A Dedicated Approach for Developing Agent Interaction Protocols

Ayodele Oluyomi and Leon Sterling

Department of Computer Science and Software Engineering
The University of Melbourne, 111 Barry Street, Carlton, Victoria 3053, Australia
{aoo,leon}@cs.mu.oz.au

Abstract. Much current research is focussed on developing agent interaction protocols (AIPs) that will ensure seamless interaction amongst agents in multi agent systems. The research covers areas such as desired properties of AIPs, reasoning about interaction types, languages and tools for representing AIPs, and implementing AIPs. However, there has been little work on defining the structural make up of an agent interaction protocol, or defining dedicated approaches for developing agent interaction protocols from a clear problem definition to the final specification. This paper addresses these gaps. We present a dedicated approach for developing agent interaction protocols. Our approach is driven by an analysis of the application domain and our proposed structured agent interaction protocol definition.

1 Introduction

Interaction is generally recognized as an important characteristic of multiagent systems (MAS) [1, 5]. A widely acceptable view conceptualizes an agent as an autonomous agent possessing the ability to *interact* with other agents to achieve its goals and that of the multiagent system. Agent Interaction Protocols (AIPs) are used for managing and controlling the interactions in MAS [3].

There are two issues that are important in thinking about AIPs. The first one is that an AIP within a MAS has its particular identity, that is, a component part of the MAS that is present to serve the interaction needs of the agents while remaining separate from the individual agents within the system. Secondly, interaction in a MAS is context sensitive. The nature and structure of interaction, and therefore the structure and properties of the AIPs that will achieve a specific interaction, are dependent on the purpose and peculiarities of the domain of application of the MAS in which the specific interaction takes place.

Several Agent Oriented Software Engineering (AOSE) methodologies have been and are still being developed [2, 1, 4]. However, a review of these AOSE methodologies reveals that most of them do not have a clear process for developing AIPs that will be used in the MAS to be developed [1, 4].

The existing body of research work on AIP is largely focused on areas such as AIP specification methods [18, 19]; analysis of interaction types e.g. negotiation, argumentation, persuasion, etc and their underlying philosophies [12, 21, 6]; AIP concatenation and extension issues [22]; languages and tools for representing AIPs [7]; implementing AIPs [13], and so on. In as much as they all have their significant contributions to the agent world, they seem to be going in their individual directions

M.W. Barley and N. Kasabov (Eds.): PRIMA 2004, LNAI 3371, pp. 162–177, 2005.

with no convergence of these efforts and their results, because there is not much effort yet in developing a dedicated approach to engineering AIPs.

Further, a good number of the existing AIPs have weaknesses. A common major weakness is that AIPs do not have important properties such as termination and rule-consistency, which may limit their suitability for their application [6]. It is also important to note that different AIP properties or different combinations of these properties are required for different domains of MAS applications.

Interaction in MAS is high level and significantly context sensitive when compared with data communication protocols. The aspiration of the agent community is to make these interactions as close as possible to human interactions. This conceptual view and the aspiration therefore emphasize the context sensitivity of interagent interaction and therefore the AIPs that will guide these interactions [2]. The contexts of the domain of MAS applications determine the structure and properties of AIPs necessary to achieve effective and efficient interactions within the MAS. Issues such as time criticality, safety criticality, concurrency, control hierarchy, goal diversity and so on are domain dependent and these all influence the way and manner in which effective interaction in such systems should occur.

Most of the MAS development methodologies that consider interaction identify the interaction needs of the system and then implement an existing AIP. For instance, a MAS that requires a *Negotiation* type interaction may implement the FIPA Contract Net Protocol [8]. Although this suggests the software engineering concept of reuse, it does not always achieve desired results due to the following possibilities: the AIP chosen may be too generic for the intended application making it inefficient; the AIP chosen may not be comprehensive enough for the intended application; the AIP chosen may not have the desired properties such as safety, confidentiality, or timing constraints, to ensure rich context based interaction that are well suited for the intended application. Also, it is well accepted that inadequately planned reuse is counter productive.

To address these inadequacies, we propose a comprehensive and dedicated approach for developing AIPs with the aim of generating readily customizable and well structured AIPs that will be appropriate for their domains of application. There are two main drivers for this approach.

1. The first driver is the recognition of the impact of the peculiarities of individual domains of application on the AIP being designed. Based on this recognition, a comprehensive analysis of the domain of application of a MAS is necessary for the development of appropriate AIPs.
2. The second driver is the need for a structure for AIPs. Apart from the concepts of micro and macro protocols, nested protocols and concatenation of protocols, it is necessary to appropriately conceptualize the definition of agent interaction protocols. This will facilitate reusability as it will make it easy to customize a protocol where there is a clear structure to its definition.

In this paper, we present a new dedicated approach for developing AIPs.

The remainder of this paper is organized in the following way. Section 2 discusses the motivation for this work. The drivers for the proposed approach are presented in section 3 with a description of the proposed structure for defining AIPs. Section 4 presents a description of the new approach for engineering AIPs. In Section 5, we present an example of the use of our protocol structure in specifying the analysis of a

domain. Section 6 is a brief discussion of AIPs and the open problems surrounding them. Section 7 concludes the paper.

2 Motivation

In principle, a MAS is a system of *interacting* agents. Regardless of the complexities or sophistication or simplicity of the individual agents in any MAS, a common characteristic of such systems is interaction amongst the different agents within the system. According to [1], interaction is arguably the most important single characteristic of complex software which constitute a MAS. According to [15], interaction is identified as an essential component of the dynamics of the MAS. The significance of interaction in MAS is expressed in fundamental attributes of a software agent. An agent in a MAS that will be reactive and proactive in its sphere of operation, its environment, will do so via interaction with the other agents. Interaction Protocols are the concrete definition and means of implementation of the interactions in a MAS. Interaction Protocols give context and direction to the interactions in a MAS. Interaction is a driver of the overall behaviour of a MAS since an agent's perception of its environment is modified by results of interactions, and these modifications influence the agent's decision process [6]. Therefore AIPs are crucial aspects of the development, implementation and operation of the MAS.

AIPs are a somewhat different component of the MAS. Unlike the individual agents which take up specific roles [16, 4], AIPs have no function without at least two agents, and two agents cannot interact effectively without an AIP. Hence an AIP is defined in the light of a minimum of two agents engaged in an interaction. So where agents require attributes that will ensure the achievement of their goals, AIPs require attributes that will ensure that two or more agents with similar or divergent goals interact effectively in order to achieve their respective objectives. Examples of these properties include rule consistency, rule simplicity, inclusiveness, architecture independence, etc. These attributes define the identity of the AIPs and determine their success in achieving interaction in the domain of application.

The significance and peculiarity of AIPs in MAS as described above, demands a dedicated approach to the development of AIPs when building MAS. However many (if not most) of the existing AOSE methodologies do not define specific processes for developing AIPs [1, 6, 5]. Though there is doubtless a large body of research work in the area of AIPs, to the best of our knowledge, very little work is focussed on defining a dedicated process for the development of AIPs [5, 11, 3].

In consideration of the existing work on AIP development, we are of the view that:

1. AIPs resulting from a general approach may not be well suited to the particular application for which they were intended.
2. Assumptions about the character of AIPs are implicit, and not separate from the actual rules guiding the sequencing of the messages in the conversation [14].
3. A proper understanding of the essence of a protocol is lacking, for instance describing the same message structure only with different parameters as separate protocols.
4. A proper understanding of a protocol structure is lacking, for instance, the difference between a protocol and a performative is unclear.
5. Reuse of existing AIPs is hard to achieve.

The process of developing appropriate AIPs requires that the *problem* be well-defined [10]. A problem should be carefully defined before design tasks are undertaken. For the problem definition to be able to solve the problem appropriately, it needs to be expressed in the light of the problem domain.

Our dedicated approach for developing AIPs is motivated by the fact that interaction and therefore AIPs describe the peculiar characteristics and overall behaviour of a domain of MAS application. The consideration is, *negotiation* in a business to business transaction is different in some attributes from *negotiation* in a business to consumer transaction. While one may be critical on time, for instance a business to business transaction for raw materials required for production scheduling, the other may not be. Another example is the difference in the *cooperation* that is required amongst a set of agents assisting surgeons in a critical operation when compared to the *cooperation* among a set of telecommunications service provider agents in a bid to present a common tariff regime to their customers. We see from these two examples that the nature of the interaction reflects the peculiar characteristics of the domains. Hence, the right applicability of MAS to these domains is dependent on how the interagent interactions in the system are conceptualized and implemented. Also, these two examples also show that the differences in similar interaction types needed for different domains could either be subtle, requiring certain attribute modifications or fundamental, impacting on the entire structure of the interaction.

This proposed AIP development approach is situated in the context of the ROADMAP methodology [4, 9] for building MAS. This approach is an extension of the Interaction model component of the methodology. However, it is being designed with the concept of an AOSE feature [17] in mind such that it can be used for developing AIPs alongside other AOSE methodologies as well.

3 Structured AIP Definition

As expressed by the examples in the preceding section, we believe that understanding the domain of application of a MAS, in order to determine its peculiar features, is pivotal to the success of interaction in the MAS. Domain understanding should be the primary driver for the AIP development process. A clear analysis of the domain of application will provide the features of the AIPs required for interaction within the system. These features will define the identity of the AIP during the design to ensure adequate applicability and also, provide a proper basis for the verification of any AIP specified for the system.

We highlight three conceptual issues informing our work on AIP. The first is determining the component parts of a complete protocol. The second is how these component parts are related to or dependent on one another in describing a complete protocol. The third is specifying the mandatory and optional components for a protocol to be complete.

These issues relate to *protocol definition,* the secondary driver of our approach. Protocol definition significantly impacts on how AIPs are designed and serves as a basis for assessing the completeness of the protocol specified.

Our new structured definition states that an AIP is made up of two broad components. These are the *Protocol Structure* and the *Protocol Property Suite.* Our work on protocol structure is based on work on communication protocols [10]. However, we

present this in the concept of interagent interaction. We also show the way in which the component parts of the protocol structure are connected to model a complete AIP, see figure 1. The protocol property suite on the other hand, defines the collection of the values of the properties of a particular protocol being specified. Our claim is that a protocol changes to become another protocol by changes to its properties, as these define the protocol's structural component and therefore its function and identity.

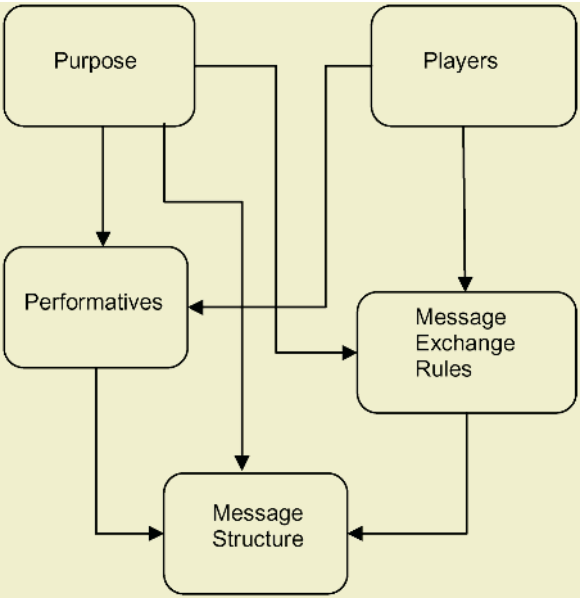

Fig. 1. Agent Interaction Protocol Structure

3.1 AIP Structure

Purpose: the purpose component of the protocol structure is the most significant component. It is the representation of the analysis of the interaction that the protocol is being designed to implement. This component provides a basis for the specification of the interaction model (specified in the Players component) in the light of the characteristic features of the interaction domain. It is not just a description of the essence of the AIP to be built, it is a structured specification that describes the interaction behind the AIP. Hence, this component determines the kind and structure of the messages and the rules that will guide the exchange of these messages amongst the interacting agents. The purpose specification also describes the properties of the protocol and these influence the other components of the protocol structure. See Table 1 for the definition of the elements of the Purpose component.

Players: the agents involved in the interaction are described as the players. This component of the protocol structure documents the interacting agents and their roles within the MAS. The relationships between these agents are also specified in this component. These relationships include organizational hierarchies, buyer/seller, competitor relationships and so on. Based on the relationships, restriction on interaction

Table 1. Elements of the Purpose component

Interaction:	A statement of the interaction e.g. Stocks Transaction
Related System Goal:	A statement of the system goal that requires the interaction e.g. optimize investments
Domain of application:	A specification of the domain stratification e.g. Business-Stock Exchange-Stock Market
Domain type:	A specification of the type of the domain e.g. open, distributed, closed, real-time, etc
Interaction objective:	A description of the essence of the interaction within the system and in the context of the domain
Interaction type:	A specification of the type of interaction e.g. negotiation, collaboration, competition, etc
System Safety:	A statement of the impact of this interaction on the physical safety of the system
Pre conditions:	A specification of the system state necessary to trigger the interaction e.g. presence of an open order on the stock exchange
Post conditions:	A specification of the expected system state (or possible states) after a successful completion of the interaction e.g. the stock is purchased

involvement to either any or specific instances of certain agent roles are specified in this component as well as the part (initiator, participant or responder) to be played by particular agents in the interaction. The players component also describes the interaction mode i.e. bilateral or multilateral interaction. See Table 2 for the definition of the elements of the Players component.

Performatives: this component is a listing of all the performatives that will be used in the AIP and their meanings. It is important to clearly define the meaning of the performatives in order to avoid misinterpretation by the interacting agents. Connections between a performative and an agent or interaction part (initiator or responder) are specified in this component, for instance, the specification that a performative is an interaction initiating performative. Also, the number of times it is permitted to have a performative in an interaction is specified in this component.

Message Structure: a message is defined by a performative, however, the structure of each message in the interaction is a specification of the information that the message will carry when it is sent. The different fields that each message should have and a representation of information in each field are specified in this component of the protocol structure.

Message Exchange Rules: this component of the protocol structure defines the characteristic behaviour of the interaction. This presents a specification of the different guidelines that direct how messages are exchanged in order to efficiently and effectively realize the interaction. The specification includes how an interaction should be initiated, how the interaction should end, message exchange mode, timing constraints between receiving and sending of messages, and so on.

Table 2. Elements of the Players component

Interacting agents:	A specification of the type of agents involved in the interaction
Initiator:	A specification of the agent that initiates the interaction
Responder(s):	A specification of the agent(s) allowed to respond to the Initiator
Inter-agent Relationships:	A description of the association between the interacting agents e.g. Client/Server, Buyer/Seller
Priviledges:	A specification of the permission given to one of the participants to change the rules of the interaction
Number of agents:	A specification of the number of agents to be involved in the interaction if known(or range i.e. Greater than two, if the number is not known)
Diversity of agents:	A description of the source of the agents in the interaction i.e. Heterogeneous or Homogeneous agents
Distribution:	A description of how the initiator(s) connect with the responder(s) based on the number of each category of agents interacting i.e. One-to-Many, Many-to-One, Many-to-Many, etc
Accessibility:	A specification of the initiator agent's awareness of the other participants and how to contact each of them (addresses). There could be Complete or Partial or No Accessibility
Inclusiveness:	A specification whether the number of participating agents is fixed or variable at the start of the interaction

The different components of the protocol structure are connected to one another, as shown in figure 1, to show how one component influences the content of another one. These connections reveal the part a component plays in specification of other components. These connections present a relationship amongst the components and they give a better conceptualization of the motivation for this proposed AIP definition. This relationship provides a guide for the sort of information and specification that should be stated in the different components.

The purpose component determines the content of the performatives, message structure and message exchange rules components. The purpose of the interaction dictates the number and type of performatives required to achieve such an interaction. The message structure is affected by the purpose as some of the information to be included or not included in the message structure will be determined by the purpose of the interaction. For instance, a purchase interaction between a buyer and a seller requires settlement details in one of the messages while an advertisement interaction between a seller and potential buyer may or may not include only modes of payment and not settlement details. The major determinant of the message exchange rules is the purpose component since the purpose details what the interaction seeks to achieve, how critical it is to the system, how quickly the interaction should happen, how it should end, how it should handle errors in the interaction, and so on.

The players specification influences the performatives and the message exchange rules since some of the players may have overriding roles in the interaction, hence considerations will be given to such roles where they exist, in defining the performatives and the message exchange rules. Performatives affect message structure since a performative gives meaning to a message. Also, message exchange rules in some cases influence the message structure, an example is where a message exchange rule states that 'the interaction initiating message must state the purpose of the message' (where this is not part of the performative's semantic meaning), the structure for such a message will therefore include a field for this information.

It is needful to specify these details explicitly because it helps in achieving uniform protocol interpretation and also in appropriate interaction error handling. For instance, if a responder is sending a *cfp* message, which has been declared in the performatives component to be an initiator performative or if a performative that is defined to be used only once is being used a second time within the same interaction, it indicates a high likelihood of an error or an exception in the interaction being executed.

3.2 AIP Property Suite

The properties of a protocol are the features or attributes that define the protocol's identity. Each of these properties has more than one possible value. The set of values of the properties applicable to an AIP make up the Protocol Property Suite, according to our definition. As the interaction is analyzed in the context of the domain, the properties that are applicable to the protocol to be designed are identified. The values of the identified properties make up the protocol property suite for this particular protocol. See Table 3 for a brief description of the properties.

We differentiate the protocol properties which are integral to our definition, from the quality attributes that the protocol is expected to have in the larger context of the MAS that the protocol is a part of. It is needful to separately represent the protocol property suite because it makes it a lot easier to modify and upgrade a protocol and also to make another protocol out of an existing one easily.

To illustrate some of the concepts we introduce here and the claim that two AIPs with the same protocol structure will differ in function by their properties, we present the following example:

Consider two AIPs P_A and P_B. They both have the same set of performatives *ask, tell* and *end* with the message sequence in the order *ask – tell – end* (an example of a subset of their protocol structure). A property *timing sensitivity* with values *False* for P_A and *True* for P_B (the values *False* and *True* for property *timing sensitivity* represent the protocol property suite) will differentiate the behaviour of the protocols by defining the following message exchange rules for the two protocols.

P_A*:* An agent A that sends the message *ask* to another agent B does not send the message *end* to close the interaction until it receives the message *tell* within a time interval of 0 – 300 seconds, after which it may close the interaction with the message *end*.

P_B*:* If an Agent A does not receive a message *tell* within a time interval of 0 – 5 seconds after sending the message *ask* to an agent B, A should send the message *ask* to another agent C. If A receives a *tell* message from either of B or C within 5 seconds of sending to C, A closes both interactions by sending *end* messages to B and C, otherwise it closes interaction with B alone.

Table 3. Elements of the Protocol Property Suite component

Timing constraint:	A description of the impact of time in achieving the interaction objective e.g. bid submission is deadline constrained
Security concerns:	A specification of the impact of this interaction on the security concerns of the system (e.g. confidentiality of information exchanged)
Error sensitivity:	A description of the sort of error (content and control) that the interaction can cope with e.g. high error sensitivity in air traffic control related interaction
Messaging mode	A specification of the message sequencing mode i.e. Asynchronous or Synchronous
Messaging mechanism	A specification of method of sending messages to the intended recipients i.e. broadcast, multicast
Interaction mode	A specification of the number of agents that an agent can simultaneously connect with for message exchange i.e. Multilateral, Bilateral
Ontology	A specification of the uniformity or otherwise of the participating agents' representation of the real world

This is a clear instance of how the values of a property, will differentiate the functions of two AIPs with the same structure. AIP P_B represents a time critical system, while P_A represents a system that is not time critical. The essence of expressiveness is evident by this illustration as the level of details in specifying the protocols will reduce ambiguity in the protocol interpretation.

Our protocol definition is conceptualized as follows. The Purpose and Players specifications *generate* the Protocol Property Suite as well as the Performatives, Message Exchange Rules and Message Structure. The protocol property suite is used to *define* the Message Exchange Rules, the Performatives and the Message structure, see Figure 2. This presents a clear relationship amongst the different aspects of our protocol definition and is useful in specifying an approach for protocol engineering.

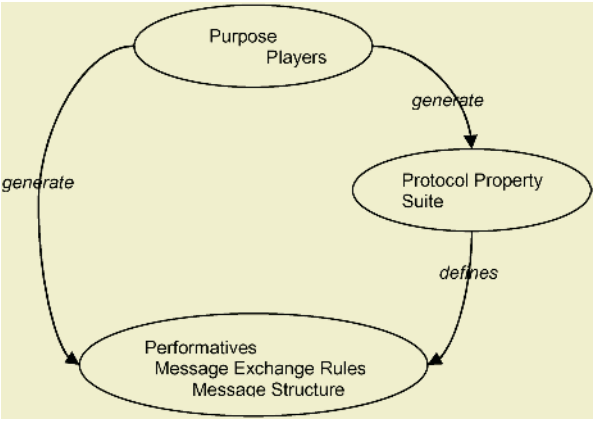

Fig. 2. Relationship between Protocol Structure and Protocol Property Suite

4 Our Dedicated Approach for Developing AIPs

In the preceding section, we presented the structured AIP definition, one of the drivers for this approach. Here, we present a brief description of the new approach. Our approach to AIP engineering is a 2 phase model consisting of the domain-directed analysis and the design/verification phases, see figure 3. This approach focuses on analysis and design since an AIP can not be implemented outside of a MAS implementation. Also, it is our intention that this AIP development approach will be applicable to different MAS development methodologies.

4.1 Domain-Directed Analysis Phase

The analysis phase of the AIP engineering process is divided into two stages. The first stage is the actual analysis which is carried out in the context of the domain of application. The characteristics of the domain are the basis for the analysis in order to draw out the requirements of the AIP to be engineered. Identity is given to the protocol developed and its appropriate applicability is assured when its requirements are analyzed in the context of the domain. Examples of domains of application of MAS technology include the smart home (a network of intelligent appliances), air traffic management, medical applications, internet based e-markets and so on. The domain-directed analysis is carried out using the following process:

1. At the completion of the MAS requirements analysis, identify the system goals that require more than one agent to achieve them.
2. For system goals that require more than one agent to fulfill them, define if the agents need to interact with one another or with external sources in order to achieve the goal.
3. For system goals that are achievable by only one agent, define if the agent requires information or assistance (resources) from other sources in order to achieve the goal.
4. Where interaction is identified to be necessary from steps 2 and 3 above, analyze the goal and the domain of application in order to define the elements of the Purpose component for each interaction required.
5. Identify the agents that are required for each interaction.
6. Analyze the goal, the domain of application and the agents involved in each interaction in order to define the elements of the Players component for each interaction.
7. Using the Purpose and the Players component and analysis of the domain of application, define the Protocol Property Suite.

The outcomes of this phase of the engineering process are the specifications of the Purpose and Players components and the Protocol Property Suite of our protocol definition. These components present a detailed and structured representation of the analysis in the context of the domain such that it can be effectively translated into design with very minimal ambiguity.

The second stage in the analysis phase of our approach is the search for existing AIPs. This is separately represented and emphasized to show our recognition of the existing body of work on AIP specifications and to emphasize our consideration for reuse. The specifications generated from the analysis are used as a basis to search for

an existing AIP that is most suited to the AIP to be developed. The outcome of the directed search could be an existing AIP that suits the intended AIP, an existing AIP that needs to be modified to suit the intended AIP or no existing AIP that is close to the intended AIP, hence requiring design from scratch.

4.2 Design/Verification Phase

The design phase of our AIP development approach has two possible paths depending on the outcome of the directed search in the analysis phase. Where a similar existing AIP is found, the reuse path is taken. If no similar AIP is found, the develop path is followed. Where an AIP that matches the intended AIP is found after the directed search, the process proceeds to the Verification phase, Figure 3.

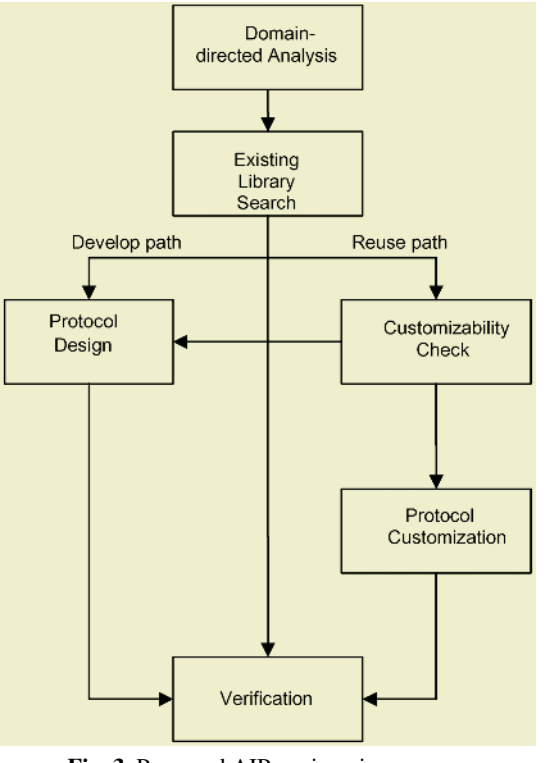

Fig. 3. Proposed AIP engineering process

Where the design follows the reuse path, the first stage in reuse is to determine if the existing AIP can be modified to make it fit into the requirements of the intended AIP. A major consideration at this stage is to assess the *cost* of modifying in terms of time and effort, against that of developing the protocol from scratch. Some of the things to consider in determining if an existing protocol can be readily modified include method of specification, understanding of the heuristics or logic behind the design, complexity of the protocol, etc. If it is determined that this existing protocol can be readily modified, the next stage along this design path is to customize the protocol using the specifications from the analysis phase.

Where there is no existing AIP that matches the intended AIP, or the existing AIPs are not customizable, the develop path is followed i.e. the intended AIP is designed from scratch. The develop path of the design phase starts by defining the set of performatives to be used by the AIP and their semantics using the specifications from the analysis phase. Then the message exchange rules are defined and a model of the message structure for each of the performatives is also defined. Subsequently, the protocol is graphically specified. Here, we propose the use of Scenario-Based Programming (SBP) for graphically specifying the protocol. Scenario-Based Programming is based on the formal language of Live Sequence Charts [20]. SBP representation of an AIP creates expressive specifications of AIPs.

Verification of the AIP developed (specified or modified) and the unmodified existing AIP is the final stage of the design/verification phase of our AIP engineering process. The verification process is dependent on the method used in specifying the AIP. SBP has automated techniques for carrying out the verification of the accuracy of the specified agent interaction protocols. The design/verification phase is iterative. It is repeated until the verification proves that the protocol has been properly specified.

The relationship between our protocol definition and the protocol engineering process is shown in figure 4. The analysis phase of the *Process* generates the three *Products* Purpose and Players specifications and the Protocol Property Suite. The design/verification phase generates the performatives, message structure and message exchange rules.

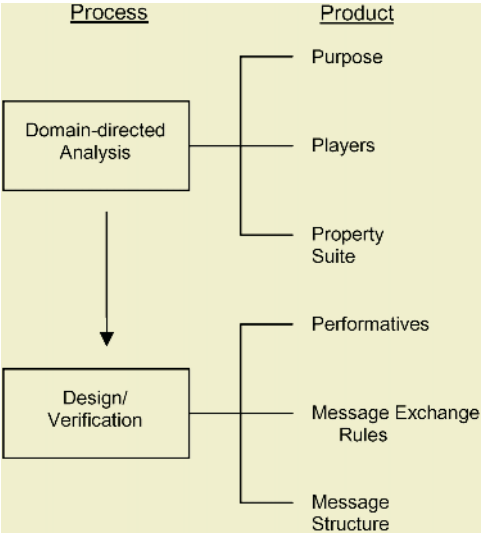

Fig. 4. Relationship between Protocol definition and Protocol development process

5 Example

In this section, we illustrate the use of our protocol structure in describing the domain analysis for developing an AIP. The AIP used for this illustration is the Provisional

Agreement Protocol (PAP) for Global Transportation Scheduling [23] developed for interaction in military operations transportation scheduling. According to [23], a typical military transportation operation is to move large quantities of resources on a global scale. As a result, a transportation operation may require the services of many transportation organizations. Each of these transportation organizations is usually only capable of moving a portion of the quantity of the resource through only a portion of the distance to be covered. The domain is open and dynamic. Transportation organizations enter and leave the system at will with the possibility of their capabilities continually changing.

We present the specifications of the Purpose, Players and Protocol Property Suite components of the PAP in the following tables, 4, 5, and 6:

Table 4. PAP Purpose component

Interaction:	Transportation Scheduling
Related System Goal:	Plan Logistics
Domain of application:	Military Operations - Transportation
Domain type:	Decentralized, Open, Dynamic
Interaction objective:	Efficient scheduling for transporting large quantities of resources globally
Interaction type:	Complex negotiations (allowing partial quantity and route bids and backtracking)
System Safety:	Interaction has no direct impact on the physical safety of the system
Pre conditions:	A minimum quantity q of resources is to be moved over a minimum distance d over a time t
Post conditions:	A comprehensive schedule within time frame A conclusion that task is not feasible within time frame

Table 5. PAP Players component

Interacting agents:	Manager Agents - Military Organization Transportation Agents - Transportation Organizations
Initiator:	Manager Agent
Responder(s):	Transport Agents
Inter-agent Relationships:	Client / Service Provider
Priviledges:	None
Number of agents:	Greater than two
Diversity of agents:	Heterogeneous
Distribution:	One-to-Many
Accessibility:	Complete
Inclusiveness:	Variable

6 Discussion

AIP is a peculiar kind of software as it serves as the software *infrastructure* for interacting agents in a system that seeks to closely model real world interactions. The behaviour of the real world system being modeled is represented and implemented by

Table 6. PAP Protocol Property Suite component

Timing constraint:	Time sensitive. Bids are deadline driven
Security concerns:	Low
Error sensitivity:	High
Messaging mode	Asynchronous
Messaging mechanism	Broadcast
Interaction mode	Multilateral
Ontology	Uniform

the AIP. AIPs are different from communication protocols as AIPs bring contextual dimension into the interactions they implement instead of merely transporting data packets with some convention. The context of an application domain is a fundamental consideration in conceptualizing and developing AIPs. Therefore, a dedicated approach is required to develop well suited AIPs in a manner that makes them readily reusable.

Most of the existing work on AIPs focus on either design or implementation without a dedicated approach for developing AIPs. Also, the crucial aspect of application domain analysis is not given the attention it demands. As a result, existing AIPs, which may not necessarily be appropriate in modeling the particular system interactions, are plugged into MAS development projects. Software quality attributes depend on the context of the application domain. To achieve good software quality, dedicated engineering approaches are required for the aspect of the software being developed [17], in this case, AIPs.

7 Conclusions

This paper presents a new dedicated approach for developing agent interaction protocols. This approach, which specifies the Analysis and Design/Verification phases of the development process, is driven by the analysis of the characteristics/peculiarities of the domain of application as they affect interaction. We also propose a new structured definition for agent interaction protocols as a second driver for the new approach. Our aim is to present a well defined and reusable process for the design and development of AIPs based on an AIP structure that facilitates productive reusability. This paper presents the description of the proposed protocol structure, establishing the link between the components of the structure. Also, we present a brief description of the approach in this paper. Work continues to further describe the process, procedures and products of the approach.

References

1. Wooldridge, M. and Ciancarini, P. Agent-Oriented Software Engineering: The State of the Art. In Agent-Oriented Software Engineering. Ciancarini, P. and Wooldridge, M. (eds), Springer-Verlag Lecture Notes in AI Volume 1957, 2001.
2. S Bussman, N.R. Jennings, M. Wooldridge. Re-use of interaction protocols for agent-based control applications 73-87 Electronic Edition (Springer Link). AOSE 2002, Bologna Italy.

3. J. L. Koning. Compiling a conversation policy's Implementation from its validated specification model. International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA, June 2000.
4. Juan, T., Pearce, A. and Sterling, L., ROADMAP: Extending the Gaia Methodology for Complex Open Systems, Proceedings of the 1st Int. Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), p3-10, Bologna, Italy, July 2002.
5. M.-P. Huget and J.-L. Koning. Requirement analysis for interaction protocols. In V. Marik, J. Mueller, and M. Pechoucek, editors, Proceedings of the Third Central and Eastern European Conference on Multi-Agents Systems (CEEMAS 2003), Prague, Czech Republic, June 2003.
6. P. McBurney, S. Parsons, and M. Wooldridge. Desiderata for agent argumentation protocols. In Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02), Bologna, Italy, July 2002.
7. S. Paurobally and R. Cunningham. Achieving common interaction protocols in open agent environments, AAMAS, 2002.
8. FIPA Specification. Foundation for Intelligent and Physical Agents, http://www.fipa.org/repository
9. Juan, T. and Sterling, L., A Meta-model for Intelligent Adaptive Multi-Agent Systems in Open Environments (Poster), Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Melbourne, Australia, July 2003.
10. G.J. Holzmann. Design and Validation of Computer Protocols. Prentice Hall, November 1990.
11. J.L. Koning, M.P. Hugget, Interaction Protocol design: Application to an agent-based teleteaching project. The Second IEEE International Conference on Cognitive Informatics (ICCI'03). August, 2003
12. J.L. Koning. Designing and testing negotiation protocols for electronic commerce applications. 34-60 Electronic Edition (Springer LINK)
13. R. König: State-Based Modeling Method for Multiagent Conversation Protocols and Decision Activities. Agent Technologies, Infrastructures, Tools, and Applications for E-Services 2002: 151-166
14. M. Greaves, H. Holmback, and J. Bradshaw. What is a conversation policy? In F. Dignum and M. Greaves, editors, Issues in Agent Communication, Lecture Notes in Artificial Intelligence 1916, pages 118--131. Springer, Berlin, Germany, 2000
15. A. E F-Seghrouchni, S. Haddad, H. Mazouzi. A formal study of interactions in multi-agent systems. In Proceedings of ISCA International Conference in Computer and their Applications (CATA `99), April 1999.
16. Wooldridge, M., Jennings, N. and Kinny, D. The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems 3 (3). 2000, 285-312.
17. Juan, T., Sterling, L., Martelli, M. and Mascardi, V.,Customizing AOSE Methodologies by Reusing AOSE Features, Proc. 2nd Int. Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Melbourne Australia, July, 2003, pp. 113-120.
18. S. Paurobally, R. Cunningham, and N. R. Jennings. Developing agent interaction protocols using graphical and logical methodologies. In Workshop on Programming MAS, AAMAS, 2003.
19. J. Odell, H.V.D. Parunak, B. Bauer. Representing Agent Interaction Protocols in UML. Agent-Oriented Software Engineering, P. Ciancarini and M. Wooldridge eds., Springer-Verlag, Berlin (2001), 121--140.
20. D. Harel and R. Marelly. Come, Let's Play: Scenario-Based Programming using LSCs and the Play-Engine. Springer-Verlag, 2003.

21. C. Bartolini, C. Preist, N.R. Jennings. Architecting for reuse: a software framework for automated negotiation. Proc. 3rd Int Workshop on Agent-Oriented Software Engineering, Bologna, Italy, 87-98.
22. M.H. Nodine, A Unruh. Constructing robust conversation policies in dynamic agent communities. Technical Report MCC-INSL-020-99, Microelectronics and Computer Technology Corporation, 1999
23. Don Perugini, Dale Lambert, Leon Sterling, and Adrian Pearce. Provisional Agreement Protocol for Global Transportation Scheduling. In Workshop on agents in traffic and transportation held in conjunction with the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS), New York, 2004.