Properties and Models of Software Agents and Agent Systems

Prof Dr Erika Horn, Mario Kupries, Dirk Glöde

Department of Computer Science
Am Neuen Palais 10
14469 Potsdam
Europe, Germany
{ehorn, kupries, gloede}@soft.cs.uni-potsdam.de

Abstract

In this article the foundations for software agents and agent systems will be laid from the point of view of Engineering Sciences by applying the basics of software architecture. For architecture modeling an object-oriented approach will be used. The components of an agent system will be mapped on to object-oriented model types. With the help of object-oriented methods special model types for interactions and adaptivity serving re-use will be derived. The sound theoretical considerations about software agents and agent systems and their object-oriented modeling and specification support and extend the functions described in the standardizations.

Software Architecture Types

The type of a software architecture (Horn and Schubert 1993) models software systems from the structural and operational point of view of the components, the relations between the components (connectors) and their composition (systems) with the appropriate operations. The common basic architecture type for software systems (Shaw and Garlan 1997) describes common properties of architecture types. The object-oriented software architecture modeling is combined with a number of advantages. The abstraction, detailing, composition and recurrence of architecture elements are semantically and syntactically representable. The explanation of structural and operational aspects is guaranteed. Agent systems are a special class of software systems. Below a model-type consideration of agent systems according to the object-oriented paradigm will be done.

Software Agents and Agent Systems

Agents are understood as computer programs employed autonomously and goal-orientedly on behalf of an authority for the fufilment of a special task (Nwana 1996). Hence it is not a question of automatic problem-solving but of the inclusion and the assistance of human beings and the situative adaptation of agent systems to changing environments. The common properties of software agents comprise life cycle, state, autonomy, locality, structural openness, authority, goal, safety and agent profile. Agent connectors describe the way agents communicate and interact with each other within the software architecture. For that, they have common properties, such as interface, protocol and filter. The interfaces of the agents have to correspond with the roles of the connectors. In the object-oriented sense the agent system is an instance of a composite class. This class defines agent classes, agent-connector classes and relations between them as well as the operations both for creating, executing, transferring and terminating agents and for establishing connections between authorities and agents and between agents themselves. Common properties of agent systems are agent system type, agent system profile, region affiliation, safety and ontology (Horn, Kupries and Glöde 1997).

Software Architecture Type ‘Agent System’

From the basic architecture type of software architecture an architecture type ‘agent system’ is derived with the help of object-oriented methods. Here software agents are derived as components, agents interactions as connectors and agent systems as systems with their properties and operations (figure 1). Agent systems are a very complex, highly developed form of software architecture. The architecture-type based approach is used for the analysis, synthesis and achieving theoretical soundness as well as for the systematic analysis of the properties, structure and classification of software agents and agent systems. From the common architecture type ‘agent system’ special architecture types (e.g. agent application system) are derived by the constructive application of object-oriented methods. This constructive application of architecture theory facilitates the pre-fabrication and re-use of agents, frameworks and patterns for agents and agent systems and the derivation of classes of agents, connectors and agent systems. The classes of the architecture type are indirectly derived from the root class of the basic architecture type (`CArchitectureRoot`) which reflects the common architecture properties, such as set creation and recurrence. In this abstraction agent systems can be again components (agents, connectors) of agent systems.

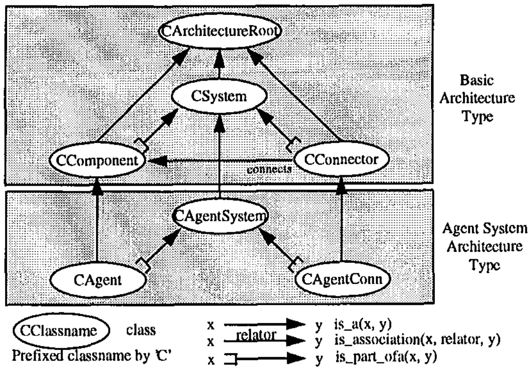

Figure 1 Architecture type 'agent system'

The architecture type-based approach is hence an engineering-scientific approach to the derivation of architecture types with their properties and operations for various classes of application systems and to the classification of their elements (Horn, Kupries and Glöde 1997). Such architecture types can be used for the classification of agent systems. They form the starting point for specialized development technologies for agent applications.

Object Oriented Modeling of Properties

Object-oriented modeling of properties of agents, connectors and agent systems is done through studies of historical sources (e.g. Agha 1986), standards (e.g. MASIF 1997) and current research (e.g. Jennings and Wooldridge 1998).

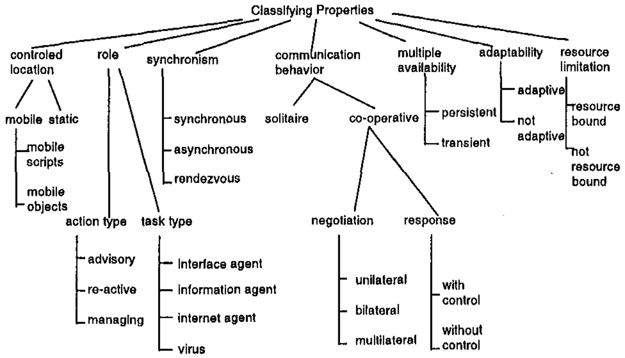

Figure 2 Classification of agents

Here common and classifying properties are analyzed (see chapter 2). The latter lead to a classification of agents in figure 2. In a further step the components of the architeture type are mapped on to object-oriented model types. Figure 3 represents a model type for connectors. From this model type further connectors for special classes of agent interactions, such as result-oriented co-operation patterns, are derived. Interactions between agents are adapted to the forms and patterns of co-operation between human beings. Hence goal-oriented interaction consists of permanently changing communication and task execution. Task execution describes the interdependent co-ordination of agents which is co-ordinated by communication.

According to its type the agent can do the job flexibly or schematically and use adequate forms of communication.

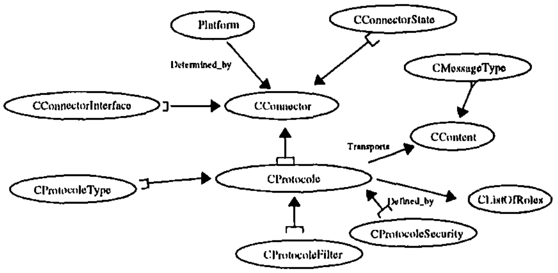

Figure 3 Object oriented model type 'connector'

Classification in Standardizations and Platforms

In the theoretical study of the properties of software agents and agent systems functionalities and interactions of software agents and agent systems have been described by technologies of software engineering (e.g. abstraction, aggregation, specializing).These findings are, for instance, contrasted with the standardizing effort MASIF and the IBM Aglets platform. It is impressive that the MASIF specification, which has emerged as an abstraction and a compromise of existing projects, and the IBM Aglets platform on the one hand have many similarities and at the same time - compared to the complexity and dynamic force of agent technology - shows incompletenesses according to the treatises. Examples (Horn, Kupries and Glöde 1997) which prove these facts are operations for regions (`ListRegions()`), the administration of rights (`SetPriority()`), multi-lingualism (`SetOntology()`) and interactions (`GetConnector()`). These concepts extend the points of view described in the standardizing effort and the platform.

References

Agha, G.A. 1986. *ACTORS: A Model of Concurrent Computation in Distributed Systems.* Cambridge MA.

Horn, E. and Schubert, W. 1993. *Objektorientierte Software-Konstruktion.* Wien, München: Hanser-Verlag.

MASIF. 1997. Mobile Agent Systems Interoperability Specification. OMG TC Document, orbos/97-10-05.

Shaw, M. and Garlan, D. 1996. *Software Architecture. Perspectives on an Emerging Discipline.* Upper Saddle River: Prentice Hall.

Nwana, H.S. 1996. Software Agents: An Overview. In *Knowledge Engineering Review*, Vol.11, No.3, 205-244, Cambridge University Press.

Horn, E.; Kupries, M.; and Glöde, D. 1997. Eigenschaften und Modelle von Software-Agenten und Agenten-Systemen. Technical Report, 006-97, Dept. of Computer Science, Potsdam University.

Jennings, N.R and Wooldridge, M.J. 1998 *Agent technology: foundations, applications, and markets.* Berlin: Springer.