Modelling Protocols for Multiagent Interaction by F-logic

Hong Feng Lai

Department of Business Management, National United University
1, LeinDa road, Miaoli city, 360, Taiwan
walden.lai@msa.hinet.net

Abstract. This paper proposes a method to represent agent Unified Modelling Language (AUML) in logic form using F-logic that provides deductive capability and uniform knowledge integration base. The AUML is used to differentiate relevant interaction more precisely at the analysis phase of developing a multiagent system. However, the AUML lacks for foundation and logic semantics. Thus we aim at constructing sufficient formality to facilitate formal analysis and to explore the behaviour and message route of the AUML. The AUML is transformed into F-logic language first by transformation rules. Secondly, a logic interpretation of this agent structure is presented. The transformation processes and results are illustrated using an example of E-commerce system. Finally, the significance of this approach is discussed and summarized.

Keywords: AUML, F-logic, message route, multiagent system, interaction protocols.

1 Introduction

As heterogeneous mobile devices continue to grow, how to integrate various types of information and knowledge is one of the most important issues in information technology. Since web technology has great potential to develop a collaborative environment, several strategies about information integration have been explored, e.g. centralized and open distributed strategies. However, centralized strategy has been shown not to be scalable. The open distributed strategy is growing and becoming difficult to manage. Beyond these methods, agent-based system with mobility is becoming a noticeable approach [1]. An agent is a computational process that implements the autonomous (actions without inputs), and communicating functionality of an application [2]. The agent-based system provides a convenient method to mobile users. Applying agent-based technologies, the services across web could be designed to be reactive, proactive, autonomous, and social.

In multiagent systems (MAS), agent interaction is ruled by interaction protocols. The agent Unified Modelling Language (AUML) is an extension of the UML, which proposes the standards for expressing the interactions of MAS. The interaction protocol (IP) diagram of AUML can help the designers to differentiate roles and messages between related agents more detailed. However, the IP diagrams do not guarantee the compliance of autonomous and heterogeneous agents to requirements [3].

T.-W. Kuo et al. (Eds.): EUC 2007, LNCS 4808, pp. 271–282, 2007.

A logical specification describes system requirements formally. Through formal semantics, it supports deductive capabilities that make specifications executable [4]. Mathematical foundations have been studied in several previous papers [5-8]. For instance, first-order logic is exploited to establish a deductive foundation for entity relationship model [5]. In [6], applying the Larch based logic [9] to represent the logical semantics of OMT. In [7], a scheme for integrating object-oriented and logic programming paradigms is proposed. In [8], dynamic master logic diagrams are used to represent time-dependent behavior and knowledge of a dynamic system.

Since the AUML lacks for foundation and logic semantics [10], in this study we aim at constructing sufficient formality to allow formal analysis and to verify the properties of the AUML. To embed deductive capability in the AUML, we transform the AUML into a logical specification language, F-logic, which is proposed by Kifer et al. [11].

How to transform the AUML into formal specifications is investigated in this study. The transformation framework between the AUML and F-logic is displayed in Fig. 1. The deductive AUML consists of three components: message space (a set of messages in F-logic form), role space (a set of roles in F-logic form), and the deductive rules for determining the message routes between message and role objects.

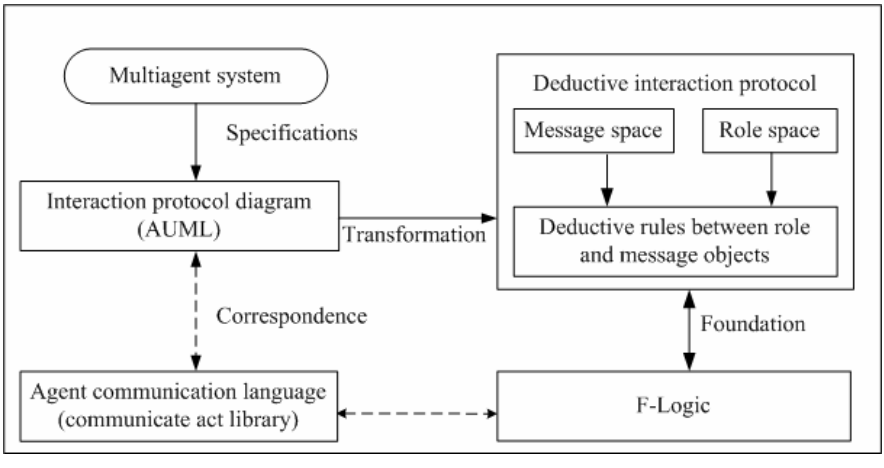

Fig. 1. The AUML/F-logic transformation framework

The structure of the paper is as follows. Section 2 gives an overview of F-logic. . Section 3 describes interaction protocol using AUML. The transformation rules and results between AUML and F-logic are presented in Section 4. Section 5 illustrates the related work. Section 6 concludes the paper.

2 F-Logic

In this section, we make a short summary of F-logic including its vocabulary and syntax. A comprehensive discussion of F-logic can be found in [11].

F-logic is a language with well-defined semantics that extends predicate logic and provides a sound and complete resolution-based proof procedure. This language is powerful in expressing object-oriented features. F-logic enhances the modeling capability of first order logic by syntactic enrichment, while it preserves its model-theoretic semantics by means of semantic structure. Elements are described by identification terms (id-terms), which consist of variables, functions, or constants, similar to terms in first order logic language.

The syntax of F-logic language, $\mathcal{L}$, consists of alphabetic symbols and the syntactic rules required to construct the well-formed formulas. The alphabetic symbols of F-logic language can be expressed in terms of its constituent parts:

a set of function symbols (object constructors), $\mathbb{C}$;
a set υ of variables;
a set $\wp$ of predicate symbols;
auxiliary symbols, such as: (,), [,], $\Rightarrow$, $\rightarrow$, =>>, $\twoheadrightarrow$, etc.;
and usual logical connectives and quantifiers, such as: $\wedge$ (and), $\vee$ (or), $\leftarrow$ (implication), $\neg$ (not), $\forall$ (universal), and $\exists$ (existential).

An id-term consists of constructor f (a member of $\mathbb{C}$) and object variable (a member of υ), similar to terms in first-order logic. For instance, f (X, g (a, Y)) is an id-term, where f and g are object constructors, 'a' is a constant, and X and Y are object variables. A ground F-term is a term not containing variables (variable-free). A ground F-term is denoted by a symbol that begins with a lower-case, while a symbol that begins with a capital letter denotes an id-term that may be non-ground.

An F-logic term (F-term) is defined as one of the following statements, which denote objects, classes, methods and predicates:

(1) An is_a assertion F-term A:b means that object A is a member of class b, or a::b denotes class a is a sub-class of class b. The is_a assertion enables attribute inheritance and subset relationship.

(2) A complex F-term is expressed as O[semicolon-separated list of method expressions], where O signifies an object or a class, and an method M expression can be either a scalar data expression: $M@A_1,\ldots,A_i \rightarrow R$, where $A_1,\ldots,A_i$, R is an id-term, or a set-valued data expression ($k \geq 0$): $M@A_1,\ldots,A_i \twoheadrightarrow <<S_1,\ldots,S_k>>$, and scalar signature expression $M@AT_1,\ldots,AT_i \Rightarrow (RT_1,\ldots,RT_k)$ or set-valued signature expression $M@AT_1,\ldots,AT_i=>>(RT_1,\ldots,RT_k)$, where, $A_1,\ldots, A_i$, $AT_1,\ldots,AT_i$ are arguments of method M, $R,S_1,\ldots,S_k$ denotes the output of the method M, $(RT_1,\ldots,RT_k)$ represents the types of the result of the method M. While a method does not need arguments, "@" will be omitted.

The implementation and application of F-logic can be found in several studies. FLORID [12] is a deductive engine for F-logic. In [13] FLORID with path expression is used to extract, restructure and manage the semi-structured web data. In [14, 15], they propose an operational knowledge specification language KARL, which contains two sublanguage Logic-KARL (F-logic) and Procedure-KARL. For specifying knowledge at conceptual and operational level, the domain layer and inference layer are expressed in Logical-KARL, while the task layer is represented by Procedure-KARL.

Thus, the KARL specification of intermediate representation can bridge the gab between an informal and an implementation of knowledge-based systems. In [16], a practical deductive object-oriented database system FLORA is provided, which integrates F-logic, Hilog and Transition logic, using complier optimization techniques to achieve its performance.

In this study we apply FLORID to express interaction protocols of multiagent system (MAS), i.e. to infer the roles and messages based on the deductive engine of FLORID.

3 The AUML

In this section, we make a brief introduction of AUML. A comprehensive discussion of AUML can be found in [10, 11].

3.1 Introduction to the AUML

The agent UML (AUML) is an extension of the UML, which proposes the standards for expressing the interactions of MAS. AUML applies graphical specification technique to describe interaction between agents. These approaches are partly based on the agent communication language (ACL) of the Foundation for Intelligent Physical Agents (FIPA) [2] using a subset of its communicative act library (CAL) of FIPA as messages.

3.2 Notations the Agent UML

An IP diagram indicates interactions between agents and roles along a timeline. Agents are assigned to roles. A role is a specification of the action that an object should fill. An object can switch roles at different times. Roles can be inserted or removed during the lifetime of agents.

Messages between agent roles are shown as arrows (Fig. 2) signifying an asynchronous communication. A diamond expresses a querying_if point that can result in zero or more communications. The line branch (no diamonds) indicates that all messages are sent concurrently. The empty diamond indicates that zero or more messages may be sent. A crossed diamond shows that exactly one message may be sent.

The message route (sequence of message) in Fig. 2 begins with a customer (initiator) which issues a request (cfp_P_order) to a manufacturer (participants). The manufacturer can reply proposing a price for satisfying the request (propose_proposal), or refusing (refuse_P_order). The customer must accept (accept_proposal) or reject (reject_proposal) the received proposals. After having received the cfp_P_order, the manufacturer must response to the customer by a given deadline, and informs the customer of propose or refuse_P_order. Analogously, the customer must response to the manufacturer by a given deadline. As the manufacturer receives the message of accept_proposal, the manufacturer must check “if stock was sufficient” (query_if_S_sufficient) whether the inventory level is true to satisfy the requirement of the customer.

The AUML diagrams are useful to analyze interactions between agents. Additionally, the agent UML can be taken as object interaction diagrams from the dynamic model viewpoint.

However, there are some limits in AUML [17]. These limits bring about extending or transforming the AUML to other model, e.g. cluttered AUML tends towards misinterpreting; unable to combine roles and cardinalities; indeterminable at decision points; hard to debug redundancy; unable to trace the history.

4 Transformation Rules and Deductive Rules

4.1 Introduction to the E-Commerce Example of a Multiagent System

In the E-commerce system**, t**he member agent customer and manufacturer in Fig. 2 invoke buyer agent and seller agent respectively. Analogously, the member agent vendor and manufacturer in Fig. 2 invoke seller agent and buyer agent respectively. The manufacturer switches roles at different times, i.e. facing customer as a seller and facing vendor as a buyer.

The message exchange in E-commerce can be modeled by mobile agent technology. A buyer agent could do purchasing for a customer, including making orders, negotiating, haggling, and potentially even paying.

A buyer agent can pass the customer's preferences to the host. If a potential match was met, the buyer agent could reply to the customer, or potentially finish the transaction delegated by the customer.

A seller agent must check (query_if_S_sufficient) whether the inventory level is true to satisfy the requirement of the customer, and negotiate price with buyer agent..

From implementation viewpoint, mobile agents are programs dispatching from one computer and transporting to a remote computer. As messages passing to a remote computer, the programs present their authorization and get access to local services and data. The remote computer may act as a broker by putting agents together with common interests and goals, and supporting a platform at which agents can coordinate.

4.2 Transformation Rules of the AUML/F-Logic

To create the message route of AUML IP diagrams involves a process of model transformation. Model transformation is a mapping from a source model to a target model using a set of transformation rules [4]. There is a natural correspondence between the AUML and F-logic. Based on the composite elements and notations in the AUML, the transformation rules from the agent UML into F-logic specifications are listed below.

The following two transformation rules express how to define AUML objects in F-logic form.

Trans_rule1. Each member_agent can be expressed as frame fields, and each role can be defined by F-logic as follows.

member_agent[has_role=>>role; name=>string; type=>string].
role[name=>string; type=>string; use=>string].

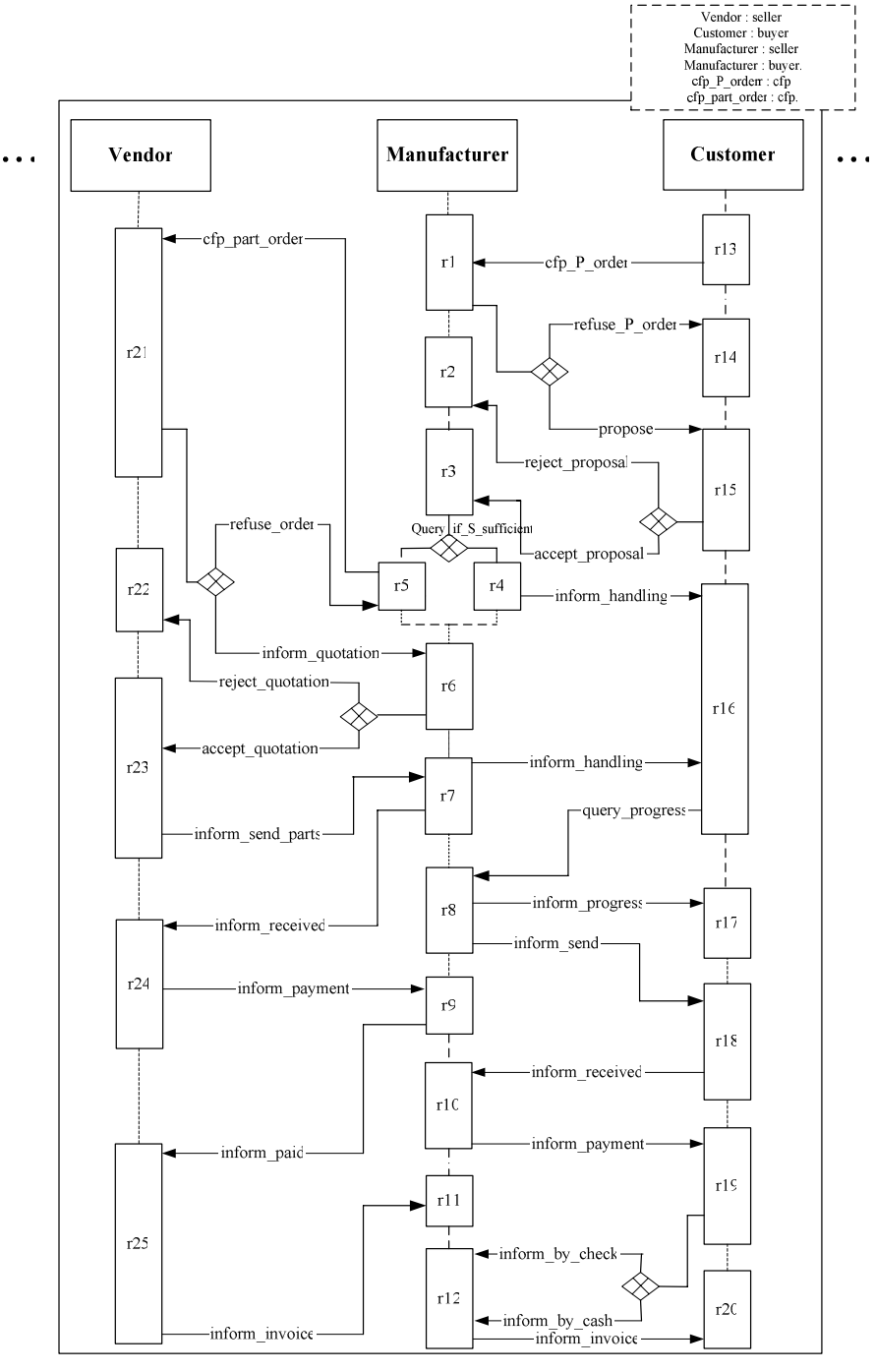

Fig. 2. The interaction protocol in E-commerce

Trans_rule2. Each message can be described via its sender, receiver, name, and type. The types of message includes: resouce_agent_role, delegation_agent_role, wrapping_agent_role, coordination_agent_role, and discovery_agent_role. The message and message_route can be defined by F-logic as follows:

message[sender=>role; receiver=>role; name=>string; type=>string].
message_route[sender=>role; receiver=>role;
add@(message)=>message_route;
add@(message_route)=>message_route].

The following two transformation rules express message and agent_role hierarchical relationships in F-logic form respectively.

Trans_rule3. For each message and its subclass, the relationship can be represented by 'is_a assertion'. This property can be denoted as follows:

refuse :: messgae.
reject :: messgae.
accept :: messgae.
inform :: messgae.
propose :: messgae.
query :: messgae.

Trans_rule4. For each agent_role and its subclass, the relationship can be represented by 'is_a assertion'. This property can be denoted as follows:

agent_role :: role.
resouce_agent_role :: agent_role.
delegation_agent_role :: agent_role.
wrapping_agent_role :: agent_role.
coordination_agent_role :: agent_role.
discovery_agent_role :: agent_role.

The resource_agent_role is used to manages local resources. The delegation_agent_role is used to invoke agent service. The wrapping_agent_role is used to transfer the coordination. The report_agent_role is used to support summarizing and reporting service. The discovery_agent_role is used to discover available external services.

Trans_rule5. For each vertical bar in AUML corresponds to a agent_role in F-logoic as follows:

agent.role : agent_roleType.

The following three transformation rules express how to transform the asynchronous message → of AUML in F-logic form.

Trans_rule6. For each message arrow of AUML can be expressed in F-logoic as follows.

message_name:message[sender→agent1.role_i; receiver→agent2.role_j; type→ message_type].

Trans_rule7. For each message with diamond arrow can be expressed in F-logoic as follows.

xor_message_name:message[sender→agent1.role_i; receiver→agent2.role_j; type→ message_type].

Trans_rule8. For each message with line branch (no diamonds) and empty diamond of AUML can be expressed in F-logoic as follows.
message_name:message[sender→agent1.role_i; receiver→agent2.role_j; type→ message_type].

4.3 The Deductive Rules of the AUML/F-Logic

Based on the transformation rules, the AUML of the E-commerce example (see Fig. 2) can be transformed into F-logic form. The deductive AUML will be expressed in terms of agent role space (a set of agent roles in F-logic), message space (a set of messages in F-logic), structural assertions, and deductive rules.

The deductive rules express the relation between roles, message, and message_route in AUML IP diagrams.

Deductive_rule1. These rules express how to add messages to message_routes as follows.

P.E:message_route[sender→X; receiver→Z] :-
P:message_route[sender→X; receiver→Y], E:message[sender→Y; receiver→Z].

Deductive_rule2. This rule expresses how to concatenate message_routes to a new message_route as follows.

P1[(P2.E)→P3] :- P1.P2[E→P3], E:message.

Deductive_rule3. These rules express how to detect loop routes and eliminate loop routes in a message_route as follows.

P:loop:- P:message_route[sender→P.receiver].
P.C = P :- P.(C:loop)[].

4.4 The Query of the Logic-Based E-Commerce System

After implementing the deductive AUML, the logic-based E-commerce system consist of a set of agent roles, a set of messages, a set of structural assertions, and some deductive rules about these elements. Various types of queries can be evaluated and answered by FLORID. For example, the query "?- M::message" state that "are there any sub-class of message".

% Answer to query : ?- M::messgae.
M/refuse
M/messgae
M/reject
M/accept
M/inform

M/propose
M/query
M/cfp

The query "?- R::agent_role" state that "are there any sub-class of agent_role". The answers are as follows:

% Answer to query : ?- R::agent_role.
R/agent_role
R/resouce_agent_role
R/delegation_agent_role
R/wrapping_agent_role
R/report_agent_role
R/discovery_agent_role

The message route is the sequence of messages passing to and fro on the IP diagrams. These behaviour properties can be also described by reachability tree in Petri net; or by message sequence chart in MSC [18]. However, a logic-based IP can provide more information, e.g. finding the message route between any two agent roles. For example, the query stated as "?- P:message_route[sender -> ven1.r21; receiver -> ven1.r25]" means that what the message route is between agent role ven1.r21 and agent role ven1.r25. The answers are as follows:

% Answer to query : ?- P:message_route[sender -> ven1.r21; receiver -> ven1.r25].
P/ven1.r21.xor_propose_quotation.(man1.r6.xor_accept_quotation).(ven1.r23.inform_sreceiver_parts).(man1.r7.inform_received).(ven1.r24.inform_payment).(man1.r9.inform_paid)
P/ven1.r21.xor_refuse_order.(man1.r5.cfp_part_order).(ven1.r21.xor_propose_quotation).(man1.r6.xor_accept_quotation).(ven1.r23.inform_sreceiver_parts).(man1.r7.inform_received).(ven1.r24.inform_payment).(man1.r9.inform_paid)
% 2 output(s) printed

5 Related Work

To express and coordinate the activities of multiagent systems, two types of approaches have been proposed: graphical and predicate approaches. The graphical approach using diagrammatic notation for intuitive understanding includes: agent UML approach [10], statechart approach [17], message sequence chart approach [18], Petri net approach [19, 20]. The textual approach using rules and declarations for consistency checking includes OMG IDL (Interface Description Language) [1] and logic-based approach [3, 17, 21].

Statecharts is a visual specification language for specifying discrete event system. It extends finite state machines that was proposed by [22] and could be described as: Statecharts = finite state machine + depth + orthogonality + broadcast communication. Statecharts has many good properties that can be applied in object-oriented information system. However, there are still some restrictions in MAS application domain. Many extensions had been proposed for improving their descriptive ability. In [23] they apply

propositional dynamic logic (PDL) to extend the description capability of statecharts for presenting interaction protocol.

The message sequence chart (MSC) diagrams are used to express basic protocols and scenarios in telecommunication systems [18]. An MSC diagram consists of a set of instances, which indicate that events may occur during the execution period. The types of events may be the creation and stopping of an instance, the trigger of a local service, the setting or resetting of a timer, the timeout, the sending or receiving of a message, etc. The main difference between MSC and MAS is that MSC using an axe represents a process of an instance, while MAS using isolated vertical bars signifies multi agent roles.

Petri net [24] is the most frequently used tool for modeling systems. Petri nets and Statecharts have equivalent representative capabilities because they are both state-based models [25]. Applying PN to AUML modelling, the message is taken as a place; the xor-message is expressed by a conflicting place; and the agent role is represented by a transition [20]. The limits of IP in PN include: hard to read, limits in model transformation, the problems of scalability and reusability [23].

The textual approaches using rules and declarations define the interaction of agents. From object-oriented viewpoint, the MAS could be taken as a set of interacting objects. To express the static and dynamical specifications of agents and roles in the agent-enhanced mobile virtual communities [1], they apply OMG IDL to differentiate agents and roles using interface specification sketch including require interfaces, provide interface, behaviour interface, and policy interface. This method resembles requirement decomposition while it lacks of deductive capability and can not check the consistency of specifications.

To verify the compliance of agents' behaviour to protocols, in [3] a logic-based formalism Social Integrity Constraints using Java-Prolog-CHR (Constraint Handling Rules) is proposed. An example of FIPA Contract-Net protocol is specified and verified by this approach. To guarantee the global properties of procedurally constructed MASs, in [21] a generalized linear temporal logic (GLTL) based agent system is constructed. The workflow properties (similar to the message route in this paer) are represented as temporal logic formulas and consequently can be verified by model checker.

The above related work illustrates the various types of approaches for presenting interaction protocol. In our approach, AUML/F-logic can be taken as a schema transformation that transforms conceptual level to operational level. Also, it plays a role of mediator for integrating objects in heterogeneous systems [26].

6 Conclusion

With the growing complexity of multiagent interaction in web-based applications, the requirement of tools and techniques for representing mobile agent is growing in the same way. This paper proposed a method to produce F-logic specifications for AUML that extends its expressive power, and provides syntax, semantics, and inference rules.

This approach was illustrated using an example of electronic commerce systems, which expressed how the various features of F-logic could be applied in AUML. These logical specifications provided more reasoning power. Additionally, the formal specification language is easily adapted to the system requirements.

The future work will explore the logic specifications of dynamical model of electronic commerce systems, i.e. in a working environment such that new rules can be inserted into the system. The transformation should reflect the corresponding logic specifications and support the transformation rules.

Acknowledgment. Financial support for this work was provided by the National Science Council Taiwan, under the contract NSC94-2416-H-239-003.

References

1. Loke, S.W., Rakotonirainy, A., Zaslavsky, A.: An enterprise viewpoint of wireless virtual communities and the associated uses of software agents. In: Rahman, S.M. (ed.) Internet Commerce and Software Agents: Cases, Technologies and Opportunities, pp. 265–287. Idea Group Publishing, Hersey, PA, USA (2001)
2. FIPA.: FIPA Agent Management Specification. Foundation for Intelligent Physical Agents(2002), //www.fipa.org
3. Alberti, M., Daolio, D., Torroni, P., Gavanelli, M., Lamma, E., Mello, P.: Specification and verification of agent interaction protocols in a logic-based system. In: Proceedings of the 2004 ACM symposium on Applied computing, pp. 72–78 (2004)
4. Mineau, G.W., Missaoui, R., Godinx, R.: Conceptual modeling for data and knowledge management. Data & Knowledge Engineering 33, 137–168 (2000)
5. Battista, G.D., Lenzerini, M.: Deductive entity relationship modeling. IEEE Transactions on Knowledge and Data Engineering 5, 439–450 (1993)
6. Bourdeau, R.H., Chen, B.H.C.: A formal semantics for object model diagrams. IEEE Transactions on Software Engineering 21, 799–821 (1995)
7. Lee, J.H.M., Pun, P.K.C.: Frame logic integration: A multi paradigm design methodology and a programming language. Computer Languages 23, 25–42 (1997)
8. Hu, Y.-S., Modarres, M.: Time-dependent system knowledge representation based on dynamic master logic diagrams. Control Engineering Practice 4, 89–98 (1996)
9. Guttag, J.V., Horning, J.J.: Larch: Languages and tools for formal specification. Springer, Heidelberg (1993)
10. Bauer, B., Muller, J.P., Odell, J.: Agent UML: A formalism for specifying multiagent interaction. In: Cuabcarubu, P., Wooldridge, M. (eds.) Agent-Oriented Software Engineering, pp. 91–103. Springer, Heidelberg (2001)
11. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. Journal of the Association for Computing Machinery 42, 741–843 (1995)
12. FLORID Homepage (2006), http://dbis.informatik.uni-freiburg.de/
13. Ludäscher, B., Himmeröder, R., Lausen, G., W.M., Schlepphorst, C.: Managing semistructured data with florid: a deductive object-oriented perspective. Information systems 23, 589–613 (1998)
14. Fensel, D.: Graphical and formal knowledge specification with KARL. In: Proceedings of the the International Conference on Expert Systems for Development, pp. 198–203 (1994)
15. Fensel, D., Angele, J., Studer, R.: The Knowledge acquisition and representation language, KARL. IEEE Transactions on Knowledge and Data Engineering 10, 527–550 (1998)
16. Yang, G., Kifer, M.: FLORA: Implementing an efficient DOOD system using a tabling logic engine. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 1078–1093. Springer, Heidelberg (2000)

17. Paurobally, S., Chachkov, S., Jennings, N.R.: Developing agent interaction protocols using graphical and logical methodologies. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS (LNAI), vol. 3067, pp. 149–168. Springer, Heidelberg (2004)
18. Rudolph, E., Grabowski, J., Graubmann, P.: Tutorial on message sequence charts (MSC). In: Proceedings of the FORTE/PSTV 1996 Conference (1996)
19. Ling, S., Loke, S.W.: Advanced Petri Nets for modelling mobile agent enabled interorganizational workflows. In: Proceedings of the Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pp. 245–252 (2002)
20. Ling, S., Loke, S.W.: Engineering Multiagent Systems Based on Interaction Protocols: A Compositional Petri Net Approach. In: Camp, O. (ed.) Enterprise Information Systems V, pp. 279–285. Kluwer Academic, Netherlands (2004)
21. Pokorny, L.R., Ramakrishnan, C.R.: Modeling and verification of distributed autonomous agents using logic programming. In: Leite, J.A., Omicini, A., Torroni, P., Yolum, p. (eds.) DALT 2004. LNCS (LNAI), vol. 3476, pp. 148–165. Springer, Heidelberg (2005)
22. Harel, D.: Statecharts: a visual formalism for complex systems. Science Computer Program 8, 231–274 (1987)
23. Paurobally, S., Cunningham, R., Jennings, N.R.: Developing agent interaction protocols using graphical and logical methodologies. In: Workshop on Programming MAS, AAMAS (2003)
24. Peterson, J.L.: Petri-Net Theory and Modeling of Systems. Prentice-Hall, Englewood Cliffs (1981)
25. Bucci, G., Campanai, M., Nesi, P.: Tools for Specifying Real-Time Systems. Real-Time Systems 8, 117–172 (1995)
26. Wiederhold, G.: Mediators in the architecture of future information systems. Computer 25, 38–49 (1992)