarding the request of a client that the distributor instance of the distribution group instance cc receives is forwarded to the client by the receptionist role instance of the client server group instance (also called open group). In the example, for reasons of presentation we assume that only one client exists. If more clients are handled at the same time, an additional condition is needed to guarantee that the right client is notified.

$\forall$ tid : TASKID, $\forall$ R : RECEPTIONIST:open_group:OPEN_GROUP,
$\forall$ C : CLIENT:open_group:OPEN_GROUP, $\forall$ info : TASKINFORMATION
$\forall$ D : DISTRIBUTOR:cc:DISTRIBUTION, $\forall$ P : PARTICIPANT:cc:DISTRIBUTION,
[INTERGROUP_ROLE_RELATION(R, D)] $\Rightarrow$ [input(D):comm_from_to(info(tid), P, D)
      $\bullet\longrightarrow\!\!\gg_{5,5,10,10}$ output(R):comm_from_to(info(tid), R, C)]

IrRI3 Participant-Distributor Intergroup Role Interaction

Property IrRI3 expresses that the Distributor of a local bank group forwards requests to the Participants of that local bank group.

$\forall$ tid : TASKID, $\forall$ tf : COMPLETIONTIME, $\forall$ D1 : DISTRIBUTOR:cc:DISTRIBUTION,
$\forall$ P1 : PARTICIPANT:cc:DISTRIBUTION, $\forall$ GI : DISTRIBUTION, $\forall$ D2 : DISTRIBUTOR:GI:DISTRIBUTION,
$\forall$ P2 : PARTICIPANT:GI:DISTRIBUTION: [GI $\neq$ cc & INTERGROUP_ROLE_RELATION(P1, D2)] $\Rightarrow$
[input(P1):comm_from_to(requested(tid, tf), D1, P1)
      $\bullet\longrightarrow\!\!\gg_{0.5,0.5,1,1}$ output(D2):comm_from_to(requested(tid, tf), D2, P2)]

TR7 Workmanager-Employee communication for assignments

If the Distributor of a local bank group instance communicates to a Participant of the local bank group instance that he is assigned some task, then this communication will be received by that Participant of the local bank group instance some time later.

$\forall$ tid : TASKID, $\forall$ GI : DISTRIBUTION, $\forall$ D : DISTRIBUTOR:GI:DISTRIBUTION,
$\forall$ P : PARTICIPANT:GI:DISTRIBUTION [GI $\neq$ cc $\Rightarrow$ [output(D):comm_from_to(assigned(tid), P2, D2)
      $\bullet\longrightarrow\!\!\gg_{5,5,1,1}$ input(P):comm_from_to(assigned(tid), P1, D1)]]

4. SOFTWARE ENVIRONMENT

A software environment has been created to enable the analysis and simulation of the dynamics within organisation models with all the usual benefits of rapid prototyping. First, Following the paradigm of executable temporal logic, cf. (Barringer et al., 1996), a 8000 line simulation program was written in C++ to automatically generate the consequences of the temporal relationships within the executable organisation specification. The program is a special purpose tool to derive the results reasoning forwards in time, as in executable temporal logic.

Second, an analysis environment with three tools has been created. A Prolog programme of about 500 lines checks if a given behavioural property is fulfilled in a given trace of set of traces. Another program, of about 4000 lines C++, takes an existing trace of behaviour and a set of temporal relationships and checks which temporal relationships hold in the trace. It will mark unexpected and expected but absent deficiencies in the trace. A third tool of about 300 lines of Prolog, is a dedicated prover of properties for all traces of an (executable) specification.

5. DISCUSSION

In this paper specification and uses of models of the dynamics within a multi-agent organisation are addressed. A declarative temporal language is proposed as a basis for simulation. This language belongs to the class of executable temporal languages; cf. (Barringer et al., 1996). Models can be specified in a declarative manner based on a temporal 'leads to' relation; within the simulation environment these models can be executed. Moreover, to specify dynamic properties another language is put forward: a temporal trace language that belongs to the family of languages to which also situation calculus (McCarthy and P. Hayes, 1969), event calculus (Kowalski and Sergot, 1986), and fluent calculus (Hölldobler and Thielscher, 1990) belong. The executable language for simulations is definable within this much more expressive language.

Supporting tools have been implemented; a software environment for simulation of a multi-agent organisation model and a software environment for analysis of dynamic properties against traces for such a model. The analysis environment, see the full paper (Jonker, Treur & Wijngaards, 2001), includes three different tools, briefly described in Section 4. These tools assume a finite time frame. A simple example organisation model illustrates the benefit of the language and of the software environment for organisational modelling.

REFERENCES

Barringer, H., M. Fisher, D. Gabbay, R. Owens, & M. Reynolds (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons.

Brazier, F. M. T., Jonker, C. M., Jüngen, F. J., and Treur, J. (1999). Distributed Scheduling to Support a Call Centre: a Co-operative Multi-Agent Approach. In: *Applied Artificial Intelligence Journal*, vol. 13, pp. 65-90. H. S. Nwana and D. T. Ndumu (eds.), Special Issue on Multi-Agent Systems.

Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organisations in multi-agent systems. In: *Proc. of the Third International Conference on Multi-Agent Systems* (ICMAS '98). IEEE Computer Society, 1998

Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E. (1999). Specification of Behavioural Requirements within Compositional Multi-Agent System Design. In: F.J. Garijo, M. Boman (eds.), *Multi-Agent System Engineering, Proc. of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, MAAMAW'99. Lecture Notes in AI, vol. 1647, Springer Verlag, 1999, pp. 8-27.

Hölldobler, S., and Thielscher, M. (1990). A new deductive approach to planning. *New Generation Computing*, 8:225-244, 1990.

Jonker, C.M., and Treur, J. (1998). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380. Extended version in: *International Journal of Cooperative Information Systems.* To appear.

Jonker, C.M, Treur, J and Wijngaards, W.C.A. (2001). Dynamics Within an Organisation: Temporal Specification, Simulation and Evaluation. Report. Vrije Universiteit Amsterdam, Department of Artificial Intelligence.

Kowalski, R., and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67-95, 1986.

McCarthy, J. and P. Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463--502, 1969.