A Formal Specification Language for Agent-Oriented Software Engineering

Hong Zhu
Department of Computing, School of Technology, Oxford Brookes University
Wheatley Campus, Oxford OX33 1HX, UK. Tel.: ++44 1865 484580
Email: hzhu@brookes.ac.uk

ABSTRACT

This paper reports a formal specification language SLABS for developing multi-agent systems. One of the most appealing features of agent technology is its natural way to modularise complex systems in terms of multiple interacting autonomous components. This feature is supported by the language facility castes in the formal specification language SLABS for modular and composable specification of multi-agent systems.

Categories and Subject Descriptors

D.2.1 [**Requirements / Specifications**]: Languages - *formal specification language*; Methodologies - *agent-oriented methodology*. I.2.11 [**Distributed Artificial Intelligence**] Intelligent agents, Languages and structures, Multi-agent systems.

General Terms

Languages

Keywords

Formal specification language, Multiagent Systems, Scenario, Software Agent, Caste

1. INTRODUCTION

In recent years, agent technology has been increasingly applied to critical application areas. It has also clearly demonstrated that its suitability for web-based applications [1]. However, developing agent-based systems is extremely difficult because their dynamic behaviours are difficult to specify, analyse, verify and validate. The new features of agent-based systems demand new methods for the specification of agent behaviours and for the verification and validation of their properties. It has been recognised that the lack of rigour is one of the major factors hampering the wide-scale adoption of agent technology [2]. On the other hand, the modularity inherent in multi-agent systems can offer a new approach to decomposing complicated formal specifications into composable modular components.

Despite the large number of publications on agents in the literature, we are lack of researches on language facilities that support the development of large-scale complicated multi-agent systems. In this paper, we report the language SLABS, which is a formal specification language for agent-based systems.

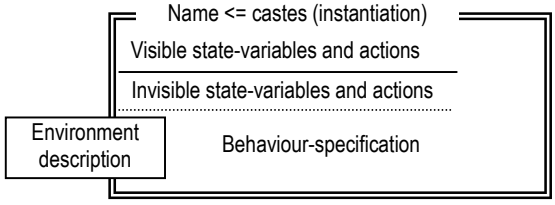
2. THE LANGUAGE SLABS

2.1 Agents and Castes

The specification of a multi-agent system (MAS) in SLABS consists of a set of specifications of agents and castes.

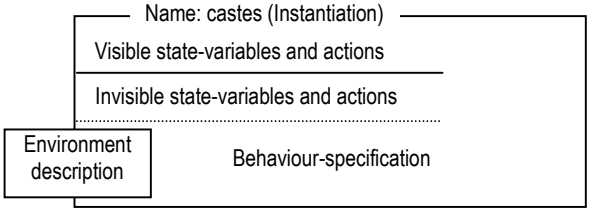
Caste is one of the novel concepts in SLABS, which is a natural evolution of the concepts of classes in object-orientation. Castes can play a significant role in the requirements analysis and specification as well as design and implementation of MAS [3]. A caste description contains a description of the structure of its states and actions, a description of its behaviour, and a description of its environment. It can be equivalently represented in a text form or in a graphic form similar to schemas in Z [4].

caste-description **::=** *Caste* name **[**<= **{** caste-name **/** ,**}** **]** **[** instantiation ; **]**
[environment-description ; **]**
[structure-description ; **]** **[** behavior-description ; **]**
end name



The clause 'Caste C <= C_1, ..., C_n' specifies that caste C inherits the structure, behaviour and environment descriptions of existing castes C_1,, ..., C_n. Thus, a binary inheritance relation $\prec$ is defined on the castes. The inheritance relation is required to be a partial ordering on castes.

The relationship between agents and castes is similar to what is between objects and classes. The difference is that an agent can join into a caste or quit from a caste at run-time. The following gives the graphic form of agent descriptions.



Let $A \in_t C$ denote that agent A belongs to caste C at time t. We require that for all agents A and castes C and D, for all times t,

$A \in_t C \wedge C \prec D \Rightarrow A \in_t D$.

2.2 Environment

The SLABS language enables software engineers to explicitly specify the environment of an agent as a subset of the agents in the system that may affect its behaviour. This is another fundamental difference between agents/castes and objects /classes. The syntax for the description of environments is given below.

Environment-description **::=**
ENVIRONMENT **{ (** agent-name **|** *All*: caste-name
| variable : caste-name **) /** , **}**$^+$

where an agent name indicates a specific agent in the system. 'All' means that all the agents of the caste have influence on its behaviour. As a template of agents, a caste may have parameters. The variables specified in the form of "identifier: class-name" in the environment description are parameters. Such an identifier can be used as an agent name in the behaviour description of the caste. It indicates an agent in the caste when instantiated.

2.3 State and Action Spaces

In SLABS, the state space of an agent is described by a set of variables with keyword VAR. The set of actions is described by a set of identifiers with keyword ACTION. An action can have a number of parameters. An asterisk before the identifier indicates invisible variables and actions.

structure-description **::=**
[*Var* **{ [** * **]** identifier: type; **}**$^+$ **]**
[*Action* **{ [***] action-declaration */ ; **}**$^+$**]**
action-declaration **::=** identifier **|** identifier (**{ [** parameter**:]** type **/** , **}**$^+$)

2.4 Behaviour

The SLABS language uses transition rules to specify agent's behaviour. Each rule consists of a description of a scenario of the environment, the action to be taken by the agent in the scenario and a condition of the agent's internal state and previous behaviour. The syntax of behaviour rules is given below.

Behaviour-rule **::=**
[< rule-name >**:]** pattern **| [** prob **]**–>event,
[*if* Scenario**] [***where* pre-cond**]** ;

In a behaviour rule, the pattern describes the agent's previous behaviour. The scenario describes the situation in the environment. The where-clause is the pre-condition of the action to be taken by the agent. The event is the action to be taken in the scenario and if the pre-condition is satisfied.

There are four basic forms of scenarios that can be logically combined by & (and), $\vee$ (or) and $\neg$ (not) to form more complicated description of the system's state:

1. *Unanimous behaviours* $\forall a \in C : p$: all agents in caste C have the same pattern p of behaviour;
2. *Existential behaviours* $\exists a \in C : p$: there is an agent in caste C that demonstrated a pattern p of behaviour;
3. *Selective (or Individual) behaviour* A: p : a specific agent A demonstrates a pattern p of behaviour;
4. *Statistical behaviours* $\mu a \in C : p$: it is the number of agents in caste C that demonstrated the pattern p of behaviour.

A pattern describes the behaviour of an agent by a sequence of observable state changes and observable actions. A pattern is written in the form of $[p_1, p_2, ..., p_n]$ where $n \geq 0$.

An example of behaviour rule is given below. It states that if all agents of the caste CRegionUsers (Critical Region Users) give permissions to access a region, the agent will enter the region by change its state from "Waiting" to 'In Region'.

[!State=Waiting] |–> !State'=InRegion;
if ∀A: CRegionUsers.[PermissionOK(Region,
MyName, RequestTime), $^k]

3. EXAMPLE

A number of examples of formal specification of MAS in SLABS have been given in our previous papers, which include the Mae's personal assistant Maxim, Ants, a simple communication protocol, speech-act and the evolutionary MAS ecosystem Amalthaea, a distributed synchronisation algorithm, etc. [5,6].

4. CONCLUSION

The SLABS language integrates a number of novel language facilities that support the specification of agent-based systems. Among these facilities, the notion of caste plays a crucial role. A caste represents a set of agents in a MAS that have same capability of performing certain tasks and have same behaviour characteristics. Such common capability and behaviour can be the capability of speaking a language, using an ontology, following a communication and/or collaboration protocol, and so on. It is a notion that generalises the notion of types in data type and the notion of classes in object-oriented paradigm. This facility can be effectively used to specify or implement a number of notions proposed in agent-oriented methodologies, such as the notions of role, team, agent society, organisation, and so on. For example, a caste can be the set of agents playing the same role in the system. However, agents of the same caste can also play different roles especially when agents form teams dynamically and determine their roles at run time. Based on the caste facility, a number of other facilities in SLABS are defined. For example, the environment of an agent can be described as the agents of certain castes. A global scenario in a MAS can be described as the patterns of the behaviours of the agents of a certain caste. The example systems and features of agent-based systems specified in SLABS have shown that these facilities are powerful and useful for the formal specification of agents in various models and theories in a modular and composable way.

REFERENCES

[1] Jennings, N. R., Wooldridge, M. J. (eds.). Agent Technology: Foundations, Applications, And Markets. Springer, 1998.
[2] Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R., Treur, J. DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework, in Int. J. of Cooperative Information Systems 1(6) (1997), 67-94.
[3] Zhu, H. The role of caste in formal specification of MAS, in Proc. of PRIMA'2001, LNCS 2132, Springer, 1-15.
[4] Spivey, J. M. The Z Notation: A Reference Manual (2nd edition), Prentice Hall, 1992.
[5] Zhu, H. Formal Specification of Evolutionary Software Agents, Proc. of ICFEM'2002 (Shanghai, China, Oct. 2002), Springer LNCS 2495, pp249-261.
[6] Zhu, H. SLABS: A Formal Specification Language for Agent-Based Systems, Int. J. of Software Engineering and Knowledge Engineering 11(5) (Nov. 2001), 529-558.