Chapter 5
A Formal Approach to Building Compositional Agent-Based Simulations

Catholijn M. Jonker and Jan Treur

Why Read This Chapter? To be introduced to a more formal "computer-science" style of simulation design, especially suited to simulations of multi-level systems (e.g. firms, departments, and people).

Abstract This chapter is an introduction to a more formal approach to designing agent-based simulations of organisations (in the widest sense). The basic method is the iterative refinement of structure, process and knowledge, decomposing each abstraction into near-decomposable components that can be (for the most part) then considered separately. Within this over all framework there are two complementary approaches: designing the organisation first, and designing the individual agents first.

5.1 Introduction

This chapter outlines a more formal approach to designing an agent system, in this case an agent-based simulation. Its approach comes from computer science, and shows how one can develop a design for a simulation model in a staged and cautious manner. This is particularly appropriate when the simulation is complex, requiring the interaction of complex and intelligent entities, and the intended design of the model is essentially known or accessible. Thus it contrasts with and complements the previous chapter describing more informal and exploratory approaches to developing simulations (Norling et al. 2013).

C.M. Jonker (✉)
Man-Machine Interaction, Delft University of Technology, Mekelweg 4, Delft 2628 CD, The Netherlands
e-mail: C.M.Jonker@tudelft.nl

J. Treur
Department of Artificial Intelligence, Vrije Universiteit Amsterdam, De Boelelaan 1081a, Amsterdam 1081 HV, The Netherlands
e-mail: treur@few.vu.nl

B. Edmonds and R. Meyer (eds.), *Simulating Social Complexity*, Understanding Complex Systems, DOI 10.1007/978-3-540-93813-2_5,

This chapter draws on approaches which were designed for engineering agent-based software systems (e.g. air traffic control) and applies them to the engineering of agent-based simulations since a simulation model *is* an example of a complex piece of software. Such a cautious approach will result in the development of the model taking more time and effort but also should result in a simulation model that is easier to maintain and adapt and create fewer bugs, or simulations artefacts (as described in Chap. 6, Galán et al. 2013).

The chapter is structured into three main sections, covering: compositional design, organisations and agents. The section on compositional design is the most general, and "works" for both the design of organisations and agents. However, there is no need to reinvent the wheel – the sections on designing organisations and agents raise more specific issues and aspects that experience has shown to be helpful when designing simulations of these entities. Although they do not recapitulate all in the section on compositional design they are essentially examples of that general approach, though they each only concentrate on part of the overall process. The differences are, roughly, that the organisational view starts with the organisation, then the roles within that organisation, working "downwards". It highlights the constraints from above that organisations place on their members, dealing with the nature of the agents after. The agent viewpoint starts with the agents and their properties first and then moves on to how they interact (including maybe as an organisation). Social phenomena often *do* involve both the "downward" actions of constraint and "immergence" as well as the upwards actions of emergence and collective outcomes. Thus both organisational and agent views are often necessary to be explicitly considered in simulations.

5.2 Principles of Compositional Design of Multi-agent Systems

Although any principled design method can be used for the development of simulations, we are concentrating in this chapter on those social systems which are compositional in nature, since compositional actors are common in the social world and this case also covers the principles in the simpler, non-compositional, case. The principles described are quite general in nature, but will be illustrated with respect to a particular compositional multi-agent development method, the *Design and Specification of Interacting Reasoning Components (DESIRE)* (Brazier et al. 1998).

The approach described here considers the design of autonomous interactive agents and explicitly models both the *intra-agent functionality* and the *inter-agent functionality*. Intra-agent functionality concerns the expertise required to perform the tasks for which an agent is responsible in terms of the knowledge, and reasoning and acting capabilities. The inter-agent functionality concerns the expertise required to perform and guide co-ordination, co-operation and other forms of social interaction in terms of knowledge, reasoning and acting capabilities.

In the approach here described, both the individual agents and the overall system are considered as compositional structures – hence all functionality is designed in terms of interacting, compositionally structured components. Complex distributed processes are the result of tasks performed by agents in interaction with their environment.

The process starts at the most general, aggregate and abstract level, works in ever increasing detail down through the specification of individual agents and finally the code that determines their behaviour. At each stage there are a number of decisions to be made, which correspond to specifying the features and properties that are described below. The decisions at the higher levels help frame those made for the next level, etc. until everything has been described. This should leave a documented "trace" of the decisions that are made during the simulation design, that will help with the description of the simulation and the explicit logging of assumptions made during the design process.

Once this has been done and a roughly working simulation obtained, the verification and validation procedure happens in the other direction, starting at testing and validating the smallest processes and components and building upwards to higher groups and units until the simulation as a whole has been checked.

5.2.1 The Design Process

The design of a multi-agent simulation is an iterative process, which aims at the identification of the parties involved (i.e., human agents, system agents, external worlds), and the processes, in addition to the types of knowledge needed. Initially broad conceptual descriptions of specific processes and knowledge are attained. Further explication of these conceptual design descriptions results in more detailed design descriptions, most often in parallel with the development of the conceptual design. During the design of these models, prototype implementations or parts or sections of the overall model may be used to analyse or verify the resulting behaviour. On the basis of examination of these partial prototypes, new designs and prototypes are generated and examined, and so on and so forth. This *evolutionary development* of systems is characteristic to the whole approach. Thus, as with any idealised design methodology, in practice there is a lot of iterating back and forth between stages and levels until a satisfactory design is obtained. The concepts presented in this chapter are to help focus what might otherwise be an unstructured process.

We distinguish the following kinds of descriptions within the development process:

- Problem description – Sect. 5.2.3
- Conceptual design – Sect. 5.2.4
- Detailed design – Sect. 5.2.4
- Design rationale – Sect. 5.2.5
- Operational design

The *problem description* is a description of the target system to be simulated and the main issues and questions to be investigated, this usually includes the *requirements* imposed on the design – what the simulation must do to be useful in this regard. Starting from the problem description, the *design rationale* specifies the choices made during each of the levels of the design process, the reasons for those choices, and the assumptions behind those choices that will impinge on its use. In other words, the *design rationale* is the strategy for "solving" the *problem description* using the design along with the reasons and justification for that strategy.

The actual design process roughly proceeds from conceptual and abstract, down to the more concrete until one has almost written the simulation code itself. The *conceptual design* includes conceptual models (the main design ideas and structures) for each entity in the model: the organisation, its roles and groups, the individual agents, the external world, the interaction between agents, and the interaction between agents and the external world. In a sense a conceptual design could apply as much to a human who will have to fulfil a role as a programmed agent, it does not concern itself with exactly *how* these aspects are to be achieved, but more about *how* it relates to other key structures. The *detailed design* of a system, based on the conceptual design, specifies all aspects of a system's knowledge and behaviour. It describes how the agent's processes will achieve its role. This can be thought of as the step where one is thinking about how a computational agent might achieve what is described in the conceptual design, but it is probably independent of which computer language, or system the agent is destined to be implemented in. A detailed design is an adequate basis for the *operational design*, which deals with some of the nitty-gritty of a specific implementation in a particular system. It stops short of actual programming code, but would be enough for a programmer to implement the system. This final stage will not be discussed in this chapter since it will be different for each programming language or system that is used to implement the final simulation.

The sequence tends to progress roughly from the conceptual towards the concrete, however this is only a general rule; there is no immutable sequence of design: depending on the specific situation, different types of knowledge are available at different points during system design which means that stages need to be iterated or even that, at times, lower level necessities might drive the higher levels of design.

5.2.2 Compositionality of Processes and Knowledge

Compositionality is a general principle that refers to the use of components to structure a design. This process of composition can be extended downwards as far as it is useful, for example, components can themselves be compositional structures in which a number of other, more specific components are grouped. During the design all the components at the different levels are identified. Processes at each of these levels (except the lowest level) are modelled as (process) *components*

composed of entities at the level one lower to the process. Clearly this approach depends upon the possibility of decomposing what is being modelled into separate entities that are somewhat independent, so that the interaction of these components can be considered in turn. In other words it is necessary that what is being modelled is a near-decomposable system (Simon 1962). Although this is not always the case, there are many social actors that, prima facie, *are* themselves composed of other actors, for example political parties or firms. Thus this is often a reasonable approach to take in the field of social simulation. Even when it is not clear that what is being modelled does divide so neatly, then this method provides a systematic approach to attempting to identify and analyse those parts that are amenable to design so that when they are all put together the desired behaviour will *emerge* from their interaction during the simulation. Such emergence is an indication of non-decomposability and can never be guaranteed – one has to find out by running the simulation model, i.e. performing simulation experiments. If the desired behaviour does not emerge then one has to try and work out the possible reasons for the failure and go back to earlier stages of the design and rethink the approach.

In this compositional approach, each process within a multi-agent system may be viewed as the result of interaction between more specific processes. A complete multi-agent system may, for example, be seen to be one single component responsible for the performance of the overall process. Within this one single component a number of agent components within a common internal environment may be distinguished, each responsible for a more specific process. Each agent component may, in turn, have a number of internal components responsible for more specific parts of this process. These components may themselves be composed, again entailing interaction between other more specific processes.

The knowledge and information that is stored, produced, and communicated is as important as the processes. The set of all terms, labels, types, structures etc. that is used to encode and process the knowledge needed in a specific domain or for the purposes of a particular agent may also be seen as a component, a *knowledge structure*. This knowledge structure can be composed of a number of more specific knowledge structures which, in turn, may again be composed of other even more specific knowledge structures.

Compositionality of processes and *compositionality of knowledge* are two independent dimensions of design. Thus a set of processes might be summarised as a single process when appropriate or, in the other direction, broken down into a system of sub-processes. The knowledge structures at one level might be adequate for the purposes of the compositional processes or, maybe, a finer system of description might be needed to be utilised by a finer grain of process representation. For example, some simulations might represent the spread of beliefs through a group as a simple contagion process, with simple entities representing the beliefs and who has them. A finer grained model might include more of a cognitive representation of the knowledge structures involved and how others are persuaded to accept these as the result of a dialogue process.

Compositionality is a means to achieve *information and process hiding* within a model: by defining processes and knowledge at different levels of abstraction,

unnecessary detail can be hidden at those stages, allowing the broader considerations to be considered separately from the component details. Clearly in the realm of social simulation being able to satisfactorily express social processes without always going down to the details is necessary if the resulting model is to be feasible – clearly we cannot simulate social actors, going all the way down to the atoms they are made of. Compositionality also makes it possible to *integrate* different types of components in one agent, providing the structure and means by which they work together.

5.2.3 Problem Description

There are many ways to write a *problem description*. Techniques vary in their applicability, depending on, for example, the situation, the task, or the type of knowledge on which the system developer wishes to focus. Therefore, no particular method will be described here. However, whichever way the problem description is developed it is crucial to capture the key requirements to be imposed on the system – that is, what one *wants to gain* from building and using the simulation. These requirements are part of the initial problem definition, but may also evolve during the development of a system. Different simulations of the *same phenomena* might well be appropriate because each might have different requirements. For example, a simulation model to predict where queues will form on a certain stretch of motorway will probably be different from one to predict whether different proportions of lorries might affect the throughput of traffic, even if both simulations are of traffic on the same stretch of road at the same times.

5.2.4 Conceptual and Detailed Design

A conceptual and detailed design consists of specifications of the following three types:

- Process composition;
- Knowledge composition;
- The relation between process composition and knowledge composition.

These are discussed in turn in more detail below.

5.2.4.1 Process Composition

Process composition identifies the relevant processes at different levels of (process) abstraction, and describes how a process can be defined in terms of lower level processes. Depending on the context in which a system is to be designed two different approaches can be taken: a *task perspective*, or a *multi-agent perspective*.

The task perspective refers to the approach, in which the processes needed to perform an overall task are distinguished first, which are then *delegated* to appropriate agents and the external world. In this approach the agents and the external world are designed later. The *multi-agent perspective* refers to the approach in which agents and an external world are distinguished first and afterwards the processes within each agent and within the external world.

Identification of Processes at Different Levels of Abstraction

Processes can be described at different levels of abstraction; for example, the processes for the multi-agent system as a whole, processes within individual agents and the external world, processes within task-related components of individual agents. Thus in a traffic simulation system processes might include the introduction and removal of vehicles, the collection of statistics and the visualisations of the system state; individual agents representing vehicles might have processes for monitoring their speed and for deciding when to change lane; within these agents might be a reactive component that monitors and adjusts the speed reacting when other traffic gets too close, a learning component that remembers which lanes were faster in the past, and a reasoning component that decides when to change lanes.

Relevant Aspects of a Process

The processes identified are modelled as *components*. For each process the *types of information* used by it for input and resulting as output are identified and modelled as *input and output interfaces* of the component (an interface is a protocol for the information and maybe some specification of a process to deal with it, translating or storing it). So in a traffic simulation the process in an agent may need the distance and the relative speed of any object in its path to be passed to it as an input and the reactions (accelerating, braking) may be passed as its outputs. Clearly in a simple simulation these interfaces will be trivial, but in more complex simulations or where a degree of modularity is required some effort in designing these interfaces might well pay off.

Modelling Process Abstraction Levels

Each level of process is either an *abstraction* of lower levels of component and/or a specialisation of the levels above. These layers of abstraction only go down so far since processes are either *composed* of other components or they may be *primitive*. Primitive components may be either reasoning components (for example based on a knowledge base), or, alternatively, components capable of performing tasks such as calculation, information retrieval, optimisation, etc.

The identification of processes at different levels of abstraction results in the specification of components that can be used as building blocks, and which components are sub-components of which other component. The distinction of different levels of process abstraction results in hiding detail from the processes at the higher levels. Thus a process to decide whether to change lane in the traffic example might be composed of a process to access the memory of how fast each lane was in the past, an estimate of the average speed of the current lane, and how fast the traffic ahead is moving.

Composition

The way in which processes at one level of abstraction in a system are composed of processes at the adjacent lower abstraction level in the same system is called *composition*. This composition of processes is described not only by the component/sub-component relations, but in addition by the (possibilities for) *information exchange* between processes (the *static* aspects), and *task control knowledge* used to control processes and information exchange (the *dynamic* of the composition).

Information Exchange

A specification of information exchange defines which types of information can be transferred between components and the ways by which this can be achieved, called *information links*. Within each of the components *private* information links are defined to transfer information from one component to another. In addition, *mediating* links are defined to transfer information from the input interfaces of encompassing components to the input interfaces of the internal components, and to transfer information from the output interfaces of the internal components to the output interface of the encompassing components. That is the mediating links are those which pass information "up" and "down" the structure of the agent: to their components or up to the entity that they are a component of. Thus in the traffic example there might well be mediating information links from each vehicle up to the simulation to pass information about its current speed and position.

5.2.4.2 Knowledge Composition

Knowledge composition identifies the knowledge structures at different levels of abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The levels of knowledge abstraction may correspond to the levels of process abstraction, but this is not necessarily the case.

Identification of Knowledge Structures at Different Abstraction Levels

The two main structures used as building blocks to model knowledge are: *information types* and *knowledge bases*. These knowledge structures can be identified and described at different levels of abstraction. At the higher levels the details can be hidden. The resulting levels of knowledge abstraction can be distinguished for both information types and knowledge bases.

Information Types

An information type defines the sorts of terms that will be used describe objects or other terms, their kinds, and the relations or functions that can be defined on these objects.[1] Information types can be specified in graphical form, or in formal textual form. Thus the *speed* of objects is a type of knowledge in the traffic example, relatable to other speeds in terms of *relative speed*.

Knowledge Bases

Knowledge bases are structured collections of information held by agents. The specification of the knowledge bases use the information types just described. To specify a knowledge base one needs to say which information types are used in as well as the relationships between the concepts specified in the information types. Thus in a (somewhat complex) driver memory there might be three kinds of information: days of the week, times of the day, lane label and categories of speed. Each lane might relate to a set of past occasions composed of day of the week, time of day and speed category.

Composition of Knowledge Structures

Information types can be composed of more specific information types, following the principle of compositionality discussed above. Similarly, knowledge bases can be composed of more specific knowledge bases. Thus in the example of memory about past lane speeds the sets of past occasions might be a list of limited size ordered first by level of annoyance and secondly by recency.

5.2.4.3 Relation Between Process Composition and Knowledge Composition

Each process in a process composition uses knowledge structures. These will be involved in the building and maintenance of the knowledge structures at their level, but could involve knowledge structures from higher or, occasionally, lower levels.

[1] Such sets of agreed terms are often called an "ontology" in computer science.

So the processes that comprise the cognitive processes of a traffic agent might well be involved in maintaining the memory of past lane speeds but might also relate to the positional clues that are associated with the highest levels of the simulation.

5.2.5 Design Rationale

The *design rationale* is important because it makes explicit the reasoning "glue" that underpins some of the other parts. Essentially it answers the question "*given the modelling target and goals why have the design decisions been made?*" Thus it describes the relevant properties of the design in relation to the requirements identified in the problem description. It also documents the verification of the design – that is how one will check that the implemented system does *in fact* meet its specification, including the assumptions under which the desired properties will hold. All the important design decisions are made explicit, together with some of the alternative choices that could have been made, and the arguments in favour of and against the different options. At the operational level the design rationale includes decisions based on operational considerations, such as the choice to implement an agent's cognitive process in a particular way in order to make the simulation run at a reasonable speed.

5.2.6 Multi-agent Systems in the Simulation of Social Phenomena

The method described above deals with the design process in terms of components and the interactions between those components. In this light, multi-agent systems are not considered specifically. However, in the context of simulating social phenomena, it comes out naturally that in many instances the appropriate "components" are the correlates of observed social actors. In other words it is almost always overwhelmingly sensible to model the target system as a multi-agent system, where agents in the model are representations of the actors (people, firms, etc.) that are known to exist. In a sense, in the social sphere almost everything is an agent, or conversely, agents are nothing special. It is simply that a component that is naturally thought of as having elements of cognition (learning, reasoning etc.) is an agent and will be endowed, as part of the simulation process, with many of the attributes that agents are expected to have (and are discussed later in this chapter). Representations of humans in a simulation will not include all aspects of cognition but, dependent on the modelling goals, might well be much simpler. On the other hand some non-human components, such as a firm, might be represented as an agent, being able to learn, react, and reason in an agent-like way.

Simulations are growing in complexity, not in the least because agents are asked to fulfil different roles over time, and to change their behaviour according to both their own internal learning mechanisms and changing role descriptions. Within the

described approach it has become good practice to first design the organisation, and then the agents and their interaction in such a way that the agents realize the organisation. The next section explicitly considers organisations. The chapter on "Assessing Organisational Design" (Dignum 2013) follows the application of the ideas that are described here.

5.3 Organisations

The organisational approach to simulation design takes the observed and inferred organisational structures as the starting point and considers individual action and agency at a later stage. This is particularly suitable for situations that *seem* to be structured in this way, that is to say the roles and the requirements significantly characterise and constrain individual action. Clearly in many observed cases there is a complex mix of organisational constraint and emergence from individual action so the decision to adopt a primarily organisational approach is a pragmatic one. In many cases a mixture of organisation-based and agent-based approaches will be necessary.

Societies are characterised by complex dynamics involving interaction between many actors and groups of actors. If such complex dynamics take place in an completely unstructured, incoherent manner, then the actors involved will probably not be able to predict much, and not able to use and exploit any knowledge that they have in a useful way. However in many social situations this is not the case, social phenomena are full of structure, and even in initially unstructured situations social actors will often quickly develop norms, rules, habits etc. – effectively creating structure. Some sociologists (e.g. Luhman) have suggested that the *purpose* of human institutional structure is to manage the complexity, in other words to simplify social action and make planning possible. Organisational structure provides co-ordination of the processes in such a manner that the agents involved can function in a more adequate manner. The dynamics in many organisational structures are much more dependable and understood than in apparently entirely unstructured situations.

One key assumption of the organisational approach to simulation design is 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 dynamics through the structure. Within the field of Organisation Theory such organisational structures regulating societal dynamics are studied (see e.g. Kreitner et al. 2001; Mintzberg 1979). In summary, organisational *structure* is used to help specify the *dynamics* (or organisational *behaviour*) of a desired type. A crucial issue for further analysis is how exactly structure is able to *affect* dynamics.

A number of organisation modelling approaches have been developed to simulate and analyse dynamics within organisations in society (e.g. Ferber and Gutknecht 1998; Hannoun et al. 1998, 2000; Hübner et al. 2002a b; Lomi and Larsen 2001; Moss et al. 1998; Prietula et al. 1997). Some of these approaches

explicitly focus on modelling organisational structure, abstracting from the detailed dynamics. Other approaches put less emphasis on organisational structure but focus on the dynamics in the sense of implementing and experimenting with simulation models. Often these simulation models are based on some implementation environment and not specified in an implementation-independent manner using a formally defined conceptual language. However, there are some exceptions to this where the specification approach is supported by an implementation framework.[2] The Agent/Group/Role (AGR) approach (previously called Aalaadin) introduced in (Ferber and Gutknecht 1998) is a good example of the organisational approach. It focusses on organisational structure, abstracting from the details of the dynamics. It helps define a formal relation between the dynamic properties and the organisational structure (Ferber et al. 1999, 2000). The relevance for this chapter is that it shows how dynamics of the organisational structure itself can be modelled: agents can dynamically create, join, or quit groups. This is particularly relevant for simulating situations where the organisational structure is somewhat fluid.

In this section the "dynamics specification" approach exemplified by AGR is presented.[3] The organisational structure is discussed and its parts defined. Then the dynamics of the organisation is discussed in terms of dynamic properties that can be associated to each element of the organisational structure. These dynamic properties can help the simulation and analysis of empirical or simulated traces. The various compositional levels within an organisation are related to the organisational dynamics via a series of relationships. Finally, as a prerequisite to realising an organisation the requirements of the agents are specified from their roles within the organisation model.

5.3.1 Specification of Organisation Structure

In this approach, an organisation is viewed as a framework for activity and interaction through the definition of groups, roles and their relationships. By avoiding an agent-oriented viewpoint, an organisation is regarded as a structural relationship between agents. In this way the organisation is described solely on the basis of its structure, i.e. by the way groups and roles are arranged to form a whole, without being concerned with the way agents actually behave. That is the systems will be analysed from the outside, as a set of interaction modes. The specific architecture of the agents is purposely not addressed in the organisational model.

[2] The Strictly Declarative Modelling Language SDML (Moss et al. 1998) and the use of the agent-oriented modelling approach DESIRE in social simulation as presented in (Brazier et al. 2001) are two examples.

[3] For more information on the use of AGR, see (Jonker and Treur 2003).

The three primitive definitions are:

- The *agents*. The model places no constraints on the internal architecture of agents. An agent is only specified as an active communicating entity which plays roles within groups. This agent definition is intentionally general to allow agent designers to adopt the most accurate definition of agent-hood relative to their application. In other words, the specification of the agent is left as flexible as possible, given the organisational constraints upon its roles.
- *Groups* are sets of agents. Each agent is part of one or more groups. In its most basic form, the group is only a way to tag a set of agents. An agent can be a member of *several* groups at the same time. A major point of these groups is that they can freely overlap.
- A *role* is an abstract representation of an agent function, service or identification within a group. Each agent can have multiple roles and each role handled by an agent is local to a group. Roles could be assigned beliefs; that is, they could reason about whether they should have a particular belief given a certain role. These beliefs can be seen as an additional requirement on the agents playing that role.

Organisation structure is often show as a diagram (for example, as kind of labelled graph; see Fig. 5.3 in Sect. 5.3.5) consisting of roles, groups, and interactions, and of relationships between these elements.

Within AGR an organisation structure consists of a set of groups, the roles in each group and the agents fulfilling those roles. To complete the picture relationships between roles can be specified.

5.3.2 Organisation Structure

An AGR specification of an organisation structure is defined by the following: groups, roles, (intergroup) interactions, transfers (intra-group interactions), which roles are in which groups, the roles that are the source of interactions, the roles that are the destination of interactions, the roles that are the source of transfers, and the roles that are the destination of transfers. Transfers, under this scheme, are within a group as opposed to interactions which may be between groups. Thus it is necessary that the source and destination of all transfers belong to the same group. Although intergroup interactions are defined above as between two roles, this can easily be generalised to intergroup interactions involving more than two roles.

5.3.3 Dynamic Properties of an Organisation

After the foundation of an organisation structure has been defined, the foundations for specification of dynamic properties in an organisation are addressed. The aim is not only to cover simple types of dynamics, such as simple reactive behaviour, but

also more complex dynamics, necessary for the simulation of realistic organisations. The challenge here is to incorporate somehow the organisational structure within the formal description of the organisation's internal dynamics. To this aim, the following approach is introduced:

For each element within the organisational structure characterise its dynamics by a specific set of dynamic properties.

This is based on the structural relations between elements in an organisational structure. Then:

Identify relationships between the sets of dynamic properties corresponding with these elements;

In general, the dynamics of an element within an organisation structure can be characterised by describing how the states of the elements change over time. For a role the 'state' needs to include descriptions of for both the input and the output of the role. Transfers and intergroup interactions are assumed to operate only on input and output states of roles. These roles do not have their own internal state, so no further state is needed to be described for such transfers and intergroup interactions.

An organisational structure defines relations between different elements in an organisation. The dynamics of these different elements are characterised by their dynamic properties. An organisational structure has the aim of keeping the overall dynamics of the organisation manageable. For this reason the structural relations between the different elements within the organisational structure have to somehow impose constraints on or dependencies between their dynamics. Logical relations defined between sets of dynamic properties allow the use of logical methods to analyse, verify and validate organisation dynamics in relation to organisation structure. Within AGR organisation models three aggregation levels are involved:

- The organisation as a whole (the highest aggregation level)
- The level of a group
- The level of a role within a group

A general pattern for the dynamics in the organisation as a whole in relation to the dynamics in groups is as follows:

Dynamic properties for the groups AND *dynamic properties for intergroup role interaction*
⇨ *Dynamic properties for the organisation*

Moreover, dynamic properties of groups can be related to dynamic properties of roles as follows:

Dynamic properties for roles AND *dynamic properties for transfer between roles*
⇨ *Dynamic properties for a group*

The idea is that these are properties dynamically relating a number of roles within one group.

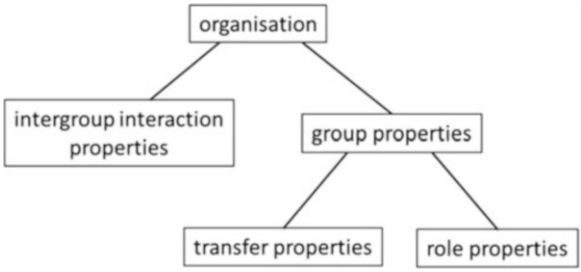

Fig. 5.1 Overview of inter-level relations between dynamic properties within an AGR organisation model

Table 5.1 Types of dynamic properties for an AGR organisation model

Property type	Relating	
Role r	Role r input	Role r output
Transfer from r1 to r2	Role r1 output	Role r2 input
Group G	Input or output of roles in G	
Intragroup interaction	Role r1 output	Role r2 output
Intergroup interaction	Role r1 input	Role r2 output
Organisation	Input or output of roles in O	

An overview of the logical relationships between dynamic properties at different aggregation levels is depicted as an AND-tree in Fig. 5.1.[4]

To define states the notion of state property is useful. The notion of *trace* as a sequence of states over a time frame is used to formalise the dynamics. To formally specify dynamic properties that are essential in an organisation, an expressive language is needed. One can do this using a formal language,[5] however in this chapter this will not be used to retain its accessibility.

We distinguish five kinds of dynamic properties that might be described during a specification (or at least thought about). Not all of these are always necessary. A summary of them is displayed in Table 5.1.

5.3.3.1 Role Dynamic Properties

The *role dynamic properties* relate input to output of that role. This is a subset of the dynamic properties of that role; it is a concern of that role only. For example, the gossip role behaviour: 'whenever somebody tells you something, you will tell it to everybody else' is expressed in terms of input of the role leading to output of the role in a reactive manner.

[4] For formalisation details of the logical relationships put forward above, see (Jonker and Treur 2003).

[5] E.g. the Temporal Trace Language (TTL), which defines the dynamics in terms of a "leads to" relation (Jonker et al. 2001). A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

5.3.3.2 Transfer Dynamic Properties

Transfer properties relate output of the source roles to input of the destination roles. That is they represent the dynamic properties of transfers from one role to another.

Typically, these sets contain properties such as: information is indeed transferred from source to destination, transfer is brought about within x time, arrival comes later than departure, and information departs before other information also arrives before that other information.

5.3.3.3 Group Dynamic Properties

Group dynamic properties relate input and/or output of roles within a group, it relates the roles within the group. An example of a group property is: "if the manager asks anyone within the group to provide the secretary with information, then the secretary will receive this information".

A special case of a group property is an *intragroup interaction* relating the outputs of two roles within a group. A typical (informal) example of such an intragroup interaction property is: "if the manager says 'good afternoon', then the secretary will reply with 'good afternoon' as well". Other examples may involve statistical information, such as "3 out of the 4 employees within the organisation never miss a committed deadline".

5.3.3.4 Intergroup Interaction Dynamic Properties

Intergroup interaction properties relate the input of the source role in one group to the output of the destination role in another group. Note that intergroup interaction is specified by the interaction of roles within the group, and not the groups themselves. Sometimes there are specialist roles for such intergroup interaction. For example, a project leader is asked by one of the project team members (input of role 'project leader' within the project group) to put forward a proposal in the meeting of project leaders (output of role 'member' within the project leaders group).

5.3.3.5 Organisation Dynamic Properties

Organisation dynamic properties relate to input and/or output of roles within the organisation. A typical (informal) example of such a property is: "if within the organisation, role A promises to deliver a product, then role B will deliver this product".

The different types of dynamic properties all relate to different combinations of input and output. Table 5.1 provides an overview of these combinations. Note that with respect to simulation, the above dynamics definition can contain elements that

are redundant: a smaller subset of dynamical properties could form an executable specification of the dynamics of an AGR type organisation – *not all of the above are always needed*.

For example, an organisation could be simulated on the basis of the role dynamic properties, the transfer dynamic properties and the intergroup interactions. The group dynamic properties, including the intragroup role interaction properties, and the organisation properties should emerge in the execution. However specifying them in advance can be used to check what is expected and help verify the simulation.

In order to make an executable organisation model the dynamical properties need to be chosen from those properties that can be executed.

5.3.4 Organisation Realisation

In this section criteria are discussed when allocation of a set of agents to roles is appropriate to realize the organisation dynamics, illustrated for the AGR approach. One of the advantages of an organisation model is that it abstracts from the specific agents fulfilling the roles. This means that all dynamic properties of the organisation remain the same, independent of the particular allocated agents. However, the behaviours of these agents have to fulfil the dynamic properties of the roles and their interactions that have been already specified. The organisation model can be (re)used for any allocation of agents to roles for which:

- For each role, the allocated agent's behaviour satisfies the dynamic role properties,
- For each intergroup role interaction, one agent is allocated to both roles and its behaviour satisfies the intergroup role interaction properties, and
- The communication between agents satisfies the respective transfer properties.

To satisfy the relationships specified above there needs to be a relevant overlap between the agent's ontologies and the role ontologies,[6] i.e. there must be some common referents so that the interactions of the agents are well defined with respect to their roles. Moreover, note that if one agent performs two roles in the group then dynamic properties of communication from itself to itself are required, i.e. that it will receive (at its input state) what it communicates (at its output state): 'it hears itself talking'. The logical relationships can be depicted as in the extension of Fig. 5.1 shown as Fig. 5.2.

Alternatively, if the roles in an intergroup interaction would not be fulfilled by one agent, but by several, this would create a mystery, since input to one agent creates output for another agent, even though the agents are not connected by any

[6] For a more detailed discussion on this issue, see (Sichman and Conte 1998).

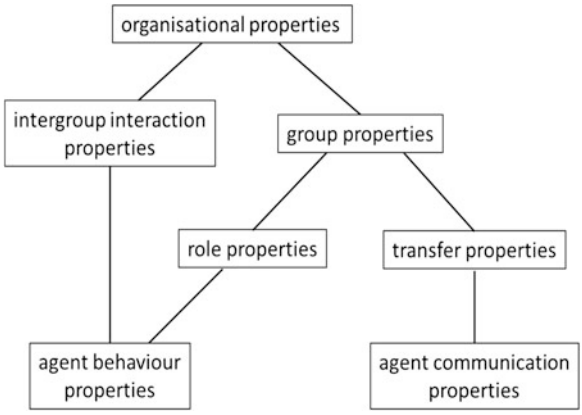

Fig. 5.2 Inter-level relations between dynamic properties for a realised organisation model

transfer since the roles they fulfil are from separate groups. This would suggest that the organisation structure is not complete. The whole idea of specifying the organisational approach through roles is that all communication and interaction is somehow made explicit – in an AGR organisation model it is assumed that the roles in an intergroup interaction are fulfilled by one agent.

5.3.5 Organisational Example

Here the organisational approach to simulation specification is illustrated. This shows how an organisation that is the target of simulation can be analysed into groups, roles and processes. This analysis can then be the basis for the design of the simulation implementation and finally its code. As described this is essentially a top-down analytic approach (in contrast to the more bottom-up agent approach described at the end of this chapter).

In this example, a factory and some of its components are considered. This factory is organised at the highest level according to two divisions: the division that produces certain components (division A) and the division that assembles these components into products (division B). At the lower level, division A is organised in two departments: the work planning department for division A (dep. A1) and the component production department (dep. A2). Similarly, division B is organised in two department roles: one for assembly work planning (dep. B1) and one for product production (dep. B2). This example is illustrated in Fig. 5.3.

Here the two divisions are modelled as groups (depicted by the larger ovals), with the departments as their roles (depicted by smaller ovals within larger ones). A third group, the Connection Group C, models the communication between the two divisions. This group consists of the two roles ‘division A representative’ and ‘division B representative’. Intergroup role interactions (depicted by pairs of dashed lines) are modelled between the role ‘department A1’ in the division A

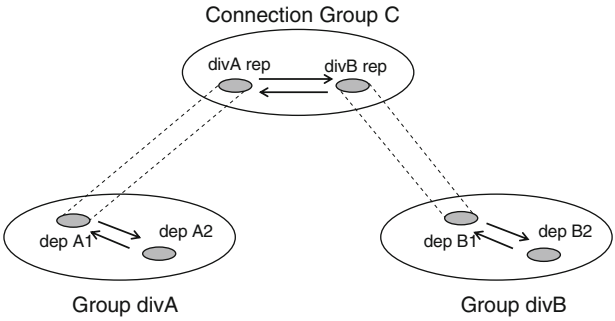

Fig. 5.3 Organisational example. The *smaller ovals* indicate roles, the *bigger ovals* groups. Connections are indicated by the two types of *lines* (*dashed* indicates an intergroup interaction, *solid arrow* indicates a transfer). Membership of a role to a group is indicated by drawing the *smaller role oval* within the *bigger group oval*

group and the role 'division A representative' within the connection group, and between the role 'department B1' in the division B group and the role 'division B representative' within the connection group. Intragroup role transfers model communication between the two roles within each of the groups (depicted by the arrows).

Connections have destination roles (indicated by the arrow points) and source roles (where the arrow originates). Based on the semantic structures of many-sorted predicate logic a more precise formal definition is the following.

5.3.5.1 Groups and Roles in Organisational Example

The example has the following groups, roles, and relationships between them:

- Groups = {divA, divB, C},
- Roles = {depA1, depA2, depB1, depB2, divArep, divBrep},
- Intergroup_interactions = {iAC, iCA, iBC, iCB}
- Transfers = {tA12, tA21, tB12, tB21},
- Some of the relationships are:

Within divA	Organisation level
Role_in(depA1, divA)	Source_of_interaction(divA, iAC)
Role_in(depA2, divA)	Destination_of_interaction(C, iAC)
Source_of_transfer(depA1, tA12)	Source_of_interaction(C, iCA)
Destination_of_transfer(depA2, tA12)	Destination_of_interaction(divA, iCA)
Source_of_transfer(depA2, tA21)	
Destination_of_transfer(depA1, tA21)	

5.3.5.2 Dynamic Properties in Organisational Example

To get the idea, consider the special case of an intragroup role interaction from role r1 to role r2, characterised by dynamic properties that relate output of one role r1 to output of another role r2. Assuming that transfer from output of r1 to input of r2 is adequate and simply copies the information, this property mainly depends on the dynamics of the role r2. Therefore in this case the relationship has the form:

Dynamic properties for role r2 AND
Dynamic properties for transfer from role r1 to role r2
⇨ *Dynamic properties of intragroup interaction from r1 to r2*

Role Dynamic Properties

DP(depA1) Progress Information Generates Planning in depA1

If within division A department A1 receives progress information on component production,then an updated planning will be generated by department A1 taking this most recent information into account.

Group Dynamic Properties

DP(A) A Progress Information Generation

This property is the conjunction of the following two properties.

DP1(A) Initial A Progress Information Generation

Department A1 receives initial progress information on component production processes, involving already available components.

DP2(A) Subsequent A Progress Information Generation

Within the division A group, for any component production planning generated by department A1, incorporating a specific required set of components, progress information on the production of these components will be received by department A1.

Intergroup Interaction Dynamic Properties

Intergroup Role Interaction between A and C: IrRI(A, C)

For the connectivity between the groups A and C, the following intergroup role interaction properties are considered, one from A to C, and one from C to A.

IrRI(depA1, divArep) Progress Information Provision A to B

If within division A progress information on component production is received by department A1, then within the connection group this will be communicated by the division A representative to the division B representative.

IrRI(divArep, depA1) B Progress Information Incorporation by A

If within the connection group the division A representative receives information from the division B representative on which components are needed, then within division A a component production planning will be generated by department A1 taking these into account.

Organisational Dynamic Properties

DP(F) Overall Progress Notification

If a request for a product is made (by a client), then progress information will be provided (for the client).

Realisation

The following allocation of agents agentA1, agentA2, agentB1, agentB2 to roles is possible:

agentA1 – depA1	*agentB1 – depB1*	*agentA1 – divArep*
agentA2 – depA2	*agentB2 – depB2*	*agentB1 – divBrep*

To realise the organisation model, for example agentA1 has to satisfy the following dynamic properties:

DP(agentA1)
If agent A1 receives progress information on component production,
Then an updated planning will be generated by agent A1 taking this most recent information into account.
IrRI(agentA1)
If progress information on component production is received by agent A1,
Then this will be communicated by agent A1 to agent B1
If agent A1 receives information on which components are needed,
Then a component production planning will be generated by agent A1 taking these components into account

5.3.5.3 Conclusion of Organisational Example

One can see how the above analysis is getting us closer to the implementation of a simulation of this organisation. Given details of an organisation this could continue

down the organisational structure. Clearly this kind of analysis is more appropriate when the structure of the organisation is known, and much less appropriate when the structure is only partially known or, indeed, emergent. However, even in those cases it could guide the documentation capturing how and which aspects of an organisation's official structure was translated into a simulation.

5.4 Organisation Design by Requirements Refinement

The previous sections address the question of how the structure and the behaviour of a given organisation can be modelled. This section takes the design perspective. This perspective does not assume a given organisation, but aims at creating a new organisation *in silica*. Whilst on the whole in simulation we are aiming to capture aspects of existing organisations one might want to design one in a number of circumstances.

For example, one may not know the internal structure of an organisation which is part of the system one is trying to model but only how it communicates or relates to the actors around it. In this case to get the simulation to run one would have to invent the organisation. Obviously the danger in this case is that the organisational structure that is chosen might subtly affect how it interacts with other agents and thus have an impact upon the simulation outcomes. However in some cases the internal workings of an organisation *are* effectively insulated from how it interacts with the outside world by regulation and self-interest – in these cases one might well have no choice but to invent its workings on the basis of its external constraints and common knowledge of how such things are arranged.

Another case is where one is not attempting to represent anything that is observed but rather exploring the space of possible organisations. For example one might wish to know which of several possible organisational structures might be best according to some specified criteria. Such "artificial societies" or "thought experiments" are reasonable common, however their relevance is questionable. If a small change in the environment or other aspect (e.g. reliability of internal communication) means that what was a good organisational structure now fails, then the results of such artificial studies are difficult to apply to other cases. In other words the general applicability of such studies is hard to establish. On the other hand if one has a good knowledge of the environment and characteristics where the results of the simulation experiments are to be applied and one does extensive 'what if' experiments testing the robustness of the designs to such small changes then this can be a helpful way forward.

Such an design process starts by specifying requirements for the overall organisation behaviour. The requirements express the dynamic properties that should 'emerge' if appropriate organisational building blocks, such as roles, groups, transfers, group interactions, role behaviours, and so on, are glued together in an appropriate manner in an organisation model. In addition, other requirements on behavioural and structural aspects of the organisation to be created may be

imposed. Given these requirements on overall organisation behaviour (and, perhaps, some additional requirements), organisational structure and organisational behaviour are designed in such a manner that the requirements are fulfilled. The approach described in this section is the method of requirements refinement, which is illustrated for an example.

5.4.1 Designing by Requirements Refinement

In Sect. 5.3.3 a scheme for specifying the dynamic properties and relationships at different levels of aggregation was described; overall organisational behaviour can be related to dynamic group properties and group interaction properties via the following pattern:

Dynamic properties for the groups AND dynamic properties for group interaction
⇨ Dynamic properties for the organisation

This scheme is also useful for the design perspective. Consider a design problem for which the requirements of the overall behaviour are given in the form of dynamic properties. This scheme says that to fulfil these overall dynamic properties, dynamic properties of certain groups and group interactions together imply the organisation behaviour requirements. This process is called *requirements refinement* in that the requirements for the whole organisation are reduced to that of its constituent groups and the interactions between these groups. It thus provides a new, refined set of requirements in terms of the behaviour of groups and group interaction.

Clearly if one has a multi-level organisation with component sub-organisations as well as groups one has a choice as to how *best* to fill in the detail of one's design. One can decide to reduce it first to the behaviour of its constituent groups but it is also possible to first refine requirements for the behaviour of the organisation as a whole to the requirements on the behaviour of parts of the organisation, before further refinement is made to refinements for groups. In each case this is a pragmatic decision and will depend on the organisation being designed.

Subsequently, the required dynamic properties of groups can be refined to dynamic properties of certain roles and transfers, making use of:

Dynamic properties for roles AND dynamic properties for transfer between roles
⇒ Dynamic properties for a group

This provides requirements on role behaviour and transfer that together imply the requirements on the behaviour of the group. Again it is possible to first refine requirements on the behaviour of a group to requirements of the behaviour of parts of the group, before further refinement to role behaviour requirements is made, depending on what is best in each case.

An overview of the inter-level relationships between these dynamic properties at different aggregation levels is depicted in Fig. 5.1, repeated for your convenience

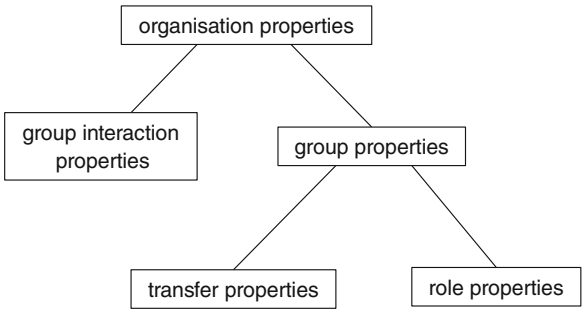

Fig. 5.4 Overview of interlevel relationships between dynamic properties within an organisation model

here as Fig. 5.4. In summary, from the design perspective, a *top-down refinement approach* can be followed. That is, the requirements on overall organisational behaviour can be first refined to requirements on behaviour of groups and group interaction, and then the requirements on behaviour of groups can be refined to requirements on roles and transfers. Notice that as part of this refinement process the organisational structure (e.g., the groups and roles) is defined.

A design problem statement consists of:

- A set of requirements (in the form of dynamic properties) that the overall organisational behaviour has to fulfil
- A partial description of (prescribed) organisational structure that has to be incorporated
- A partial description of (prescribed) dynamic properties of parts of the organisation that have to be incorporated; e.g., for roles, for transfers, for groups, for group interactions.

A *solution specification* for a design problem is a specification of an organisation model (both structure and behaviour) that fulfils the imposed requirements on overall organisation behaviour, and includes the given (prescribed) descriptions of organisation structure and behaviour. Here 'fulfilling' the organisation behaviour requirements means that the dynamic properties for roles, transfers, and group interactions within the organisation model imply the behaviour requirements.

In specific circumstances, part of the organisational structure and/or behaviour may already be prescribed by requirements. For example, the organisational structure may already be prescribed; in such a case only the organisation dynamics is designed, for the given organisational structure. Other, more specific cases are, for example, *role behaviour design* and *interaction protocol design*.

5.4.1.1 Role Behaviour Design

For role behaviour design the organisational structure and the transfers and interactions are completely prescribed. However, appropriate dynamic properties for the different roles have yet to be found, to satisfy the requirements for the

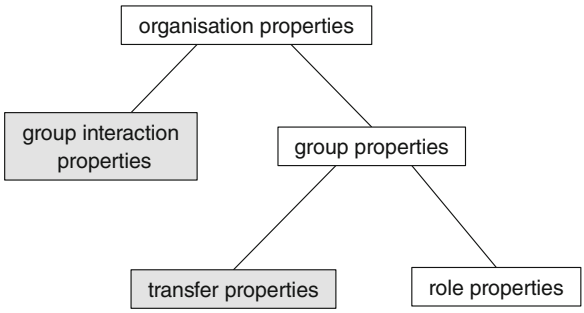

Fig. 5.5 Role behaviour design

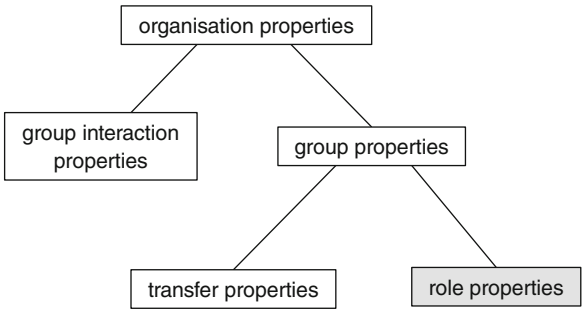

Fig. 5.6 Organisation design

organisational behaviour that are imposed; see Fig. 5.5. Here (and in Fig. 5.6) the grey rectangles indicate what is already given as prescribed and the transparent rectangle what has to be designed.

5.4.1.2 Interaction Protocol Design

For interaction protocol design the organisational structure and role dynamics are completely prescribed, but appropriate transfer and interaction dynamics have to be found to satisfy given requirements for the organisational behaviour that are imposed; see Fig. 5.6.

5.5 The Agent Approach

The agent approach is contrary to the organisational approach. It starts with the agents and its properties and attempts to work upwards towards the whole system. This is more useful in situations where the "top down" social constraints upon action are weak or non-existent, and it is the "upwards" emergence of outcome from the many micro-level interactions that is more important. Clearly, if one *was* in a situation where top down social and/or organisational constraints were severe then

one would have no guarantee that working bottom-up in this manner one would be able to meet those constraints at the higher levels of the structure. It would be like trying to organise the production of a kind of car by the random meeting of people with particular parts and skills without any planning. However, especially in the *development of new social structure* such "bottom-up" processes can be crucial, so that the agent approach can be appropriate for investigating such issues. Often, for social phenomena some mix of both approaches is necessary, first a bit of one, then a bit of the other etc.

5.5.1 Some Agent Notions

The term agent has been used for a wide variety of applications, including: simple batch jobs, simple email filters, mobile applications, intelligent assistants, and large, open, complex, mission critical systems (such as systems for air traffic control).[7] Some of the key concepts concerning agents lack universally accepted definitions. In particular, there is only partial agreement on what an agent is. For example, simple batch jobs are termed agent because they can be scheduled in advance to perform tasks on a remote machine, mobile applications are termed agent because they can move themselves from computer to computer, and intelligent assistants are termed agents because they present themselves to human users as believable characters that manifest intentionality and other aspects of a mental state normally attributed only to humans. Besides this variety in different appearances of agents, the only precise description of the agents involved is their implementation code. As a result, existing agent architectures are only comparable in an informal manner – just because something is *called* an agent-architecture or an agent does not mean that it is suitable for simulating a human or social actor. Especially if the goal of the agent-based system is a complex simulation, a principled, design-oriented description of the organisation, and of the agents in it at a conceptual and logical level is of the essence, since the control, testing, verification and validation of such complex simulations is an issue. Spending time on a more formal and staged approach to simulation design can make life a lot easier later. Due to the organisational nature, and the complexity of intelligent agents and their interaction, a more formal compositional design method for agents is necessary.

As agents show a variety of appearances, perform a multitude of tasks, and their abilities vary significantly, attempts have been made to define what they have in common. The *weak notion of agent* is seen as a reference. The weak notion of agent is a notion that requires the behaviour of agents to exhibit at least the following four types of behaviour:

[7] Many of the notions discussed in this and the following section are adopted from (Wooldridge and Jennings 1995), (Nwana 1996), (Nwana and Ndumu 1998) and (Jennings and Wooldridge 1998).

- Autonomous behaviour
- Responsive behaviour (also called reactive behaviour)
- Pro-active behaviour
- Social behaviour

Autonomy relates to control: although an agent may interact with its environment, the processes performed by an agent are in full control of the agent itself. A*utonomous* behaviour is defined as:

> . . . where the system is able to act without the direct intervention of humans (or other agents) and should have control over its own actions and internal state.

This means that an agent can only be requested to perform some action, and:

> The decision about whether to act upon the request lies with the recipient.

Examples of autonomous computer processes are: process control systems (e.g., thermostats, missile guiding systems, and nuclear reactor control systems), software daemons (e.g., one that monitors a user's incoming email and obtains their attention by displaying an icon when new, incoming email is detected), and operating systems.

Many processes that exhibit autonomous behaviour are called agents. However, if such agents do not exhibit flexible behaviour, they are not, in general, considered to be *intelligent* agents. An intelligent agent is a computer system that is capable of flexible autonomous actions in order to meet its design objectives – indeed Randall Beer (1990) defined intelligence as "the ability to display adaptive behaviour". Intelligence requires flexibility with respect to autonomous actions, meaning that intelligent agents also need to exhibit responsive, social, and pro-active behaviour.

An agent exhibits *responsive* (or *reactive*) behaviour if it reacts or responds to new information from its environment. Responsive behaviour is where:

> Agents perceive their environment (which may be the physical world, a user, a collection of agents, the Internet, etc.) and respond in a timely fashion to changes that occur in it.

A barometer is a simple example of a system that exhibits responsive behaviour: It continually receives new information about the current air pressure and responds to this new information by adjusting its dial.

Pro-active behaviour is where:

> Agents do not simply act in response to their environment, but are be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate.

Pro-active behaviour in some sense is the most difficult of the required types of behaviour for an agent defined according to the weak agent notion. For example, pro-active behaviour can occur simultaneously with responsive behaviour. It is possible to respond to incoming new information in an opportunistic manner according to some goals. Also initiatives can be taken in response to incoming new information from the environment, and thus this behaviour resembles responsive behaviour. However, it is also possible to behave pro-actively when no new

information is received from the environment. This last behaviour can by no means be called responsive behaviour.

An agent exhibits *social* behaviour if it communicates and co-operates with other agents. Jennings and Wooldridge define social behaviour as when:

> Agents are able to interact, when they deem appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities.

An example of an agent that exhibits social behaviour is a car: it communicates with its human user by way of its dials (outgoing communication dials: speed, amount of fuel, temperature) and its control mechanisms (incoming communication control mechanisms: pedals, the steering wheel, and the gears). It co-operates with its human user, e.g., by going in the direction indicated by the user, with the speed set by that user.

Agents can also be required to have additional characteristics. Here three of these characteristics are discussed: adaptivity, pro-creativity, and intentionality.

Adaptivity is a characteristic that is vital in some systems. An adaptive agent learns and improves with experience. This behaviour is vital in environments that change over time in ways that would make a non-adaptive agent obsolete or give it no chance of survival. This characteristic is modelled in simulations of societies of small agents, but also, for example, in adaptive user interface agents.

Pro-creativity is of similar importance to find agents that satisfy certain conditions. The chance of survival is often measured in terms of a fitness function. This characteristic is found in various simulations of societies of small agents (see the literature in the area of Artificial Life). A computer virus is an infamous form of a pro-creative agent.

An *intentional system* is defined by Dennett to be an entity

> . . . whose behaviour can be predicted by the method of attributing beliefs, designs and rational acumen.

Mentalistic and intentional notions such as *beliefs*, *desires*, *intentions*, *commitments*, *goals*, *plans*, *preference*, *choice*, *awareness*, may be assigned to agents. The *stronger notion of agenthood,* in which agents are described in terms of this type of notions, provides additional metaphorical support for the design of agents.

5.5.2 Representative Agents

Of course a software agent need not have all the characteristics of something it *represents*. Thus, depending on the model purpose, it is quite possible to represent an intelligent actor by a relatively simply object, that might not even be meaningfully called an agent. For example, if simulating the movement of a crowd or traffic, it might not be necessary to include much, if any, of the features discussed above.

So sometimes something is called an agent because it *represents* an agent, but this usage conflates what is being modelled and the model, which may cause confusion. How and when it is necessary to include the various features that are known about the actor being modelled in the model is a crucial modelling question. However it is not a question to which there is a general answer since this goes to the heart of the difficulty of making sense of the complexity that we observe.

The power of the agent idea and approach to programming, clearly comes from the apparent efficacy of observed social actors that seem to be able to organise themselves in useful and adaptive ways, that they have the characteristics listed above. This differs qualitatively from more traditional computer science approaches to programming. It also comes from the power of the analogy with humans to guide the direction of programming – we have a deep mundane knowledge of how humans work (c.f. Dennett 1996) and this can help in design decisions, for example whether a certain feature or ability is necessary to model a particular social system. In this sense the idea of an agent can be thought of as a stance, in comparison to the "intentional" stance mentioned above – it may be useful to think of a computational object as an agent, having some of the sort of properties we know real human actors have. However there are clearly more and less useful applications of this stance: I may think of a light switch as an agent, but it has limited usefulness in terms of understanding or modelling it. In the contrary direction on can think of a fully autonomous and intelligent agent such as a human as a merely a physical particle for some circumstances, however this is open to question, depending upon the purpose and target of the exercise. Clearly, in many circumstances and for many purposes, treating complex social actors as if they were simple (for example acted upon in the manner of a simple linear influence plus some randomness) is insufficient since individual complexity can impinge upon the social complexity that results.

5.5.3 *Agent Properties*

The notions of agency discussed above are highly abstract notions. In order to design agents, it is necessary to be familiar with a number of primitive agent concepts.[8] These primitive concepts serve as an ontology or vocabulary used to express analyses and designs of applications of agents and multi-agent systems. Two classes of primitive notions are distinguished: those used to describe the behaviour of agents in terms of their external (or public) states and interactions (Sect. 5.5.3.1), and those used to describe the behaviour of agents in terms of their internal (or private) states, and processes (Sect. 5.5.3.2). To illustrate these concepts, some example agents are discussed in Sect. 5.5.4.

[8] The material in this section is based on (Brazier et al. 2000).

5.5.3.1 External Primitive Concepts

Two types of interaction of an agent with its environment are distinguished, depending on whether the interaction takes place with an agent or with something else (called an *external world*), for example a database, or the material world. For each of these two types of interaction specific terminology is used.

Interaction with the External World

Two primitive types of interaction with the external world are distinguished. The first type of interaction, *observation*, changes the information the agent has about the world, but does not change the world state itself, whereas the second type, *performing an action*, does change the world state, but does not change the information the agent has about the world. Combinations of these primitive types of interaction are possible; for example, performing an action, and observing its results.

Observation

In which ways is the agent capable of observing or sensing its environment? Two types of observation can be distinguished: the agent passively receives the results of observations without taking any initiative or control to observe (*passive observation*), or the agent actively initiates and controls which observations it wants to perform; this enables the agent to focus its observations and limit the amount of information acquired (*active observation*).

Execution of Actions in the External World

An agent may be capable of making changes to the state of its environment by initiating and executing specific types of actions.

Communication with Other Agents

Two directions of communication are distinguished, which can occur together: *outgoing communication* (is the agent capable of communicating to another agent; to which ones?), and *incoming communication* (is the agent capable of receiving communication from another agent; from which ones?).

5.5.3.2 Internal Primitive Concepts

A description in terms of the external primitive concepts abstracts from what is inside the agent. In addition to descriptions of agents in terms of the external concepts, descriptions in terms of internal concepts are useful. The following internal primitive agent concepts are distinguished.

World and Agent Models

An agent may create and maintain information on (a model of) external *world* based on its observations of that world, on information about that world communicated by other agents, and its own knowledge about the world. The agent may also create and maintain information on (models of) *other agents* in its environment based on its observations of these agents as they behave in the external world, on information about these agents communicated by other agents, and knowledge about the world.

Self Model and History

Some agents create and maintain information on (a model of) their own characteristics, internal state, and behaviour. Or the agent creates and maintains a history of the world model, or agent models, or self model, or own and group processes.

Goals and Plans

To obtain pro-active, goal-directed behaviour, an agent represents, generates, and uses explicit goals and its own plans of action in its processing.

Group Concepts

Besides individual concepts, agents can use group concepts that allow it to co-operate with other agents. For example, joint goals: is the agent capable of formulating or accepting and using goals for a group of agents, i.e., goals that can only be achieved by working together? Or joint plans: is the agent capable of representing, generating, and using plans of action for joint goals, i.e., involving which actions are to be performed by which agents in order to achieve a certain joint goal? Also commitments to joint goals and plan, negotiation protocols and strategies can be useful group concepts for agents, depending on their role and function.

Table 5.2 External primitive concepts for an elevator

External primitive concepts	Elevator
Interaction with the world	
Passive observations	Presence of objects between doors (optically)
	Total weight
	Its position
Active observations	Presence of objects between the doors (mechanically)
Performing actions	Moving
	Opening and closing doors
Communication with other agents	
Incoming communication	From users in the elevator
	Where they want to go (pushing button in elevator)
	From users outside
	Where they want to be picked up (pushing button outside elevator)
Outgoing communication	To users in the elevator
	Where we are (display)
	There is overweight (beep)
	To users outside
	Where is the elevator (display)
	In which direction it moves (display)

5.5.4 Example of the Agent Approach: An Elevator

Let us illustrate the agent concepts introduced above by an example: an elevator is analysed from the agent perspective using these basic concepts (Table 5.2). This might be an element in the simulation of how people move around a building. The advantage of using an elevator as an example is that it does interact with users as an agent, but it is well known and simple enough to make a clear example of specifying an agent using the agent-oriented approach.

5.5.4.1 External Primitive Concepts (Table 5.2)

Observation

Observations are performed continually. However, it receives passive observation results on the presence of objects between the doors (an optical sensor), the total weight of its contents, and its position in the building (at which floor). Besides it is able to perform active observation: the presence of objects between the doors (a mechanical sensor which is moved in the door opening just ahead of the doors themselves).

Table 5.3 Internal primitive concepts for an elevator

Internal primitive concepts	Elevator
World model	The current floor, max load, current load
Agent models	A user wants to be picked up from floor X
	A user wants to go to floor Y
Self model	When maintenance is next due
History	When maintenance was last performed
Goals	To go to floor X to pick up somebody
	To go to floor Y to deliver somebody
Plans	The order in which the required floors are visited
	Sometimes: the speed that is taken
Group concepts	
Joint goals	With other elevators to transport people and goods as efficiently as possible
Joint plans	Some elevators are capable of distributing the work
Commitments	The elevators then commit to their part of the work
Negotiation protocol	To reach a good distribution, they may have to negotiate
Negotiation strategies	To reach a good distribution, they may have to negotiate

Performing Actions

Its actions are moving itself (and people) vertically from one position to another and opening and closing doors.

Incoming Communication

The elevator receives communication from users by buttons that have been pressed outside (to call the lift and indicate the direction they wish to go in) and inside the lifts (providing information about the floor to which they wish to be transported).

Outgoing Communication

The elevator communicates to a user by indicating which floor the lift is on (both inside and outside the lifts) and sounding beeps (information about overload) (Table 5.2).

5.5.4.2 Internal Primitive Concepts (Table 5.3)

World and Agent Models

Elevators know where they are, to do this they keep track of which floor they are on based on their actions (going two floors up, going one floor down) they perform.

Table 5.4 Types of behaviour for an elevator

Types of behaviour	Elevator
Autonomy	Yes
Responsiveness	In reaction to user requests
	In immediate reaction to observed objects between the doors
Pro-activeness	Taking the initiative to go to a normally busy floor, if empty and not being called by a user
Social behaviour	Co-operation with users, and, sometimes, with other elevators
Own adaptation and learning	Often not possible

Furthermore, the elevator knows if the weight in the lift is over its maximum limit. The agent information of the user goals (where they want to go) may be maintained as well.

Self Model and History

The agent does not know what actions it previously performed to perform its current task. It might have an explicit representation of when it has last received maintenance.

Goals and Plans

Modern elevators make use of the explicit goals (adopted from the goals communicated by the users). The goals are used to determine which actions to perform. They may even make plans for reaching these goals: determine the order of actions, for example when one of the users has the goal to be at a higher floor and another on a lower floor.

Group Concepts

The elevator co-operates with its users. The elevator might also be designed to co-operate with other elevators so that they could strategically distribute themselves over the floors. The goals adopted from the goals communicated by the users are *joint goals* (joint with the users), and sometimes even joint with the other elevators. Modern elevators are capable of distributing the work load, and thus of making *joint plans*. To achieve the joint goals an elevator must *commit* to its part of the work as specified in the joint plans. To make a joint plan, the elevators might negotiate using a particular *strategy* as to which elevator goes where. Negotiation is only possible if a *negotiation protocol* is followed.

5.5.4.3 Types of Behaviour (Table 5.4)

Autonomy

As soon as it is activated, no system or human is controlling its machinery, and (normally) it is not switched off and on by the user. The elevator has full control of its motor, doors, and lights.

Pro-activeness

The simplest elevators stay where they are (some take the initiative to close their doors) when no longer in use, but more intelligent elevators go to a strategic floor (e.g., the ground floor).

Reactiveness

The elevator reacts to the immediate stimuli of buttons pressed, therefore, it is shows reactive behaviour. People often have to wait for the elevator as the elevator picks up people on other floors, however, the elevator does not forget a signal and will, eventually, come to the requested floor.

Social Behaviour

The elevator co-operates with users and, sometimes, with other elevators.

Own Adaptation and Learning

Simple elevators are not capable of adjusting their own behaviour to new situations, nor are they capable of learning. However, it is possible to conceive of more intelligent elevators that can learn the rush hours for the different floors.

5.5.4.4 Conclusion of Elevator Example

One can see that the above analysis has clarified what is needed to implement a model of this element within a simulation. The objects, properties and processes of the simulation code is an easy step from here. One also sees that some of the complexities of the agent have been considered before the implementation starts, in this way cleaner and more maintainable code might be produced, fewer mistakes

made in the implementation and some of the complexities in terms of user-lift interaction considered and anticipated. Of course in the case of simulating a human agent there are likely to be many unknowns in terms of their goals, group concepts etc. – unless the simulators simply add in their informed guesses they will have a considerable job finding evidence to guide them in the answers to fill in within such an analysis. Thus this kind of analysis is not a total solution when simulating complex social actors whose attributes and internal states may be largely unknown.

5.6 Conclusion

More formal approaches to simulation design can help make complex implementations manageable and can probably save one time in the longer-run. It also makes the simulation easier to check, validate, re-implement and further develop. These approaches do this in a three principled ways. *Firstly*, by encouraging the more complete documentation of the intentions and decisions of a designer/implementer. One can see a lot of this chapter as a check-list of all the aspects one might think about and record. *Secondly*, it helps encourage doing this in an explicit and exact manner. We have not displayed some of the formal notation that can be used in this chapter, since we did not want to overwhelm the reader, however for those who wish to utilise this style of design to the fullest will naturally find them themselves adopting a variety of formal languages and diagrams in pursuit of precision. *Thirdly*, it suggests a method by which a complex specification can be iteratively refined from abstract and high-level entities towards a detailed implementation. In this way the design decisions do not all have to be made simultaneously but can be made in stages.

Further Reading

For readers interested in software engineering approaches Bergenti et al. (2004) give a thorough introduction to and overview of current methodologies. Gilbert and Terno (2000) offer suggestions on techniques for building and structuring agent-based simulation models, particularly geared towards use in the social sciences.

In addition to methodologies, a lot of work has been done in the development of programming languages and platforms to support the implementation of multi-agent systems and models. Bordini et al. (2010) focus on a comprehensive presentation of MAS programming, including four approaches that are based on formal methods, whereas Railsback et al. (2006) provide a review of platforms for agent-based simulations.

References

Beer RD (1990) Intelligence as adaptive behavior: an experiment in computational neuroethology. Academic, Boston

Bergenti F, Gleizes M-P, Zambonelli F (eds) (2004) Methodologies and software engineering for agent systems: the agent-oriented software engineering handbook. Kluwer, Boston

Bordini RH, Dastani M, Seghrouchni AEF (eds) (2010) Multi-agent programming: languages, platforms and applications. Springer, Berlin

Brazier FMT, Jonker CM, Treur J (1998) Principles of compositional multi-agent system development. In: Cuena J (ed) IT & KNOWS – information technology and knowledge systems, IFIP world computer congress 1998 (Schriftenreihe der Österreichischen Computer Gesellschaft, 122). OCG, Wien, pp 347–360

Brazier FMT, Jonker CM, Treur J (2000) Compositional design and reuse of a generic agent model. Appl Artif Intell J 14:491–538

Brazier FMT, van Eck PAT, Treur J (2001) Modelling a society of simple agents: from conceptual specification to experimentation. J Appl Intell 14:161–178

Daniel C. Dennett (1996), *The Intentional Stance* (6th printing), Cambridge, Massachusetts: The MIT Press, ISBN 0-262-54053-3 (First published 1987)

Dignum V (2013) Assessing organisational design. Chapter 20 in this volume

Ferber J, Gutknecht O (1998) A meta-model for the analysis and design of organisations in multi-agent systems. In: Demazeau Y (ed) Proceedings of the third international conference on multi-agent systems (ICMAS'98), Paris, 3–7 July 1998. IEEE Computer Society Press, pp 128–135

Ferber J, Gutknecht O (1999) Operational semantics of a role-based agent architecture. In: Jennings NR, Lespérance Y (eds) Intelligent agents VI, proceedings of the 6th international workshop on agent theories, architectures and languages, ATAL'99 (Lecture notes in Computer Science), vol 1757. Springer, Berlin, pp 205–217

Ferber J, Gutknecht O, Jonker CM, Müller JP, Treur J (2000) Organization models and behavioural requirements specification for multi-agent systems. In: Proceedings of the fourth international conference on MultiAgent systems (ICMAS), Boston, 10–12 July 2000. IEEE Computer Society, pp 387–388

Galán J et al (2013) Checking simulations: detecting and avoiding errors and artefacts. Chapter 6 in this volume

Gilbert N, Terno P (2000) How to build and use agent-based models in social science. Mind Soc 1(1):57–72

Hannoun M, Sichman JS, Boissier O, Sayettat C (1998) Dependence relations between roles in a multi-agent system: towards the detection of inconsistencies in organization. In: Sichman JS, Conte R, Gilbert N (eds) Multi-agent systems and agent-based simulation, proceedings of the first international workshop on multi-agent based simulation, MABS'98. (Lecture notes in artificial intelligence), vol 1534. Springer, Berlin, pp 169–182

Hannoun M, Boissier O, Sichman JS, Sayettat C (2000) MOISE: an organizational model for multi-agent systems. In: Monard MC, Sichman JS (eds) Advances in artificial intelligence, international joint conference 7th Ibero-American conference on AI, 15th. Brazilian symposium on AI (IBERAMIA-SBIA 2000), Atibaia, 19–22 Nov 2000. Proceedings (Lecture notes in computer science), vol 1952. Springer, Berlin, pp 156–165

Hübner JF, Sichman JS, Boissier O (2002a) A model for the structural, functional and deontic specification of organizations in multiagent systems. In: Bittencourt G, Ramalho GL (eds) Advances in artificial intelligence. Proceedings of 16th Brazilian symposium on artificial intelligence (SBIA'02), Porto de Galinhas/Recife, Brazil, 11–14 Nov 2002. (Lecture notes in computer science), vol 2507. Springer, Berlin, pp 439–448

Hübner JF, Sichman JS, Boissier O (2002b) MOISE+: towards a structural, functional and deontic model for MAS organizations. In: Castelfranchi C, Johnson WL (eds) Proceedings of the first international joint conference on autonomous agents and multi-agent systems, AAMAS 2002, Bologna, 15–19 July 2002. ACM Press, pp 501–502

Jennings NR, Wooldridge M (eds) (1998) Agent technology: foundations, applications, and markets. Springer, Berlin

Jonker CM, Treur J (2003) Relating structure and dynamics in organisation models. In: Sichman JS, Bousquet F, Davidsson P (eds) Multi-agent-based simulation II, third international workshop, MABS 2002, Bologna, July 2002, revised papers (Lecture notes in AI), vol 2581. Springer, Berlin, pp 50–69

Jonker CM, Treur J, Wijngaards WCA (2001) Temporal languages for simulation and analysis of the dynamics within an organisation. In: Dunin-Keplicz B, Nawarecki E (eds) From theory to practice in multi-agent systems, proceedings of the second international workshop of central and eastern Europe on multi-agent systems, CEEMAS'01 (Lecture notes in computer science), vol 2296. Springer, Berlin, pp 151–160

Kreitner R, Kunicki A (2001) Organisational behavior. McGraw-Hill, New York

Lomi A, Larsen ER (2001) Dynamics of organizations: computational modeling and organization theories. AAAI Press, Menlo Park

Mintzberg H (1979) The structuring of organisations. Prentice Hall, Englewood Cliffs

Moss S, Gaylard H, Wallis S, Edmonds B (1998) SDML: a multi-agent language for organizational modelling. Comp Math Org Theory 4(1):43–70

Norling E, Edmonds B, Meyer R (2013) Informal approaches to developing simulation models. Chapter 4 in this volume

Nwana HS (1996) Software agents: an overview. Knowl Eng Rev 11(3):205–244

Nwana HS, Ndumu DT (1998) A brief introduction to software agent technology. In: Jennings M, Wooldridge NR (eds) *Agent Technology: Foundations, Applications and Markets*. Springer Verlag, Berlin, pp 29–47

Prietula M, Gasser L, Carley K (1997) Simulating organizations. MIT Press, Cambridge, MA

Railsback SF, Lytinen SL, Jackson SK (2006) Agent-based simulation platforms: review and development recommendations. Simulation 82(9):609–623

Sichman JS, Conte R (1998) On personal and role mental attitudes: a preliminary dependence-based analysis. In: Oliveira F (ed) Advances in AI. Proceedings of the 14th Brazilian symposium on artificial intelligence, SBIA'98 (Lecture notes in artificial intelligence), vol 1515. Springer, Berlin, pp 1–10

Simon HA (1962) The architecture of complexity. Proc Am Philos Soc 106(6):467–482

Wooldridge M, Jennings NR (1995) Agent theories, architectures, and languages: a survey. In: Wooldridge M, Jennings NR (eds) Intelligent agents, proceedings of the first international workshop on agent theories, architectures and languages, ATAL'94 (Lecture notes in AI), vol 890. Springer, Berlin, pp 1–39