

Agile Integration Modeling Language (AIML): A conceptual modeling grammar for agile integrative business information systems[☆]

Hong Zhang^{a,1}, Rajiv Kishore^{b,*,2}, Raj Sharman^{b,2,3}, Ram Ramesh^{b,4}

^a Computer Information Systems, Glass Hall 376, Missouri State University, 901 S National Ave, Springfield, Missouri 65897, United States

^b Department of Management Science and Systems, School of Management, The State University of New York at Buffalo, Buffalo, NY 14260-4000, United States

Received 7 February 2006; received in revised form 16 March 2007; accepted 23 April 2007

Available online 13 May 2007

Abstract

The proliferation of newer agile integrative business information systems (IBIS) environments that use the software agent and the multiagent systems paradigms has created the need for a common and well-accepted conceptual modeling grammar that can be used to efficiently, precisely, and unambiguously, model agile IBIS systems at the conceptual level. In this paper, we propose a conceptual modeling grammar termed Agile Integration Modeling Language (AIML) based on established ontological foundation for the multiagent-based integrative business information systems (MIBIS) universe. The AIML grammar provides adequate and precise constructs and semantics for modeling agile integration among participating work systems in terms of quickly building and dismantling dynamic collaboration relationships among them to respond to fast-changing market needs. The AIML grammar is defined as a formal model using Extended BNF and first order logic, and is elaborated using a running example in the paper. The grammar is also evaluated in terms of its syntactic, semantic, and pragmatic qualities and is found to exhibit a high degree of quality on all these three dimensions. In particular, the pragmatic quality of AIML measured in terms of grammar complexity evaluated using complexity metrics indicates that AIML is much easier to learn and use as compared to the Unified Modeling Language (UML) for modeling agile integration of work systems in organizations.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Conceptual modeling grammar; Systems modeling; Requirements specifications; Role-based modeling; Multiagent systems modeling; Integrative business information systems modeling

[☆] The authors are grateful to Vijayan Sugumaran, participants at the SOM-Informatics Working Group seminar at SUNY Buffalo, and anonymous reviewers and attendees at the AMCIS 2003 and HICSS 2004 conferences for providing invaluable suggestions during the course of development of this paper. We are also grateful to Sukhamay Kundu for his critical evaluation of the AIML formal grammar included in the paper.

* Corresponding author. Tel.: +1 716 645 3507; fax: +1 716 645 6117.

E-mail addresses: HongZhang@MissouriState.edu (H. Zhang), rkishore@buffalo.edu (R. Kishore), rkishore@buffalo.edu (R. Sharman), rramesh@acsu.buffalo.edu (R. Ramesh).

¹ Tel.: +1 417 836 4121; fax: +1 417 836 6907.

² The second and third authors have contributed equally to this paper and their names are listed in the alphabetical order.

³ Tel.: +1 716 645 3507; fax: +1 716 645 6117.

⁴ Tel.: +1 716 645 3258; fax: +1 716 645 6117.

1. Introduction

Information systems have been playing an important role in supporting business integration. Traditional integrative business information systems (IBIS) [18,19] such as ERP, EAI, and workflow management systems have brought significant benefits to businesses in terms of improved planning, timely deliveries, reduced inventories, reduced costs, and responsive and improved customer service [19]. However, they take significant amounts of time and effort to develop as all their component work systems⁵ are tightly coupled to each other. Tight coupling also results in difficulties in adding new and modifying or deleting existing collaborative relationships among participating work systems. On the other hand, current hypercompetitive business environment requires business organizations to quickly build as well as dismantle dynamic collaboration relationships among various participating work systems, both internally and externally, to respond to fast-changing market needs.

Consequently, newer agile IBIS systems⁶ should allow participating work systems to integrate with each other while preserving their local autonomy and coordinating in a decentralized manner. In an agile IBIS system, we envision no overall control over participating work systems and integration occurs through dynamic coordination among the participating work systems, in addition to the necessary integration at the technology and data levels [19]. In such a scenario, business processes are rather dynamic and emergent, relying on the judgments and decisions of individual work systems. To achieve this type of integration, an IBIS system should allow participating work systems to reach agreements about service contracts on their own without central control. Further, users and developers should be able to configure various business processes and collaboration relationships among work systems dynamically as the goals and needs of the organization change.

Recently, multiagent systems comprising of collaborating software agents have emerged as a new technology to solve complex problems in a distributed environment [40]. A software agent is “a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives” [41]. An intelligent software agent

within a multiagent system is autonomous (i.e., it acts without human intervention), reactive (i.e., it responds to events in the environment), proactive (i.e., it is goal-directed), and social (i.e., it interacts with other software agents to get what it needs) [41].

The multiagent systems paradigm has a number of parallels with the IBIS paradigm as noted in earlier work [19]. Further, due to their dynamic coordination and collaboration capabilities, software agents in multiagent systems are uniquely capable of supporting integration in agile IBIS systems as they provide flexibility in resolving inconsistencies and dependencies involved in work systems coordination. Further, software agent as a modeling paradigm minimizes the semantic gap between work system coordination and information system modeling [14]. As a result, software agents have been adopted by a number of researchers in various IBIS applications such as e-commerce [e.g., [16,28]], business process management [e.g., [15,21]], supply chains management [e.g., [12,33]], enterprise integration [e.g., [22,32]], and manufacturing [e.g., [10,20]] to create more agile integrative environments.

This proliferation of newer agile IBIS environments that use the software agent and the multiagent systems paradigms has created the need for a common and well-accepted conceptual modeling grammar that can be used to efficiently, precisely, and unambiguously, model agile IBIS systems. This is because existing modeling grammars such as ERD, object-oriented techniques, and business process modeling techniques lack requisite capabilities to model autonomy, intelligence, and agile coordination and collaboration that are central to agile IBIS systems. Most agent-oriented systems development methodologies also do not have conceptual-modeling-level constructs formally defined with unambiguous semantics. Further, researchers and designers that apply the software agent principles and concepts in various agile IBIS applications have defined their own unique and application-specific constructs and rules for conceptual modeling of agile IBIS systems which limits knowledge sharing and reuse in the agile IBIS community and may result in compatibility issues among future IBIS systems. In response to this state of craftsmanship, Kishore et al. [19] synthesized literature in the areas of IBIS systems and multiagent systems with the intent of developing a comprehensive foundation ontology for the universe of Multiagent-based Integrative Business Information Systems (MIBIS) that can become the basis for a sound conceptual modeling grammar for a variety of agile IBIS systems.

We extend Kishore et al.'s work and develop a formal conceptual modeling grammar termed *Agile Integration*

⁵ A work system is “a system in which human participants and/or machines perform a business process using information, technology, and other resources to produce products and/or services for internal or external customers” [2].

⁶ We use the term IBIS system in this paper for ease of reading even though the IBIS acronym includes the initial S for the term *system*.

Modeling Language (AIML)⁷ for modeling of agile IBIS systems that belong to the MIBIS universe and utilize the notions and principles of multiagent systems for agile integration. This grammar is defined in ISO/IEC 14977 Extended BNF [13] and formally specified in first-order logic. AIML facilitates MIBIS modeling in several ways. First, AIML serves as a foundation ontology for MIBIS knowledge representation, which can be shared and reused during the process of system analysis and design. Second, the formally defined AIML constructs serve as templates to be customized and enhanced for individual MIBIS applications. Finally, the rigorous formalism of AIML serves as an analytical tool for developers to detect and avoid possible conflicts in MIBIS modeling.

The paper is organized as follows. In Section 2, we summarize the conceptual framework for the MIBIS universe described by Kishore et al. [19] and discuss the need for a conceptual modeling grammar for this universe. In Section 3, the AIML grammar is elaborated, formally defined, and explained through examples. In Section 4, we evaluate the quality of the AIML grammar. Finally, Section 5 concludes the paper with remarks on some future research directions.

2. The MIBIS conceptual framework

MIBIS is an information system that uses software agents to support coordination among multiple work systems. Coordination is defined as “managing dependencies between activities” [24]. In MIBIS, dependencies are resolved through interactions between agents.

⁷ In earlier versions of this paper and in other related papers, we had termed this conceptual modeling grammar as MibML (for Multiagent-based Integrative Business Modeling Language). However, recently through discussions with some colleagues in the field, it became clear to us that the proposed grammar had a much wider applicability than just in multiagent-based IBIS systems. For example, a colleague informed us that he is using the eight ontological constructs that we identified in our earlier MIBIS paper for analyzing and modeling computer games through roles, interactions, etc. but without the use of multiagent systems. We also became aware that other colleagues in the field are using the MIBIS ontological constructs in business process management applications. Therefore, we have now renamed the grammar as AIML to capture the essence of agility that is provided by multiagent systems and that is needed in newer integrative systems. We removed the term multiagent from the name of the grammar because multiagent systems provide just one way to implement the notions of agile integration. This renaming will allow the further extension and development of the current grammar for other agile integration universes without the need for creating a new grammar for those domains. However, we continue to discuss the AIML grammar in this paper in the context of MIBIS systems to draw from and build upon previous work.

Fig. 1 describes the conceptualization of a MIBIS system. Multiple work systems are brought together by a MIBIS to form a business enterprise⁸ so they all can contribute to and benefit from the accomplishment of various business goals. Each work system is composed of software agents and information resources. Some agents work as interfaces to interact with other work systems and to exchange information with users and other entities in the MIBIS environment. Agents in MIBIS are intelligent in the sense that they make real-time decisions based on their own knowledge and information. They control what information to share, who to interact with, and how to negotiate in order to maximize their own benefits.

Kishore et al. [19] investigated various modeling paradigms in the IBIS domain and systems development methodologies in the multiagent systems domain to assess their suitability for modeling of MIBIS systems, as conceptualized above. Based on this review and synthesis, they identified a set of eight ontological constructs for MIBIS modeling that are minimally required to model a system efficiently, precisely, and unambiguously in the MIBIS universe. These eight constructs are *agent*, *role*, *goal*, *interaction*, *task*, *resource*,⁹ *information*, and *knowledge*. Due to length limitations, we only provide a brief discussion about these constructs and a detail elaboration can be found in [19].

Agent is a central construct in MIBIS modeling. From a modeling perspective, agent provides a much higher-level abstraction than the traditional object concept for modeling human actors in business organizations. Unlike objects, agents are able to control their behaviors through their “mentalist” components such as knowledge, belief, intention, and obligation. In addition, agents engage in conversations instead of single-message exchanges in object communications. From a system architecture perspective, multiagent systems provide excellent support for the distributed, decentralized, and complex IBIS environment. As a result, a MIBIS system can mimic the social system to a great extent by introducing the notion of agent into MIBIS modeling.

While agents are transient both in organizations as well as on computer network in the sense that they may join (appear) and leave (disappear), role is a more

⁸ A business enterprise can be a single business organization or it can be an extended, networked, or virtual strategic alliance.

⁹ “Resource” construct is not included in our discussion because its semantics overlaps with the “information” construct in the conceptual modeling context. Information about resources rather than resources themselves are a matter of concern when we are engaged in conceptual modeling of work systems and business integration.

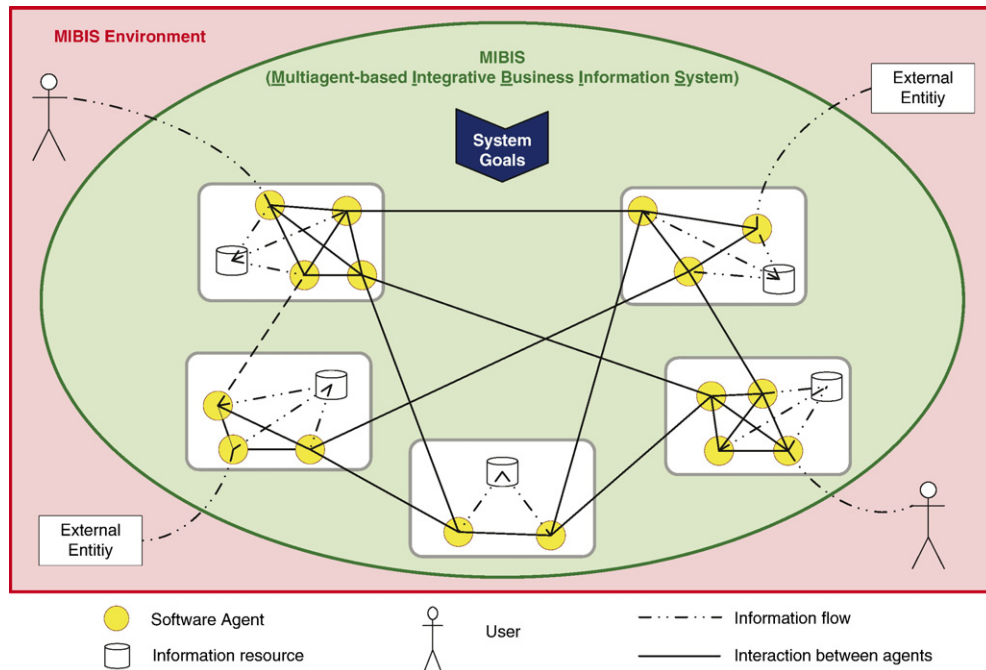


Fig. 1. The conceptualization of MIBIS (adapted from Kishore et al. [19]).

enduring concept that introduces the organizational view into MIBIS. Business organizations define roles, such as that of a purchase manager, a master technician, etc., as templates for the work these abstract entities will perform and goals they will achieve. In systems modeling, role has been used to decouple business process definitions from concrete resources such as physical individual actors [3], and provides “a new abstraction that can unify diverse aspects of a system” [42]. In this sense, inclusion of the role concept in MIBIS modeling provides the advantages of design focus, reusability, and flexibility. Further, the notion of autonomous and loosely-coupled roles with their own set of tasks provides the necessary capability for achieving agile integration as roles can be added, modified, or removed as necessary to meet dynamic business needs.

A system that supports business integration brings together work systems that share the same overall business goals and contribute towards achieving those goals. In other words, goals provide the *raison d'être* for coordination among work systems and, thus, for MIBIS systems. In view of its importance, goal has been identified as an essential concept for capturing user requirements in systems modeling. It is capable of aiding in the elicitation and elaboration of requirements, relating system requirements to organizational and business contexts, clarifying requirements, and dealing with conflicts [43]. Accordingly, information systems developed based on goals are

more stable than those based on functions, processes or information structures that often change with time [7]. At the same time, the notion of goals provides another mechanism for achieving integration agility as it is business goals that change in dynamic environments.

One of the central problems in work systems integration is to resolve inconsistencies or conflicts caused by goal dependencies, task dependencies, and resource dependencies. Interaction is one of the basic means for managing and resolving dependencies. Through interactions, agents are able to communicate with each other for sharing resources, checking for task/goal accomplishment, checking for availability of other agents for performing certain tasks, negotiating prices and timelines, subcontracting tasks to other agents, deciding upon future courses of action, and constructing dynamic business processes to reach system goals, etc.

Task is fundamental to business integration and practically all IBIS modeling techniques provide for modeling of tasks, activities, or processes. It is also an essential concept in multiagent systems. Each agent is responsible for performing some tasks to solve problems. Therefore, task is included as a foundation construct for MIBIS modeling.

A variety of information is required by work systems to perform their tasks in order to accomplish their goals. Information exchanges also take place between work systems in integrative business systems to coordinate

their activities and resources. Agents require information as inputs in order to make context-dependent real-time decisions to manage and resolve interdependencies. Therefore, information is fundamental to MIBIS modeling.

Knowledge is a personal justified belief of an entity (a human actor or software agent) rather than an absolute and static true belief (for all entities at all times), following recent and prominent viewpoints about organization knowledge [11,26]. Knowledge is fundamental to MIBIS modeling because it represents “mentalistic” characteristics of software agents. Software agents can possess declarative knowledge (know-that) and/or procedural (know-how). Declarative knowledge is what an agent believes about itself, other agents, and its environment, while procedural knowledge represents business rules that control how the agent performs its tasks and interactions. Knowledge is different from information in several perspectives. First, declarative knowledge is context-related, run-time data internal to an agent. Agents have full control of their declarative knowledge. Information on the other hand is external to the agent. Second, through incoming information agents may revise their current beliefs and form new beliefs, thereby adding to, restructuring, or changing their current declarative knowledge. Further, agents make decisions based on their knowledge and not simply based on information. For example, with incomplete information on a competitor’s pricing strategy, a sales agent may form a belief that the current price is the lowest price that its competitor will offer and may, thus, offer a 5% discount on the current price to attract a customer. Third, unlike procedural knowledge, information cannot govern agents’ behaviors.

As mentioned earlier, in order to model a MIBIS system efficiently, precisely, and unambiguously as well as to facilitate MIBIS knowledge sharing and reuse, there is a need for a formal conceptual modeling grammar that has adequate power to capture and represent the above MIBIS foundational constructs and their semantics. However, there is currently a lack of such a conceptual modeling grammar in the literature. Traditional general-purpose modeling grammars do not capture all the constructs and their semantics that are necessary for the MIBIS universe. Traditional modeling techniques such as ERD and DFD focus only on limited information and process perspectives but ignore a number of behavioral and coordination perspectives. Process-oriented techniques such as Petri-nets are primarily oriented towards analysis of task timing and conflict resolution considerations in stable business processes but do not consider interaction and coordination that are

essential in agile integrative business systems. Object-oriented modeling techniques may be useful for defining MIBIS specifications in terms of objects, but they provide no explicit support for the agent, role, knowledge, and extended agent interaction concepts. It is, therefore, difficult to model agent knowledge and interactions inherent to MIBIS systems. While several methodologies (e.g., Gaia, MaSE, etc.) exist for multiagent systems analysis and design, they all utilize unique constructs with unique semantics leaving little room for knowledge sharing and knowledge reuse. Further, these methodologies also do not explicitly define the constructs underlying their models and their semantics in a formal manner. Similarly, in the context of IBIS systems, most enterprise and workflow models have been criticized for the lack of essential constructs and semantics to concisely and precisely represent specific activities, tasks, business processes, business goals, and organization structures of a business [29,38]. Thus, there is a need for a new conceptual modeling grammar that can be used to efficiently, precisely, and unambiguously analyze and model at the conceptual level an agile IBIS system in the MIBIS universe. Such a grammar termed *Agile Integration Modeling Language* (AIML) is formally developed and discussed next.

3. The AIML Grammar

The AIML grammar is a further refinement and extension of the MIBIS foundational constructs discussed in §2 above. This grammar formalizes the constructs and defines their representation schema in ISO/IEC 14977 Extended BNF and first order logic. Relationships among the constructs and the various constraints are established in the grammar as well. We also provide a simple but comprehensive example of an agile IBIS system for a fictitious online retailer that sells made-to-order computers. We intersperse the example throughout the section to illustrate the grammar. The IBIS systems involves three participating work systems: *Sales* (for managing orders), *Factory* (for PC assembly), and *Shipping* (for order delivery). When the *Sales* work system receives an order from a customer, it tries to assess from the *Factory* and *Shipping* work systems whether they will be able to satisfy the customer’s requirements. If *Factory* and *Shipping* systems indicate that the order is not possible to be met, the *Sales* work system will decline the customer’s order. The *Sales* work system is also responsible for tracking the order status once it is accepted. Our simplified example demonstrates how AIML can help capture and model the agile integration of these three work systems at the conceptual level.

3.1. The AIML Fundamentals

As conceptualized in §2 above, the AIML grammar regards a MIBIS system as a goal-oriented, role-centric, and agent-based information system. Corresponding to system development stages, the AIML grammar uses the notion of goal to capture system requirements, uses the notion of role to assign responsibilities for the identified goals and to define coordination mechanisms between work systems to accomplish those goals, and uses the notion of agent to implement roles in MIBIS applications.

As a central concept in MIBIS conceptual modeling, role is the focus of the AIML grammar. In the classic role theory, a role is defined as “a collection of duties and rights” [4]. In AIML, a role is also a collection of duties and rights. Duties of a role in AIML involve performing *tasks* and *interactions*. A task is a series of processing acts performed by a single role alone to pursue its goals while an interaction is a series of communicative acts that form a conversation between at least two roles to resolve interdependencies. Where there is no need to differentiate between these two types of acts, we refer to both tasks and interactions as activities. Each role, thus, performs tasks individually and interacts with one or more roles in the system. The rights of a role in AIML pertain to access rights for information that may be required for performing tasks and interactions. A role in AIML also possesses knowledge so it can make decisions that are necessary for the tasks and interactions at hand and thereby achieve its goal. The notion of information is used to model inputs required and outputs generated during the course of task

Table 1

Convention of notations used in AIML specifications

	Individual	Collection
Type	Bolded lower case letter(s)	Bolded upper case letter/(s)
Instances	Lower case letter(s)	Upper case letter/(s)

performance by a role. The notion of knowledge is used to represent a role’s, and thereby an agent’s, internal beliefs and business rules that govern its behaviors. Fig. 2 depicts these fundamentals in a meta-model of the AIML grammar. In essence, a role provides the complete contextual knowledge and relationship of an agent with all other entities within a MIBIS system. A role is an abstraction for the tasks it needs to perform and the interactions it needs to have with other roles to achieve its individual goal, the information that it needs to access or it will generate during the course of performance of its tasks and interactions, and the knowledge that it needs for the successful execution of its tasks and interactions and for the successful achievement of its assigned goal. The key features of the AIML grammar are, therefore, role centrality, goal orientation, interaction focus (for coordination), and knowledge encapsulation (for autonomous behavior) and it is these features that allow AIML with the necessary capabilities for modeling agile integration in an effective manner.

3.2. Formal Specification of AIML

In this section, we define the AIML constructs as a set of schemas using *ISO/IEC 14977 Extended BNF* [13]. The relationships, axioms, and constraints which

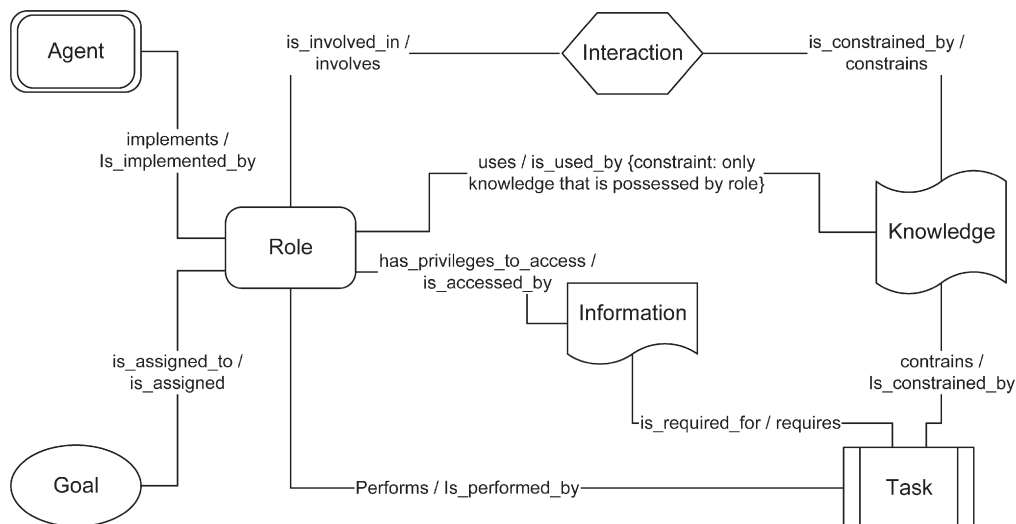


Fig. 2. The AIML fundamental constructs and their relationships.

Table 2
Symbols used in AIML specifications

Notation	Description
<i>General</i>	
Ω	MIBML Grammar where $\Omega = \{\Psi, T\}$
Ψ	Collection of AIML constructs $\Psi = \{\psi; i, \dots, n\}$
ψ	AIML Construct
T	Collection of constraints $T = \{\tau_i i = 1, \dots, n\}$
τ	Constraints
$C_{\text{attributes}}$	Common attributes
a_i	Activity (a task or an interaction)
<i>Goal</i>	
g	Goal
status	Goal status
c	Completed
e_x	Executing
w	Waiting
s	Suspended
<i>Role</i>	
r_{general}	Generic roles
r	Normal roles
$r_{\text{coordinator}}$	Special role to maintain goal status
d	Duties of a role
p	Privilege
<i>Interaction</i>	
i	Interactions
msg	Communication from initiator role to responder role
resp	Response to a message
speech	Communicative act
speech _{Act}	Action verb for communication
content	Example: Assertion, Query, etc. Actual message
<i>Task</i>	
t	Tasks
ϕ	Property / Logic / Subcomponent.
input	Represents input to a task
output	Represents output of a task
<i/o>	Represents both input and output
<i>Knowledge</i>	
k	Knowledge
k_d	Declarative Knowledge
k_p	Procedural knowledge
w_p	Workflow patterns
f	Facts
$rule_d$	Deduction rule
$x_{\text{structure}}$	Execution structure
x_{order}	Execution order
$x_{\text{constraint}}$	Execution constraint
ev	Event
ev_{external}	External event
ev_{temporal}	Temporal event
ev_{state}	State event
<i>Information</i>	
i'	Information
i'_f	Information flow

Table 2 (continued)

Notation	Description
<i>Information</i>	
i'_e	Information entity
e	Information entity instance
e	Information entity type
e_{info}	Internal entity information
$i'_{\text{flow-control-info}}$	Metadata for information flow control
c	control
$i'_{\text{flow-source}}$	Source from which an information flow emanates
$i'_{\text{flow-sink}}$	Entity receiving the information flows
i'_{data}	Actual data with flow
$i'_{\text{flow-external}}$	External source/sink for information flow
$i'_{\text{flow-frequency}}$	Information flow frequency
<i>Agent</i>	
a_g	Agent

govern the usage of the AIML constructs are defined in first order predicate logic. Table 1 presents the conventions adopted for the notations and symbols used in specifying the AIML grammar. These conventions apply to all the symbols used to explicate the various ontological categories that arise during the specification, namely: *instance*, *collections*, *types*, and *collection of types*. Table 2 provides an explanation of the symbols used in specifying the AIML grammar. This grammar consists of a collection of (a) AIML constructs (Ψ), and (b) constraints (T). Table 3 unambiguously defines the key AIML constructs in BNF and Table 4 provides a list of predicates that are used in specifying constraints. In the rest of this section, we provide a more detailed explanation of the fundamental AIML constructs along with their constraints and relationships based on theories in both the IBIS and multiagent literatures. During our discussion, we have attempted to avoid trivial axioms and constraints that can be easily reasoned.

3.2.1. Goal dependency model

Goal dependencies are captured by the goal construct in terms of a goal tree. The business goals identified during business analysis are organized into a goal tree to represent user requirements at different levels of detail. In the goal tree, a goal is decomposed iteratively until it reaches a collection of individual goals (g_i) at the leaf goal level. For ease of discussion, we refer to the collection of all individual goals (g_i) as a goal-set (G). The individual goals (g_i) are mutually exclusive and collectively exhaustive of the MIBIS system goals. As a result, the goal-set (G) forms a *whole-part* relationship in which an individual goal (g_i) is a component of the goal-set (G).

Table 3

The AIML conceptual modeling grammar in BNF

$\psi := G[R][T][K]A$;
 (* Upper case letters represent collections — see Table 1 for details *)
 $C_{\text{attributes}} := \text{"id", "name", "description"}$;

Goal:
 $g = C_{\text{attributes}}, \text{status}$;
 $\text{status} = \text{"c"} | \text{"e"} | \text{"w"} | \text{"s"}$;

Role:
 $r_{\text{General}} = r | r_{\text{coordinator}}$;
 (* A role is either a role that is regular role (one that is assigned to a leaf goal or a goal that updates the status of goals. If the duty is to update the status of a goal we call it the coordinator role*)
 $r_{\text{General}} = C_{\text{attributes}}, k, d$;
 $d = a | d, a$;
 (* duties of a role may consist of single activity or a sequence of activities *)
 $a = \{t | i\} | a, \{t | i\}$;
 (* In Extended BNF, $\bar{}$ implies symbols repeated one or more times *)

Interaction:
 $i = C_{\text{attributes}}, \text{Speech}$;
 $\text{Speech} = \text{Msg} | \text{Speech}, \text{Resp}$;
 (* The symbol Message is used to express both Msg and Resp as they have the same format. *)
 $\text{Message} = \text{initiator}, \text{responder}, \text{speech}_{\text{Act}}, \text{content}$;
 (* initiator is a role that initiates a communication and responder is also a role with responds to the message*)
 (* Assertion, Query, Request, Perform, Commitment, Denial, etc represent speech acts *)
 $\text{speech}_{\text{Act}} = \text{"Assertion"} | \text{"Query"} | \text{"Request"} | \text{"Perform"} | \text{"Commitment"} | \text{"Denial"}$;
 (* content represents the actual message communicated during the interaction *)

Task:
 $A = a | A, a$;
 $a = t | i$;
 $t = C_{\text{attributes}}, \text{input}, \text{output}, \text{method}$;
 (* The symbol $\langle i' o \rangle$ is used to express both $\langle \text{input} \rangle$ and $\langle \text{output} \rangle$ as they have the same format *)
 $\langle i' o \rangle = k_d | i' f | \langle i' o \rangle, k_d | \langle i' o \rangle i' f$;
 $\text{method} = \Phi$;
 $\Phi = \phi | \Phi, \phi$;
 (* ϕ is one of the methods or properties *)

Information:
 $i' = i' | i' f$;
 $i' e = C_{\text{attributes}}, \text{input}, \text{output}, \text{method}$;
 (* e_{info} includes semantic, syntactic and structural information about the entity *)
 (* e represents the data associated with the entity instance *)
 $i' f = C_{\text{attributes}}, i' \text{flow_control_info}, i' \text{flow_data_info}, i' \text{data}$;
 $i' \text{flow_control_info} = i' \text{flow_source} | i' \text{flow_sink} | i' \text{flow_frequency} | i' \text{flow_response_time}$;
 $i' \text{flow_source} = r | i' e | i' \text{flow_external} | i' \text{flow_source}, r | i' e | i' \text{flow_external}$;
 $i' \text{flow_sink} = r | i' e | i' \text{flow_external} | i' \text{flow_sink}, r | i' e | i' \text{flow_external}$;
 (* $i' \text{flow_external}$ is either an end user, or MIBIS entity or a non-MIBIS entity $i' \text{flow_external} = \text{"End_User"} | \text{"MIBIS_entity"} | \text{"Non_MIBIS_entity"}$ *)
 (* $i' \text{flow_data_info}$ is a data structure or format information *)

Table 3 (continued)

Information:
 (* i'_{data} is instance data *)
 $i'_{\text{chunk}} = \text{"Entity"} | \text{"Entity_instance"} | \text{"Entity_attribute"} | \text{"Entity_instance_attribute_value"}$;
 $P = \text{"Read"} | \text{"Write"} | \text{"View"} | \text{"Grant"} | \text{"Print"} | P, \text{"Read"} | \text{"Write"} | \text{"View"} | \text{"Grant"} | \text{"Print"}$;

Knowledge:
 $k = k_d | k_p | k_d, k_p$;
 $k_d = C_{\text{attributes}}, F | C_{\text{attributes}}, \text{RULE}_d | k_d, F | k_d, \text{RULE}_d$;
 $F = f | F, f$;
 (* f is simply a context relevant assertion *)
 $\text{RULE}_d = \text{rule}_d | \text{RULE}_d, \text{rule}_d$;
 (* rule_d is a deduction rule *)
 $k_p = C_{\text{attributes}}, w_p | k_p, w_p$;
 $w_p = x_{\text{structure}} | x_{\text{constraints}}$;
 (* $x_{\text{structure}}$ is either a sequence, parallel split, exclusive choice, etc *)
 $x_{\text{structure}} = \text{"sequence"} | \text{"parallel_split"} | \text{"exclusive_choice"} | \dots$;
 $x_{\text{constraint}} = \text{"ev_external"} | \text{"ev_temporal"} | \text{"ev_state"} | \text{"ev_resource"}$;
Agent:
 (* Agent is assigned a role using a predicate function defined in Table 4. *)

The highest level of the goal tree indicates the work system that is involved in business integration at the conceptual level. Goal decomposition can be performed using existing approaches for goal-oriented requirement engineering [e.g., [6,17]]. Fig. 3 describes a simplified goal tree for an integrative business system to provide made-to-order PC service to customers. The goal tree indicates that it requires coordination from 3 work systems: *customer service* as an interface for customer to request, change, or cancel an order, *PC assembly* to assemble a PC as requested, and *PC deliver* to deliver PC to customers. The goal tree has 4 individual goals (G1.1.1, G1.1.2, G1.2, and G1.3), each of which has to be completed in order for the overall goal (G1) to be completed.

The implication that all the individual goals (g_i) have to be accomplished in order to satisfy the overall goals of the system leads to Axiom 1.

Axiom 1. System goals are accomplished if and only if all individual goals are accomplished.

$$(\forall g) \text{has_state}(g, c) \rightarrow \text{has_state}(G, c) \quad (1)$$

In order to ensure the enforcement of Axiom 1 in the process of system development, we include an attribute called status in the goal schema (as shown in Table 3) in addition to the attributes that are common to all AIML constructs. The attribute is used to maintain the state of a goal. An individual goal can be in one of the following states: completed (*c*), executing (*e*), waiting (*w*), or suspended (*s*). Although the attribute status is a design

Table 4

List of predicates used to specify the AIML grammar

Predicate	Meaning
$\text{accesses}(r, i'_{\text{chunk}})$	Role r accesses the information i'_{chunk}
$\text{accomplish}(g, r)$	Goal g is accomplished by role r
$\text{agent}(a_g)$	a_g is an agent
$\text{assign}(p, r, i'_{\text{chunk}})$	Privilege p is assigned to role r for i'_{chunk}
$\text{assigned}(r, g)$	Role r is assigned to goal g
changed_goal_stat $(r_{\text{coordinator}}, g, \text{status})$	Role $r_{\text{coordinator}}$ changes the state of goal g to status
$\text{concurrent}(a_{t1}, a_{t2})$	Activity a_{t1} is in parallel with activity a_{t2}
$\text{execute}(r, a_i)$	Role r performs activity a_i
$\text{frequency}(a_i, n)$	Activity a_i must be performed with frequency n
$\text{goal}(g)$	g is a goal
$\text{has_goal}(a_g, g)$	Agent a_g has a goal g
$\text{initiator}(i, r)$	Role r is the initiator of an interaction
has_permission $(r, p, i'_{\text{chunk}})$	Role r has permission p to access i'_{chunk}
$\text{has_properties}(t_i, \Phi)$	Task t_i has a collection of Properties / Methods Φ
$\text{responder}(i, r)$	Role r is the responder of the interaction
$\text{has_state}(g, \text{status})$	Goal g has state status
$\text{information}(i')$	i' is either an information entity or flow
$\text{interaction}(i)$	i is an interaction
$\text{isa_typeof}(t_i, t_j)$	Task t_i is of type t_j and further task t_i inherits from task t_j collection of Properties/ Methods Φ
$\text{knowledge}(k)$	k is a knowledge nugget
$\text{fact}(f)$	f is a fact
$\text{deduction}(f)$	Fact f is obtained by deduction
$\text{operate_on}(\text{rule}_d, F)$	Deduction rule has to operate on facts
$\text{optional}(a_i)$	Activity a_i is optional
$\text{overrides}(t_i, \phi_1, \phi_2)$	Task t_i overrides property ϕ_1 with property ϕ_2
$\text{partOf}(t_i, t_j)$	Task t_i is a sub-task of part of task t_j
$\text{plays}(a_g, r)$	Agent a_g plays role r
$\text{precedes}(a_{t1}, a_{t2})$	Activity a_{t1} is executed before activity a_{t2}
$\text{revoke}(p, r, i'_{\text{chunk}})$	Privilege p is assigned to role r for i'_{chunk}
$\text{role}(r)$	r is a role
$\text{subordinate}(r_i, r_j)$	Role r_i is subordinate to role r_j
$\text{task}(t)$	t is a task
$\text{trigger}(ev, a_i)$	Activity a_i is triggered by event ev

consideration, we include it in the goal schema to facilitate the transition from conceptual models to design models in system development.

3.2.2. Role determination

The AIML roles are design artifacts designed to accomplish individual goals at the leaf goal level of the goal tree. The relationship between the goal and role is *bijective* one-to-one mapping. It implies that an individual goal can be assigned to only one role, and a role is responsible for only one individual goal. Moreover, it indicates that the process of building goal tree and that of

designing roles are interactive — the goal tree may need to be modified while roles are designed and vice versa. The bijective relationship between individual goals (g_i) and individual roles (r_i) are expressed in Axiom 2.

Axiom 2. Each individual goal is assigned to a unique role, and each role is responsible for a unique individual goal.

$$\begin{aligned} &\text{goal}(g_k) \wedge \text{role}(r_i) \wedge \text{assigned}(r_i, g_k) \\ &\rightarrow \neg \text{assigned}(r_j, g_k) \wedge \text{role}(r_j) \wedge (r_i \neq r_j) \end{aligned} \quad (2)$$

$$\begin{aligned} &\text{goal}(g_i) \wedge \text{role}(r_k) \wedge \text{assigned}(r_k, g_i) \\ &\rightarrow \neg \text{assigned}(r_k, g_j) \wedge \text{goal}(g_j) \wedge (g_i \neq g_j) \end{aligned} \quad (3)$$

In our example, four distinct roles need to be designed to take responsibility for the four individual goals shown in Fig. 3 and these are described and mapped in Table 5.

Roles within AIML do not have any hierarchical relationships. Existence of a role hierarchy would imply master–slave relationships. In the current conceptualization, a MIBIS system is viewed as a collection of distributed autonomous agents without any central control. Incorporation of role hierarchies in AIML is beyond the current scope and is a significant area for future research. Therefore, all agents within a system are conceptualized to be in peer-to-peer relationships. Consequently, supervisory roles do not exist as reflected in Axiom 3.

Axiom 3. No role can control other roles in a MIBIS system.

$$\text{role}(r_i) \wedge \text{role}(r_j) \wedge (r_i \neq r_j) \rightarrow \neg \text{subordinate}(r_i, r_j) \quad (4)$$

Furthermore, in order to ensure the accomplishment of the overall systems goals in a peer-to-peer environment, AIML provides one special role termed *coordinator* to track goal accomplishment status of other roles. The task of the *coordinator* role ($r_{\text{coordinator}}$) is to update the status of all the goals in the system and ensure that every goal in the overall system goal tree is accomplished. Therefore, we have the following axiom:

Axiom 4. The state of the goal is changed by a special role called the coordinator role which possesses the necessary permission to change the status of goals in the MIBIS system.

$$\begin{aligned} &\text{goal}(g) \wedge \text{role}(r_{\text{coordinator}}) \wedge \text{changed_goal_stat}(r_{\text{coordinator}}, g, \text{status}) \\ &\rightarrow \text{has_state}(g, \text{status}) \end{aligned} \quad (5)$$

where the status is set to either completed, executing waiting or suspended and $\exists! r_{\text{coordinator}} \text{role}(r_{\text{coordinator}}) \in R$

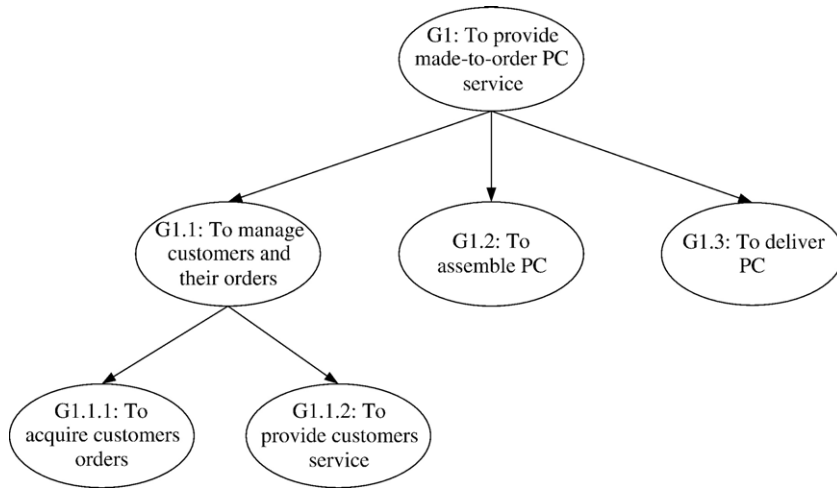


Fig. 3. A goal tree for the online PC made-to-order system.

where R is the collection of roles. Please note that we have used the notation $\exists!$ to specify uniqueness quantification.

3.2.3. Interaction specification

Following the speech act theory, an interaction in AIML is defined as a coordinated sequence of speech acts [8,39]. Speech Acts (SA) are utterances that contain information needed to assert and perform actions and serve as building blocks of communication protocols. They define what people do while communicating [9,30]. Speech act verbs are used in speech act utterances, to perform actions such as booking, complaining, forgiving, etc. [34,35]. Therefore, the specification of an interaction in AIML includes *initiator*, *responder*, *speech act*, and *message* as defined in Table 3. The initiator starts an interaction by sending a message using a speech act. However, the message may or may not evoke a response from the responder. Likewise, the response from the responder may or may not evoke a new message from the initiator. Table 6 shows an example specification of the interaction for Sales to find a factory to assemble PCs for customers.

In MIBIS, not a single role has complete knowledge and capabilities to accomplish the overall system goal. That is, all roles need to interact with other roles. On the other hand, every interaction must involve at least two distinct roles.

Table 5
The bijective mapping between goals and roles

Individual Goal	Role
G1.1.1 to acquire customers' orders	R1.1 Sales
G1.1.2 to provide customer service	R1.2 Customer service
G1.2 to assemble PC	R2 Factory
G1.3 to deliver PC	R3 Shipping

Therefore, the relationship between the role and the interaction construct is governed by Axioms 5 and 6.

Axiom 5. Every role is involved in at least one interaction either as an initiator or a responder.

$$\forall r \text{ role}(r) \rightarrow \exists! i (\text{interaction}(i) \wedge (\text{initiator}(i, r) \vee \text{responder}(i, r))) \quad (6)$$

Axiom 6. Every interaction involves an initiator role and a responder role.

$$(\forall i [(\exists! r_i) \text{role}(r_i) \wedge \text{interaction}(i) \wedge \text{initiator}(i, r_i)]) \rightarrow (\exists! r_j) \text{role}(r_j) \wedge \text{responder}(i, r_j) \wedge (r_i \neq r_j) \quad (7)$$

where R is the collection of roles in the MIBIS universe and $r_i, r_j \in R$.

3.2.4. Task specification

A task can be as simple as a single activity or as complex as a business process or a workflow. AIML supports

Table 6
Specification of the interaction “Find Factory”

Interaction	
Identifier	Inter1
Name	Find Factory
Description	Sales contacts possible factories and find an appropriate one for customer's order
Initiator role	R1.1 Sales
Responder role	R2 Factory
speech act messages	Sales (Request) propose a contract for customer's order Factory (Acceptance) accept the contract Sales (Assertion) confirm the contract

tasks to be decomposed recursively into two types of hierarchy: sub-activities and subtypes. The former is a decomposition of a task into AND/OR sub-activity components, while the latter represents an IS-A hierarchy. For example, as shown in Fig. 4, a parent task “process order” can be decomposed into the constituent sub-activities of “receive order”, “find factory”, “find shipping”, and “confirm order”; the activity “find shipping” can be further decomposed into specialized subtypes of “find ground shipping” and “find express shipping”. Similar conceptualization is also employed in Malone et al. [25].

To support the task hierarchy, AIML provides two predicates: (a) $\text{partOf}(t_i, t_j)$ – task t_i is considered a subpart of task t_j ; and (b) $\text{isa_typeof}(t_i, t_j)$ – task t_i is considered to be of type task t_j . Axiom 7 describes the non-reflexivity properties (C 8 and C 9) of task decomposition. These properties are important because they ensure task hierarchical structure to be represented as a Directed Acyclic Graph (DAG).

Axiom 7. A task and its sub-level tasks are non-reflexive.

$$\text{partOf}(t_i, t_j) \rightarrow \neg \text{partOf}(t_j, t_i) \quad (8)$$

$$\text{isa_typeof}(t_i, t_j) \rightarrow \neg \text{isa_typeof}(t_j, t_i) \quad (9)$$

Partitioning along the dimension of subtypes (isa_typeof relationships) allows for inheritance from a more generalized type to a more specialized type, in a manner that is similar to the type hierarchy considered in most object-oriented approaches. It is important to note that while the Part-Whole ($\text{partOf}(t_i, t_j)$) is transitive (see Axiom 8 — C 10), the inheritance relationship ($\text{isa_typeof}(t_i, t_j)$) is not constrained to be so. The transitive closure in the inheritance relationship ($\text{isa_typeof}(t_i, t_j)$) is not enforced so that a task does not automatically inherit from ances-

tors properties and methods that the parent tasks have overridden.

Axiom 8. Transitive closure of part-whole relationship

$$\text{partOf}(t_i, t_j) \wedge \text{partOf}(t_j, t_k) \rightarrow \text{partOf}(t_i, t_k) \quad (10)$$

Similar to the object-oriented approach, AIML allows for subtypes to override inherited properties as described by Axiom 9.

Axiom 9. A subtype is able to override the properties inherited from its parent.

$$\text{isa_typeof}(t_i, t_j) \wedge \text{has_properties}(t_j, \Phi_1) \wedge \text{overrides}(t_i, \Phi_1, \Phi_2) \rightarrow \text{has_properties}(t_i, \Phi_2) \quad (11)$$

The task construct has been conceptualized to include the following attributes: *Input*, *Output*, *Parent Tasks*, *Sub-Activities*, *Sub-Types*, and *Method* as defined in Table 3. Inputs are information required for task execution, while Outputs are information generated from the task; the Method metaphor embodies the procedural knowledge that specifies the detailed logic for execution of the task. A method can be described using different mechanisms, including structured English and pseudo code. Table 7 is an example specification for the task “receive order”.

3.2.5. Information modeling

Information refers to data resources available within a MIBIS application and may pertain to both the functional (business-aware) and the non-functional (business-unaware; IT system-specific) aspects of the MIBIS application. The information construct of AIML consists of information entities and information flows. The specifications of both information entities and information flows are detailed in Table 3.

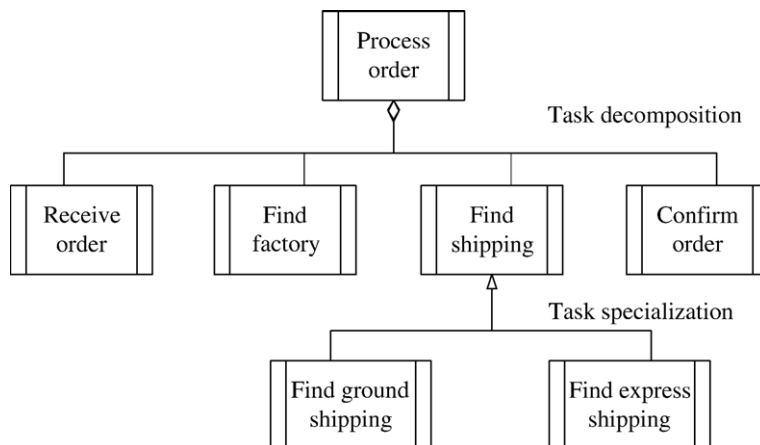


Fig. 4. an example of task decomposition.

Table 7
Specification of task “Receive order”

Task	
Identifier	T1.1
Name	Receive order
Description	Capture customer’s order
Input	Customer information Order request
Output	Order
Parent Tasks	Process order
Sub-Activities	None
Sub-Types	None
Method	Do for each order If new customer, register new account, Endif Get customer account information Create order Enddo

Information entities refer to internal data within the system that are part of data stores. They represent regular business objects (such as order, customer, etc.) and other materialized views of data. The schema of information entities includes the structure of tables, the relationships, entity integrity constraints, referential integrity constraints, cardinality constraints, etc. Information entities may be implemented using relational databases and the schema corresponds to items typically stored in database repositories. The specifications of information entities have no difference from those in traditional data modeling. Table 8 indicates that customer number, name, address, phone, and internal credit ranking must be recorded in the database.

Information flows represent data that are in transit; for example, data moving between external users and roles, between information entities and roles, or between other external systems and roles. Information flows are different from information entities in their time orientation [5]. Information flows are temporal. They cease to exist once they are acted upon by an agent or stored in a data store or provided to an entity external to the MIBIS system. Information flows represent data resources for agents to update their beliefs or for information entities to update their states. Table 9 describes the information flow from customers to make order requests.

Table 8
Specification of information entity “Customer”

Information entity	
Identifier	INE1
Name	Customer
Description	Customer information and his/her credit status
Attributes	Customer number, name, address, phone, internal credit ranking

Table 9
Specification of information flow “Customer order request”

Information Flow	
Identifier	INF1
Name	Customer order request
Description	Order request from customers to sales
Flow source	External entity Customer
Flow sink	R1.1 Sales
Flow frequency	When customers place orders
Flow response time	Synchronous
Flow data	Item number, quantities

All data resources and information available within the MIBIS application are protected and access is based on the privileges a role possesses. Privileges reflect authority of roles to view, manipulate, create, or take other alternative actions on information. Privilege to access information entities and flows are granted at various levels of granularity (i_{chunk}) as determined by business rules. A role has to be granted the privilege at the intended level of granularity if it has to be able to access information. Therefore, we have the following axiom.

Axiom 10. A role has to be granted a privilege in order to enable it to access necessary information.

$$(\forall r)(\forall i'_{\text{chunk}})[\text{role}(r) \wedge \text{information}(i'_{\text{chunk}}) \wedge \text{accesses}(r, i'_{\text{chunk}})] \rightarrow \text{has_permission}(r, p, i'_{\text{chunk}}) \quad (12)$$

$$\text{has_permission}(r, p, i'_{\text{chunk}}) \rightarrow \text{assign}(p, r, i'_{\text{chunk}}) \quad (13)$$

3.2.6. Knowledge specification

Human knowledge resides in the mind of individuals [1] and not in the collection of information. Similarly, knowledge in AIML is conceptualized to exist within individual roles and is therefore private to the roles. Accordingly, a role must possess the needed knowledge in order to use it. This constraint is represented in Axiom 11.

Axiom 11. A role can only use its own knowledge.

$$(\forall r)(\forall k)[\text{role}(r) \wedge \text{knowledge}(k) \wedge \text{uses_knowledge}(r, k)] \rightarrow \text{has_knowledge}(r, k) \quad (14)$$

Individual and organizational knowledge are very broad constructs consisting of both explicit and tacit knowledge [1]. However, the knowledge construct in AIML is viewed from a somewhat narrower perspective, in that it captures and represents only computational knowledge that is explicitly defined.¹⁰ Individual and

¹⁰ If tacit knowledge can be converted into explicit knowledge, it can be represented using the knowledge component of the AIML grammar.

organizational knowledge includes declarative (know-that) or procedural (know-how). Correspondingly, explicit computational knowledge in AIML consists of both declarative knowledge (d_k) and procedural knowledge (p_k). Declarative knowledge is composed of facts and deduction rules. Facts are beliefs that a role keeps about itself, about other roles in the system, and about the environment it resides in. Deduction rules empower roles to engage in deductive reasoning with the constraint that at least two existing facts are needed to deduce a new fact. This is further elaborated in Axiom 12.

Axiom 12. A new fact can only be deduced from at least two existing facts using a deduction rule.

$$\text{fact}(f_k) \vee \text{deduction}(f_k) \rightarrow (\exists \text{rule}_d)(\text{operates_on}(\text{rule}_d, F) \quad (15)$$

where F is a collection of facts with the collection containing at least two elements i.e. $\exists^{\geq 2} f \in F$, and the quantification expression $\exists^{\geq 2}$ is used to indicate ‘there exists at least two’.

Procedural knowledge dictates issues relating to precedence, timing, frequency, etc. of activities, and is conceptualized to consist of activity execution structure and activity execution constraints. Activity execution structure relates to knowing the order in which activities need to occur. As discussed earlier, an activity can either be a task or an interaction. Activity execution structure also includes information regarding how frequently activities have to be performed and whether certain activities are optional. The predicates ($\text{precedes}(a_{t1}, a_{t2})$, concurrent (a_{t1}, a_{t2}), frequency(a, n) and optional(a)) are used to express activity execution structure as shown in Table 4. Activity execution constraints determine the events that trigger activities. In AIML, an activity is always triggered by an event. The event can be an external event (ev_{external}), a temporal event (ev_{temporal}), or a state event (ev_{state}). External events emanate either from the environment including other roles or from end-users (such as a customer placing an order). External events generated by another role in the system correspond to inter-role-task dependencies. For example, a role r_1 may start executing a task t_1 only when another role r_2 completes a task t_2 . In order to preserve the autonomy of roles and hence the agents playing the roles, AIML does not permit modeling such constraints as part of the task execution structure of the role r_1 . However, such constraints can be modeled as an external event for role r_1 . Temporal events are constraints placed on the execution of tasks or interactions based on some time consideration such as the elapse of some time period. A state event is an event that occurs inside a role, and changes the state of the role, and ac-

cordingly triggers a task or a set of tasks. The requirements of activity execution constraints are expressed in Axiom 13.

Axiom 13. Activities in the MIBIS universe must be triggered by an event. When an event (external, temporal, state, etc) trigger an activity, a role executes the activity.

$$(\forall a_t)[\text{role}(r) \vee \text{executor}(r, a_t) \rightarrow (\exists ev)\text{trigger}(ev, a_t)] \quad (16)$$

Table 10 describes an example specification of knowledge for sales.

3.2.7. Agent modeling

In an organizational context, a role is assigned to one or more physical human actors to accomplish organizational goals. In a similar manner, an abstract role in the MIBIS universe is instantiated by autonomous component agent. The component agent is modeled as a system entity that is capable of sensing its environment and acting autonomously to meet its design objectives [41]. Axiom 14 and Axiom 15 describe the conceptualization of the relationship between a role and an agent.

Table 10
Specification of knowledge for Sales

<i>Knowledge</i>	
Identifier	K1
Name	Sales Knowledge
Description	Knowledge on pricing, credit ranking, financing, order rejection/approval policy, and constraints on performing tasks
<i>Declarative Knowledge</i>	
Facts	Reliable factories Reliable shipping
Deduction rules	If no factory can be found to satisfy customer's exact request, a comparable configuration is suggested.
<i>Procedural knowledge</i>	
Execution structure	T1.1 “receive order” precedes T1.2 “find factory” and T1.3 “find shipping”. T1.2 “find factory” and T1.3 “find shipping” are parallel. T1.2 “find factory” and T1.3 “find shipping” precedes T1.4 “confirm order”.
Execution constraints	T1 “process order” is triggered by receiving customer's order request.

Axiom 14. Every role must be played by at least one agent.

$$\forall r \text{ role}(r) \rightarrow \exists \text{agent}(a_g) \wedge \text{plays}(a_g, r) \quad (17)$$

Axiom 15. Every agent must play at least one role.

$$\forall a_g \text{ agent}(a_g) \rightarrow \exists \text{role}(r) \wedge \text{plays}(a_g, r) \quad (18)$$

It may be noted that for every role there is at least one agent that plays that role. However the agents are not constrained in any other way. This allows for the assignment of multiple agents to a role. Similarly we constrain using expression C20 that every agent must play a role. There are no further constraints and this essentially allows for the assignment of many roles to an agent.

4. AIML quality evaluation

AIML conceptual grammar is developed following the helix-spindle model for ontological engineering [18] to ensure a high quality of this grammar. Following this model, the development process goes through three major phases—a conception phase, an elaboration phase, and a definition phase. At each phase, the AIML constructs and their relationships are defined based on well-founded theories and tested by building an application domain framework. If any problems are detected during framework building, the AIML development slides back to the beginning of the phase to guarantee coherence and extendibility of the grammar.

Conceptual modeling is essentially making statements in some language and it is closely linked to linguistic concepts. In this section, we follow Lindland et al.'s framework [23] for discussing the quality of the AIML grammar in terms of *syntactic*, *semantic*, and *pragmatic* quality.

Syntactic quality deals with how well the conceptual models correspond to the modeling grammar. Its goal is to ensure syntactic correctness. There are three basic mechanisms for ensuring syntactic quality: error prevention, error detection, and error correction. Error prevention is the mechanism by which insertion of erroneous statements into the model is rejected. Error detection is finding errors after erroneous statements have been inserted into a model. Error correction deals with replacing a detected error with a correct statement. Obviously, AIML provides enough support for syntactic quality by defining AIML formally in Backus-Naur form (BNF) and first-order logic. Conceptual models created in AIML can be easily analyzed to detect errors and inconsistencies. While it may be difficult to automate error correction, software

tools may be developed to automate error prevention and error detection based on formally-defined AIML syntax.

Semantic quality deals with how well the conceptual models correspond to the problem domain. The more closely a conceptual model reflects the problem domain, the better the semantic quality of that conceptual model. There are two semantic quality goals: validity and completeness. Validity ensures that all statements made by a model are correct and relevant to the problem. Completeness means that the model contains all the statements about the domain. While it is impossible to achieve total validity and completeness for anything but extremely simple problems, the AIML grammar is designed to capture the semantics of the MIBIS universe in a consistent, comprehensive, and unambiguous manner with a minimum number of modeling constructs. In [19], Kishore et al investigate the characteristics of the MIBIS universe based on the literatures in both the IBIS and multiagent systems domains. After analyzing various concepts involved in modeling MIBIS, they propose 8 minimal ontological foundation constructs for the MIBIS universe, including goal, role, interaction, task, information, knowledge, resource, and agent. The semantics of these MIBIS ontological constructs were also evaluated by Zhang et al. using the Bunge-Wand-Weber framework [44] who found the grammar to be quite expressive and comprehensive. This paper extends [19] by formally defining the internal structure of these ontological constructs and their relationships. Axioms are included to ensure the semantic consistency of conceptual models. Table 11 indicates that the AIML grammar overcomes UML limitations in supporting agent-based information systems and fully supports software agent characteristics. Therefore, the AIML grammar is a good fit with the MIBIS universe. On the other hand, the AIML grammar also provides support for completeness by supporting modularity. The AIML constructs are self-contained and allow new specifications to be added without modifying existing part of conceptual models. Each AIML building block can be changed and replaced without much effect on others. For example, one can assign new roles to agents and remove ones with no effect on the internal model of the roles.

Pragmatic quality deals with how well a conceptual model corresponds to its audience interpretation. The goal of pragmatic quality is to improve users' understanding of a conceptual modeling grammar and reduce the misuse of the grammar in constructing conceptual models. Pragmatic quality of the AIML grammar was evaluated earlier through an ontological analysis using the Bunge-Wang-Weber (BWW) model [36] and we discuss it briefly below. It is also evaluated in the current

Table 11
AIML support for agent-based information systems

Agent-based IS Characteristics [41]	AIML Support	UML Limitations
Autonomous	A role encapsulates its functionality (i.e., it is responsible for its interactions and tasks). This functionality is internal and is not affected by the environment; further a role also encapsulates internal knowledge that allows it to perform its tasks and interactions; these features of AIML represent the autonomy of a role.	Although an object encapsulates its functionality, it has no control on what actions to take. Its internal methods can only be initiated by external invocation. Further, an object cannot refuse an external request. In other words, UML lacks constructs to support the decision-making aspect of an agent's autonomy.
Reactive	The knowledge construct captures the events that trigger interactions and tasks; thus, AIML supports reactivity for roles and agents that play those roles.	UML fully supports the reactive aspect of an agent by allowing object's internal methods to be triggered by external events.
Proactive	The knowledge construct models possible activity execution paths and these allow the roles and agents in AIML to be proactive in decision making to accomplish goals specified in the goal construct.	UML has no explicit "mentalistic" constructs to support the agent's proactive aspect. It does not support agent's goal-oriented behaviors.
Social	The interaction construct models the protocols, the communication paths, and the speech acts of the messages, thus, giving roles and agents the social ability to interact.	While UML is able to model low-level message exchanging, it lacks capabilities to model agent conversations, which includes not only a sequence of messages but also semantics associated with communicative (speech) acts.

paper through a complexity analysis using metrics proposed by Rossi and Brinkkemper [27] and this is also discussed below.

From an ontological analysis perspective, an information system is a representation of the perceived real-world system. Therefore, a good conceptual modeling grammar must manifest the meaning of the real-world to be represented. By mapping the grammatical constructs of a conceptual modeling grammar to the ontological constructs of an ontological model (such as BWW model) which represents the real-world situation, we are able to identify ontological deficiencies of the grammar. Such ontological deficiencies result in user confusion in interpreting and using the grammar. According to Wand & Weber [37], four ontological deficiencies may be found in a grammar: 1) *ontological incompleteness* occurs when ontological constructs do not have equivalent constructs in the modeling grammar; 2) *construct redundancy* occurs when several constructs of the conceptual modeling grammar map onto a single ontological construct; 3) *construct overload* occurs when several ontological constructs are mapped onto a single construct in the modeling grammar; and 4) *construct excess* occurs when a grammatical construct might not map to any ontological construct. Because the BWW ontology is one of the most used higher-level information systems ontologies for evaluating conceptual modeling grammars, we conducted an ontological analysis of the AIML grammar using the BWW ontology to remove ontological deficiencies before formally defining the grammar in this paper. This onto-

logical analysis, details of which are available in [44], indicated that the ontological semantics of the AIML grammar are quite clear and that the grammar has the potential to enhance user communication. We also found that the AIML grammar may also benefit MIBIS system developers as they will be able to identify easily the correct AIML constructs to represent problem domain knowledge and to develop precise conceptual models for a MIBIS system.

The second evaluation of pragmatic quality of the AIML grammar was conducted through a complexity analysis of the AIML grammar using metrics proposed by Rossi and Brinkkemper [27]. Complexity is a key measure of the effectiveness of a language because complexity directly affects the learning ability and the ease-of-use of the language [31]. Rossi and Brinkkemper [27] proposed a set of seventeen complexity metrics to evaluate systems development methods and individual techniques within those methods. Considering that AIML is a conceptual modeling grammar that covers only the early stages of the systems development life cycle, we include only the independent measures and aggregate metrics for individual systems development techniques in our analysis. Following are the definitions of the metrics we used in our analysis.

Let the model of a modeling technique T be given as M_T , its object types as O_T (object is a thing that exists on its own), its property types as P_T (properties are characteristics of other meta-types), its relationship types as R_T (relationship is an association between two

Table 12

Complexity metrics values for AIML and UML conceptual-level diagrams

		AIML	UML Diagrams (see footnote 11)					
			Class	Activity	Sequence	Collaboration	StateChart	Aggregate
Metric 1	$n(O_T)$	7	7	8	6	4	10	35
Metric 2	$n(R_T)$	12	18	5	1	1	4	29
Metric 3	$n(P_T)$	19	18	6	5	7	11	47
Metric 5	$P_o(M_T)$	2.71	1.71	0.75	0.67	1	1	5.13
Metric 7	$P_R(M_T)$	3.33	1.22	0.20	6	8	0.5	15.92
Metric 9	$R_o(M_T)$	2.86	2.57	0.63	0.17	0.25	0.40	4.02
Metric 11	$C(M_T)$	0.52	0.10	0.13	0.13	0.14	0.09	0.59
Metric 12	$C'(M_T)$	23.54	26.40	11.18	7.87	8.12	15.39	65.38

or more objects), its role types as X_T (role is the name given to the link between an object and its connection with a relationship), and the function $n(A)$ denotes the number of elements in the set of A .

Metric 1 $n(O_T)$ is the count of object types per technique. This metric demonstrates the number of individual object types used to specify object systems. The metric values for various object-oriented techniques range from 1 to 10.¹¹

Metric 2 $n(R_T)$ is the count of relationship types per technique. It indicates the number of concepts that are used for describing connections between objects. The metric values for various object-oriented techniques range from 1 to 18 (see footnote 11).

Metric 3 $n(P_T)$ is the count of property types per technique. The metric values for various object-oriented techniques range from 3 to 18 (see footnote 11).

Metric 4 $P_o(M_T, o) = n(P_T(o))$, where $o \in O_T$. This metric counts the number of properties for a given object type.

Metric 5 $\bar{P}_o(M_T) = \frac{1}{n(O_T)} \sum_{o \in O_T} P_o(M_T, o)$ This metric is average number of properties per object type. The metric values for various object-oriented techniques range from 0.67 to 5 (see footnote 11).

Metric 6 $P_R(M_T, e) = n(p_T(e)) + \sum_{x \in R_T(e)} n(p(\text{role}(x)))$, where $e \in R_T$. This metric is the number of properties of a relationship type and its accompanying role types.

Metric 7 $\bar{P}_R(M_T) = \frac{1}{n(R_T)} \sum_{e \in R_T} P_R(M_T, e)$ This metric counts the average number of properties per relationship type. It shows the complexity of

the interface between object types. The metric values for various object-oriented techniques range from 0.20 to 8 (see footnote 11).

Metric 8 $R_o(M_T, o) = n(\{e \in R_T | o \in \cup_{x \in R_T(e)} \text{object}(x)\})$, where $o \in O_T$. This metric gives the number of relationship types that can be connected to a certain object type.

Metric 9 $\bar{R}_o(M_T) = \frac{1}{n(O_T)} \sum_{o \in O_T} R_o(M_T, o)$ This metric gives the average number of relationship types that can be connected to a given object type. The metric values for various object-oriented techniques range from 0.17 to 5 (see footnote 11).

Metric 10 $C(M_T, o) = \frac{P_o(M_T, o)}{\sum_{e \in A} P_R(M_T, e)}$, where $A = \{x \in$

$R_T | o \in \cup_{x \in R_T(x)} \text{object}(y)\}$ The quotient indicates the division of work in this technique, i.e. are things described by their internal properties, or by external connections.

Metric 11 $\bar{C}(M_T) = \frac{1}{n(O_T)} \sum_{o \in O_T} C(M_T, o)$ This metric shows the average complexity for the whole technique. The metric values for various object-oriented techniques range from 0.09 to 3.

Metric 12 $C'(M_T) = \sqrt{n(O_T)^2 + n(R_T)^2 + n(P_T)^2}$ This metric gives the total conceptual complexity of a technique. The metric values for various object-oriented techniques range from 3.32 to 26.40.

In our complexity analysis, we compare AIML complexity metrics with UML diagrams that are used in the early stages of system development to specify user requirements and capture similar types of constructs and relationships as AIML. We select UML as the reference grammar for our comparison because UML has become the *de facto* modeling grammar accepted widely for object-oriented systems analysis and design. Table 12

¹¹ The metric values are from Table 3 — Complexity values for UML and other OO methods in [31] on page 31.

compares the complexity metrics values¹² of the AIML grammar with UML diagrams at the conceptual level. While UML class diagram captures data requirements, UML activity diagram, sequence diagram, collaboration diagram, and StateChart diagram capture and describe system requirements from a behavioral perspective. The real-world information captured in these UML models is similar to the real-world information that is captured through AIML constructs. Table 12, therefore, aggregates the metric values for the selected UML conceptual-level diagrams and compares it as a group with AIML metric values to provide an apples-to-apple comparison.

Table 12 shows that AIML has significantly lower values for all complexity metrics as compared to the aggregate UML metric values except for $\bar{C}(M_T)$ which is also lower for AIML indicating that AIML is much more intuitive, and easier to learn and use. This is very much in line with contemporary anecdotal evidence that suggests that UML is a fairly complex modeling formalism. The complexity of UML stems in part from the fact that it is a general purpose modeling formalism that caters to all kinds of object-oriented systems. AIML, on the other hand, is a special-purpose modeling formalism that is developed specifically for modeling the agile integration of work systems using the notions and principles of multiagent systems and has a much narrower focus. These complexity metrics provides further evidence about the high degree of pragmatic quality of AIML and its superiority over UML, the de facto standard for conceptual modeling, in terms of ease of learning and ease of use.

5. Conclusion

Large-scale use of multiagent technology in various integrative business information systems requires a special-purpose conceptual-modeling grammar to facilitate the analysis and design of MIBIS systems. In response to such needs, we have developed the AIML grammar for conceptual modeling and high-level design of agile IBIS systems in the MIBIS universe. The grammar provides formal definitions for constructs that form an ontological foundation for representing the MIBIS universe, and provides a starting point for ontology-driven MIBIS development. The grammar benefits both IBIS researchers and practitioners. On the one hand, it

advances researchers' understanding of the MIBIS universe and thus forms a foundation for developing various methodologies for development of agile IBIS systems in the MIBIS universe. On the other hand, the formal definitions of the AIML constructs provide templates for IBIS practitioners to avoid developing individual applications from scratch each time, thereby facilitating system development knowledge reuse.

There are several future research directions to further this study. First, in order to enhance the reuse of MIBIS domain-specific knowledge, lower-level ontological categories for the AIML foundation constructs need to be developed (e.g., an ontology of MIBIS roles, an ontology of MIBIS goals, etc.). Second, further elaboration of the AIML grammar and axiomatic proofs are needed to increase the rigor of this grammar. Third, interactions in current study are limited to direct interactions between a pair of roles. In future work, interactions should be extended to include those between more than two roles. Fourth, knowledge in this study is limited to deductive reasoning and it is possible to extend AIML to include other types of knowledge and reasoners, such as abductive, inductive, and case-based reasoning, etc. Last, but not the least, appropriate methodologies and software tools will also need to be developed to support the analysis, design, and development of agile IBIS systems in the MIBIS universe.

References

- [1] M. Alavi, D.E. Leidner, Knowledge management and knowledge management systems: conceptual foundations and research issues, *MIS Quarterly* 25 (1) (2001) 107–136.
- [2] S. Alter, A general, yet useful theory of information systems, *Communications of the Association for Information Systems* 1 (13) (1999) 1–70.
- [3] A. Basu, A. Kumar, Research commentary: workflow management issues in e-business, *Information Systems Research* 13 (1) (2002) 1–14.
- [4] B.J. Biddle, E.J. Thomas, *Role Theory: Concepts and Research*, John Wiley & Son, Inc., 1966.
- [5] S. Conger, *The New Software Engineering*, Wadsworth Series in Management Information Systems, Wadsworth Publishing Company, Belmont, CA, 1994.
- [6] A. Dardenne, A. van Lamsweerde, S. Fickas, Goal-directed requirements acquisition, *Science of Computer Programming* 20 (1993) 3–50.
- [7] S.A. Deloach, M.F. Wood, C.H. Sparkman, Multiagent systems engineering, *International Journal on Software Engineering and Knowledge Engineering* 11 (3) (2001) 231–258.
- [8] J.L.G. Dietz, DEMO: towards a discipline of organisation engineering, *European Journal of Operational Research* 128 (2) (2001) 351–363.
- [9] J. Habermas, *The Theory of Communicative Action*, vol. I, Beacon Press, Boston, 1984.
- [10] S.S. Heragu, R.J. Graves, B.-I. Kim, A. St Onge, Intelligent agent based framework for manufacturing systems control, *IEEE*

¹² Metrics 4, 6, and 8 are not included in the table because they are metrics for a given object type or a given relationship type and not for a technique. Their definitions are provided for explanation of metrics 5, 7, and 9.

- Transactions on Systems, Man and Cybernetics, Part A 32 (5) (2002) 560–573.
- [11] G. Huber, Organizational learning: the contributing processes and the literatures, *Organization Science* 2 (1) (1991) 88–115.
 - [12] M.N. Huhns, L.M. Stephens, Automating supply chains, *IEEE Internet Computing* 5 (4) (2001) 90–93.
 - [13] ISO/IEC, Information technology-Syntactic metalanguage-Extended BNF, Published ISO standard 14977:1996(E), ISO/IEC, 1996).
 - [14] N.R. Jennings, An agent-based approach for building complex software systems, *Communications of the ACM* 44 (4) (2001) 35–41.
 - [15] N.R. Jennings, T.J. Norman, P. Faratin, P. O'Brien, B. Odgers, Autonomous agents for business process management, *Journal of Applied Artificial Intelligence* 14 (2) (2000) 145–189.
 - [16] N. Kang, S. Han, Agent-based e-marketplace system for more fair and efficient transaction, *Decision Support Systems* 34 (2) (2003) 157–165.
 - [17] E. Kavakli, Goal-Oriented Requirements Engineering: A Unifying Framework, *Requirements Engineering* 6 (4) (2002) 237–251.
 - [18] R. Kishore, H. Zhang, R. Ramesh, A helix-spindle model for ontological engineering, *Communications of the ACM* 47 (2) (2004) 69–75.
 - [19] R. Kishore, H. Zhang, R. Ramesh, Enterprise integration using the agent paradigm: foundations of multiagent-based integrative business information systems, *Decision Support Systems* 42 (1) (2006) 48–78.
 - [20] A.D. Kwok, D.H. Norrie, Intelligent agent systems for manufacturing applications, *Journal of Intelligent Manufacturing* 4 (4) (1993) 285–293.
 - [21] F.-R. Lin, Y.-H. Pai, Using multi-agent simulation and learning to design new business processes, *IEEE Transactions on Systems, Man and Cybernetics, Part A* 30 (3) (2000) 380–384.
 - [22] F.-R. Lin, G.W. Tan, M.J. Shaw, Multiagent enterprise modeling, *Journal of Organizational Computing and Electronic Commerce* 9 (1) (1999) 7–32.
 - [23] O.I. Lindland, G. Sindre, A. Solvberg, Understanding quality in conceptual modeling, *IEEE Software* 11 (2) (1994) 42–49.
 - [24] T.W. Malone, K. Crowston, The interdisciplinary study of coordination, *ACM Computing Surveys* 26 (1) (1994) 87–119.
 - [25] T.W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C.S. Osborn, A. Bernstein, G. Herman, et al., Tools for inventing organizations: toward a handbook or organizational processes, *Management Science* 45 (3) (1999) 425–443.
 - [26] I. Nonaka, A dynamic theory of organizational knowledge creation, *Organization Science* 5 (1) (1994) 14–37.
 - [27] M. Rossi, S. Brinkkemper, Complexity metrics for systems development methods and techniques, *Information Systems* 21 (2) (1996) 209–227.
 - [28] A.F. Salam, L. Iyer, R. Singh, Intelligent agents in supporting information sharing in B2B eMarketplaces, *Information Systems Management* 22 (3) (2005) 37–49.
 - [29] A.-W. Scheer, *ARIS-Business Process Modeling*, Springer, Berlin, 1999.
 - [30] J.R. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, 1969.
 - [31] K. Siau, Q. Cao, Unified Modeling Language (UML) — a complexity analysis, *Journal of Database Management* 12 (1) (2001) 26–34.
 - [32] R. Sikora, M. Shaw, Multi agent enterprise modeling, in: C. Holsapple, V. Jacob, H.R. Rao (Eds.), *Business Modeling: A Multidisciplinary Approach Essays in honor of Andrew B. Winston*, Kluwer Academic Press, 2002, pp. 169–185.
 - [33] R. Singh, A.F. Salam, L.S. Iyer, Agents in eSupply Chains, *Communications of the ACM* 48 (6) (2005) 108–115.
 - [34] J. Verschueren, *The Analysis of Speech Act Verbs: Theoretical Preliminaries*, Indiana University Linguistic Club, Bloomington, Indiana, 1977.
 - [35] J. Verschueren, *On Speech Act Verbs*, John Benjamins, Amsterdam, 1980.
 - [36] Y. Wand, R. Weber, An ontological model of an information system, *IEEE Transactions on Software Engineering* 16 (11) (1990) 1282–1292.
 - [37] Y. Wand, R. Weber, On the ontological expressiveness of information systems analysis and design grammars, *Journal of Information Systems* 3 (4) (1993) 217–237.
 - [38] H. Weigand, W.-J.v.d. Heuvel, Cross-organizational workflow integration using contracts, *Decision Support Systems* 33 (2002) 247–265.
 - [39] T. Winograd, A language/action perspective on the design of cooperative work, *Journal of Human-Computer Interaction* 3 (1) (1987–88) 3–30.
 - [40] M. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, Ltd., West Sussex, England, 2002.
 - [41] M. Wooldridge, N.R. Jennings, Intelligent agents: theory and practice, *Knowledge Engineering Review* 10 (2) (1995) 115–152.
 - [42] M. Wooldridge, N.R. Jennings, D. Kinny, The Gaia methodology for agent-oriented analysis and design, *Autonomous Agents and Multi-Agent Systems* 3 (3) (2000) 285–312.
 - [43] E. Yu, J. Mylopoulos, Why goal-oriented requirements engineering, *Proceedings of 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, Pisa, Italy, Presses Universitaires de Namur, 1998, pp. 15–22.
 - [44] H. Zhang, R. Kishore, R. Ramesh, Semantics of the MibML conceptual modeling grammar: an ontological analysis using the Bunge–Wang–Weber framework, *Journal of Database Management* 18 (1) (2007) 1–19.

Hong Zhang is an Assistant Professor of Computer Information Systems at Missouri State University. His research interests are computational ontology, enterprise integration, system analysis & design, multi-agent systems, and service-oriented architecture. His papers have been published or accepted for publication in *Communications of the ACM*, *Decision Support Systems*, *INFORMS Journal on Computing* and *Journal of Database Management*. He is an executive committee member of AIS SIG on Ontology-Driven Information Systems (SIG-ODIS).

Rajiv Kishore is an Associate Professor in the School of Management at the State University of New York at Buffalo. His primary research interest is in improving organizational and IT performance through the effective management of global IT and business process outsourcing projects, agile methods for business process analysis and integration, and technology and innovation management. His papers have been published or accepted for publication in *Journal of Management Information Systems*, *IEEE Transactions on Engineering Management*, *Communications of the ACM*, *Decision Support Systems*, *Information & Management*, *Information Systems Frontiers*, *Journal of Database Management*, and *Advances in Management Information Systems*, among others. Rajiv has presented his research at ICIS, HICSS, AMCIS, SIM, etc. He received a best paper award at AMCIS

2001 and was nominated for a best paper award at AMCIS 2003 and HICSS 2004. He also received a multi-year National Science Foundation research grant as a co-principal investigator in the area of IT outsourcing. Rajiv has consulted with a number of large companies, some of which include BellSouth, Blue Cross Blue Shield of Minnesota, IBM, and Pioneer Standard Electronics.

Raj Sharman is an Assistant Professor in the School of Management at the State University of New York at Buffalo. He received his B. Tech and M. Tech degree from IIT Bombay, India and his M.S. degree in Industrial Engineering and Ph.D. in Computer Science from Louisiana State University. His research streams include Distributed Computing, Decision Support Systems, Information Assurance, and Disaster Response Management. His papers have been published in a number of national and international journals. He is also the recipient of several grants from the university as well as external agencies. Raj Sharman is the co-author of the Springer-Verlag edited book entitled “Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems” along with Rajiv Kishore and Ram Ramesh.

Ram Ramesh is a Professor in the School of Management at the State University of New York at Buffalo. His research streams include Conceptual Modeling (Ontologies, Connectionist Modeling and Nonmonotonic Reasoning), Economics and technologies of Internet Capacity Provision Networks (CPN), and Database systems and distributed computing frameworks. His research has been funded by several DoD organizations and contractors including Army Research Institute (ARI), Air Force Office of Scientific Research (AFOSR), USAF Airforce Research Laboratory, Naval Training Systems Center (NTSC), Westinghouse, Raytheon and Samsung among others. He currently serves as an Associate Editor for *INFORMS Journal on Computing*, *Communications of the AIS*, *Journal of Semantic Web and Information Systems* and *Journal of Intelligent Information Technologies*. He is a co-Editor-in-Chief of *Information Systems Frontiers* and has edited volumes in *Annals of OR* and *CACM*. He is currently guest-editing an issue of the *Journal of AIS* on Ontologies in the context of Information Systems. He has published extensively in the above streams of research. His publications appear in journals such as *INFORMS Journal on Computing*, *Information Systems Research*, *IEEE/TKDE*, *ACM/TODS*, *IEEE/SMC*, *Naval Research Logistics*, *Management Science* and *Communications of the ACM* to name a few.