A Formal Compositional Model of Multiagent Interaction

Sea Ling
School of Computer Science and Software Engineering
Monash University, P O Box 197
Victoria 3145, Australia
sling@csse.monash.edu.au

Seng Wai Loke
School of Computer Science and Software Engineering
Monash University, P O Box 197
Victoria 3145, Australia
swloke@csse.monash.edu.au

ABSTRACT
Interaction protocols for multiagent systems have been described diagrammatically in an extension of UML called AUML (Agent UML). In this paper, we show how to translate these protocols to equivalent Petri net specifications. A novelty of our approach is that the Petri nets are modular, clearly separating the protocol from the interaction behaviour of agents induced by their participation in the protocol, yet compositional. We contend that compositionality is useful since multiagent systems and their interactions are inherently modular, and so that mission-critical parts of a system can be analysed separately. Our model can serve at least two purposes in multiagent systems engineering: firstly, specification and verification, and secondly, as a basis for synthesising skeleton code of interacting agents from specifications in the spirit of interaction-oriented programming.

Categories and Subject Descriptors
D.2.4 [Software Engineering]: Software Verification—formal methods; D.2.1 [Software Engineering]: Specifications

General Terms
Design, Verification

Keywords
Petri nets, interaction protocols, agent-oriented software engineering, compositional specifications

1. INTRODUCTION
In a multiagent system, agents communicate with each other via some interaction protocol. The protocol typifies a set of message passing communications to be executed by the agents in some correct order. A number of interaction protocols have been proposed by the organization FIPA (The Foundation for Intelligent Physical Agents) ¹, whose purpose is to provide software standards for interacting agents for agent-based systems. Examples of interaction protocols are the contract net protocol, the English auction protocol and the Dutch auction protocol. The protocols have been specified in AUML (Agent UML) [3], an extension of the Unified Modelling Language (UML). The notation used in the specification document thus far is a variant of UML’s sequence diagram developed for agent-based systems.

It has been proposed that each interaction protocol can be viewed as a pattern to be used as a “reusable aggregate of processing” [3]. In different problem domains, the pattern becomes a template that can be reused in such a way that the basic interaction and message sequencing remain the same while the agent roles and the message details will be modified to adapt to a different scenario. Such templates can be used by programmers as a guide when building their multiagent system. However, it often requires clever and careful programming to ensure that agents built do implement a particular protocol. How can we build agents that conform to a given interaction protocol? The challenge grows when the agent has to interact with different parties using different protocols or sub-protocols. An approach is to start from specifications whose correctness have been verified, and then carefully derive (perhaps automatically) code from the specifications. Further debugging, if needed, can then be done as an additional step.

2. MULTIAGENT INTERACTIONS
Our work begins with the simple observation that a given interaction protocol imposes particular constraints on the behaviour of participating agents. Informally but intuitively, an interaction protocol represented in AUML can be translated into a Petri net modelling the pattern of interaction [5], and each participating agents’ (or agent roles’) behaviour can be represented by a Petri net. The Petri net for each agent (or agent role) acts as a specification of the aspect of the agent’s behaviour that is induced by their involvement in the interaction protocol. Such a specification is in the spirit of the agent skeletons proposed in [7], where a skeleton captures the interaction aspect of an agent (with respect to that protocol). Since Petri nets are used, it is natural to use existing Petri net analysis methods [6] to analyse and verify the correctness of specifications. Because we start from the interaction protocol, we can view our approach as being interaction-oriented [7], i.e. a multiagent system is

¹http://www.fipa.org/
A multiagent system consists of a number of agents and a number of interaction protocols. Every pair of agents communicate with each other via one interaction protocol. This is modelled by a Multiagent Interaction Protocol Net (MIP-net) which is a combination of A-nets and IP-nets using some synchronising elements. The MIP-net definition is adapted from the definition of Interorganizational Workflow (IOWF) [2]. It should be noted that IOWF essentially combines several WF-nets describing processes from different organizations. We treat agents as individual processes and combine them with their interaction protocols to form a MIP-net.

The MIP-net in Figure 1 combines two A-nets (Initiator and Participant) and an IP-net (of the contract net protocol) by synchronising pairs of appropriate transitions illustrated by the dotted lines. Note that each agent participates only on one side of the protocol. Note also that we consider only synchronous communications between transitions because it is more natural to assume that a task activated by an agent will immediately and synchronously trigger a corresponding activity (transition) in the protocol.

The purpose of modelling multiagent systems in Petri nets is to make full use of the well-established analysis methods proposed for Petri nets. These methods are commonly used to detect the liveness and the boundedness properties of systems modelled [6]. We also want to use Petri nets to provide a notion of correctness for multiagent systems and then to be able to verify and analyse multiagent systems.

Since A-nets are essentially WF-nets [1], the correctness of an A-net is defined similar to WF-net correctness, which is called soundness. It has been shown that workflow soundness is related to liveness and boundedness properties [1]. We modify workflow soundness for agent soundness.

We find that some given A-nets may be sound, but a combination of these agents with an interaction protocol might be subject to protocol synchronization errors. The resulting MIP-net may not be “correct”. To verify whether a MIP-net is “correct”, we modify the “unfolding” technique proposed for IOWF [2]. The unfolding means to create a single, global A-net from the MIP-net. The result of an unfolding is a single Petri net which can then be subject to analysis, as mentioned previously. We define the soundness property of MIP-nets similar to IOWF-soundness [2], i.e. a MIP-net is sound if and only if every A-net in the system is sound and the unfolding of the MIP-net is also sound.

3. CONCLUSION

Compared with other recent works, our work on agent interactions has a different emphasis in that we intend to use Petri nets to develop minimal agent specifications using interaction protocols. Such a specification is in the spirit of the agent skeletons proposed in [7], where a skeleton captures the interaction aspect of an agent (with respect to that protocol). However, Singh’s work [7] does not use Petri nets or AUML descriptions. We view IP-nets as template components which can be reused in different environments. In order to use or reuse the templates, agents must satisfy the requirements as stipulated in the interaction protocol and must adhere to (at least) the minimal expected behaviour.

4. REFERENCES


