TOWARDS PROBLEM SOLVING METHODS IN MULTI-AGENT SYSTEMS

Paul Bogg[1], Ghassan Beydoun[2] and Graham Low[1]
[1]*University of New South Wales, Australia*
[2]*University of Wollongong, Australia*

Keywords: Knowledge engineering, Multi-agent system methodologies, Problem-solving methods.

Abstract: Problem Solving Methods (PSM) are abstract structures that describe specific reasoning processes employed to solve a set of similar problems and have proved very effective at enhancing reuse and extensibility in developing knowledge-based systems. We envisage that off-the-shelf PSMs can similarly assist in the development of agent-oriented solutions using Multi-Agent Systems (MAS). A challenge towards the effective use of PSMs in MAS is that current approaches to formulating PSMs do not adequately address the complexity of problems to which agent-oriented systems are suited. Towards addressing this, this paper focuses on providing an approach to guide developers in adequately formulating PSMs for complex problem-solving where interactions are involved, such as in domains where negotiation and cooperation are essential for solving a problem.

1 INTRODUCTION

The demand for agent-oriented software motivated the creation of new development approaches, such as INGENIAS (Pavon et al., 2005), Tropos (Bresciani et al., 2004) and MOBMAS (Tran & Low, 2008). None adequately addressed extensibility, interoperability and reuse other than (Beydoun et al., 2007; Tran & Low, 2008) where it was argued that an ontology-based approach is needed for a truly domain-independent agent-oriented development.

Following the reuse paradigm promoted in knowledge-based systems development (Schreiber et al., 2001), the work in (Beydoun et al., 2006) proposes a process that revolves around a *domain-dependant ontology* to build individual agents with *problem-solving methods* (PSMs). PSMs are high-level structures describing a reasoning process employed to solve general types of problems (Fensel et al., 2002). Continuing the work in (Beydoun, Tran et al., 2006), we envisage that engineering problem-solving knowledge as domain-independent ontology-based PSM structures is beneficial towards achieving domain-independent agent-oriented methodologies and systems. A library of these PSMs would assist the development of agent-oriented systems in domains where existing problem-solving knowledge may be reused. A set of modular, reusable problem-solving components has the potential to reduce development costs and speed up the development process. More specifically, this paper investigates the role that *task* and *problem-solving knowledge* play, arguing that current approaches to PSMs do not adequately address the complexity of problems to which agent-oriented systems are suited. In particular, where problem-solving software components are dependant on interactions appropriate PSMs that address interaction functionality have not been fully investigated. This paper proposes an extension to PSMs with an additional interaction dependency construct through which interaction specific problem-solving knowledge can be used. *Interaction-specific PSMs* describe knowledge about interactions for problem-solving, and how to design methods to resolve complex problems where interactions are necessary. Negotiation is used as brief example of how PSMs may be used to design MASs for interaction dependent problem-solving.

2 RELATED WORK

By using a domain ontology and an appropriate PSM, it was envisaged that knowledge based

Bogg P., Beydoun G. and Low G. (2009).
TOWARDS PROBLEM SOLVING METHODS IN MULTI-AGENT SYSTEMS .
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 308-313
DOI: 10.5220/0002252603080313
Copyright © SciTePress

systems can be easily developed as new problems are encountered (Figure 1) (Studer et al., 1998).

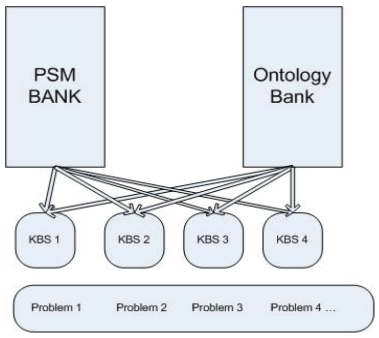

Figure 1: As new problems arise, the PSM and the ontology banks are used to construct suitable KBSs. An ontology from the *ontology bank* strengthens a given PSM from the *PSM bank* to suit the domain.

Recently, approaches have begun to address the elicitation of PSMs from common problem-solving knowledge. CommonKADS (Schreiber, Akkermans et al., 2001), is a prominent approach which provides a *Task model,* which provides a hierarchical description of tasks, and an *Expertise model*, which provides the method for achieving a task. CommonKADS provides reusable task-specific PSMs for composing the Expertise model to solve a variety of pre-determined types of tasks (such as diagnosis). Knowledge engineering also leverages ontologies for eliciting and developing domain-independent and reusable PSMs. One early approach by (Fensel et al., 1997) tackled reusability by incorporating ontologies for domain, task, and PSM-specific knowledge. Another approach, OntoKADS, extends CommonKADS by way of introducing ontologies to comprise the expertise model (Bruaux et al., 2005).

UPML (Fensel, Motta et al., 2002) encapsulated previous approaches to describing general task and problem-solving knowledge with general ontology-based PSM structures. One limitation in UPML is the absence in consideration given to PSMs for tasks where multiple software components are required to interact in order to solve a problem. For instance, e-commerce problem-solving agents negotiate for trade. Interaction-dependent problem-solving (such as negotiation) is prevalent in agent-oriented systems. To leverage the benefits of PSMs in AOSE, PSM structures addressing interaction-dependent problem-solving need to be developed. Recent approaches to incorporating PSMs into agent-oriented architectures have not addressed this. MAS-CommonKADS (Iglesias & Garijo, 2005) advocates task and problem-solving knowledge use in its methodology. However, it presumes the existence of PSM libraries suitable for complex, interaction-dependent problem-solving. The ORCAS framework (Gómez & Plaza, 2007) introduces methods to adapt PSMs to agent capabilities. Their work addresses cooperation as "agent teams" at the knowledge level. However, it doesn't address interaction dependent problem-solving knowledge required for negotiation. This is the focus here.

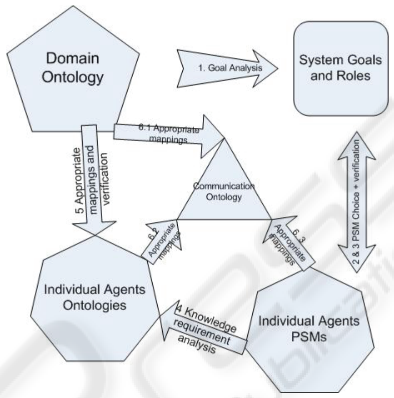

Figure 2: Ontology-based MAS development using PSMs: (1,2) Domain Ontology produces Goal Analysis is used to select PSMs from a PSM bank. (3, 4) Knowledge analysis delineates local agent knowledge. (5, 6).

In (Beydoun, Tran et al., 2006), software engineering requirements to use PSMs were mapped out resulting in a methodological model (Figure 2). This work did not address the issue of how to best formulate the PSMs for interaction-dependent problem solving. This paper continues this work by formulating an appropriate way to construct PSMs for distributed multi-agent systems (MAS). Much previous work has gone into integrating ontologies and PSMs e.g. (Fensel, Motta et al., 1997). It is not yet clear whether that work needs to be extended for the integration of domain ontologies with PSMs for MAS. Invetigating this is left as future work.

3 FORMULATING PSMS FOR MAS

Three types of knowledge are consistently identified in formulating a PSM structure (e.g. in (Decker et al., 1999; Fensel, Motta et al., 2002)): *domain knowledge, task knowledge,* and *problem-solving knowledge*. In these terms, PSMs are structured *problem-solving* knowledge suited to achieving *tasks/goals* in particular *domains*. UPML (Fensel, Motta et al., 2002) defines a PSM in terms of these knowledge components. Complex distributed problems to which MASs are suited to solve may require interactions between agents to coordinate solutions. Towards MAS-specific PSMs, this section extends the UPML PSM definitions. It adds a new construct notation, *interaction dependencies*, noting that when multiple agents are required to solve a particular problem then further analysis is required to determine what type of interaction is necessary.

When problem-solving depends on interactions, further consideration needs to be given towards understanding how different PSM definitions are related. This needs to be accounted for in order to properly formulate PSMs for MAS. For instance (Fig. 3), in designing two agents required to coordinate building a house, PSMs for a carpentry agent may *depend* on PSMs for a brick layer agent. Where this type of relationship between PSM definitions exists, we use the term PSM *co-dependency*. Where co-dependencies exist between PSM definitions for *separate* agents, we use the term PSM *interaction dependency* to specifically mean that agents may be required to interact with one another in order to successfully solve problems.

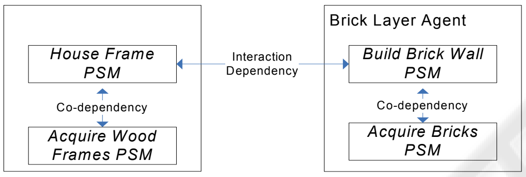

Figure 3: Agent-level PSM composition.

Interaction dependent PSMs bring additional dynamics to a MAS software development process. Firstly, interaction dependent PSMs suggest the presence of additional methods and/or agents during an analysis phase. Secondly, interaction dependant PSMs may assist in designing the interaction structure between agents by suggesting what type of exchange is required between agents. The type of exchange required might be as simple as an enquiry, or as sophisticated as negotiation. Thirdly, since interaction dependant PSMs are ontology-based, reuse (as suggested in (Breuker, 1999)) is a natural feature for future MAS development.

From an individual *agent-level* perspective, for distributed problems in which agents are required to interact, an interaction dependent PSM may be aimed at achieving agent-level goals. For instance, a negotiating agent may have a ‘Buy Item PSM’ that depends on negotiation to satisfy an agent-level goal to acquire a good. However, a software engineer may not only be interested in agent-level goals, but may also be interested in *system-level* goals.

From a *system-level* perspective, another type of relationship may exist between PSMs. As is illustrated in interaction-dependent problem-solving literature (such as in negotiation (Jennings et al., 2001)), sometimes the software engineer is interested in designing agents whose interactions produce system-level properties. For instance, optimal utilitarian agreements can be engineered by designing negotiating agents to use a correct combination of strategies under particular circumstances (e.g. (Fatima et al., 2004)). A PSM approach may be used to engineer systems where the selection of Strategy PSM ‘A’ suggests that the selection of another Strategy PSM ‘B’ facilitates system-level properties (such as utility optimisation) in addition to agent-level goals. PSMs with *system-level dependencies* may be used to design agent interactions such that system-level goals are achieved without resorting to “agent teams” (e.g. (Gómez & Plaza, 2007)) – coordination and cooperation are achieved at the agent-level. (Müller, 2002) argues that this may produce agent-oriented systems more widely applicable to general types of problems.

Table 1: Examples of interaction dependent PSMs.

PSM	*PSM*	*Interaction Dependency*
Buy Item	Sell Item	Negotiation for trade
Compensate for failure of agent Y	Compensate for failure of agent Y	Coordination to continue system operation during component failure
Procure Service for consumer	Provision Service to consumers	Negotiate terms of service agreement

Examples of interaction dependent PSMs are provided in Table 1. The first example is PSMs for commercial activities requiring interactions to achieve individual agent goals. The type of interaction required may be a simple retail exchange, or be a complex multi-issue negotiation. A system-level goal might be that all agent-level interactions are optimal according to some criteria (e.g. utilitarian optimal in (Fatima, Wooldridge et al., 2004)). The second example may appear in MASs where robustness is an important system-level requirement. PSMs may be needed to design coordinative actions assuring compensation during component failure (for instance, sensor agents compensate for the loss of other sensor agents in a battlefield information system (Deloach et al., 2008)). The third example may occur where agents procure service level agreements (for instance, in acquiring satellite and cable channels for television viewing, such as in (Cattoni et al., 1999)).

These knowledge engineering-based guidelines may be used in designing PSM repositories for interaction dependent problem-solving knowledge for use in AOSE. During AOSE analysis PSM repositories may not contain all relevant PSMs, and may need to be refined or developed – this process is not described here, and left as future work.

4 INTERACTION-SPECIFIC PSMS

Applying the insights of the previous section, this section adds new constructs to UPML to accommodate complex interactions used to formulate our new type of PSMs, *interaction-specific PSM.* This assists the designers of MASs by providing a structure to interaction-specific problem-solving knowledge. This new type is needed wherever interaction dependent PSMs suggest the exchange between two agents is sophisticated (such as negotiation, coordination or cooperation).

Interaction-specific PSMs are intended to be reusable. Knowledge about interaction-dependent problem-solving is reusable in different domains, and for different tasks e.g. similar methods for negotiation in e-commerce trade might be adopted in the negotiation of free trade agreements. We use literature on designing agents for negotiation, cooperation, and coordination to identify three types of interaction-specific PSMs (Fig. 4):

- *Interaction Protocol PSM:* defines the rules for interaction engagement. An interaction protocol defines an order to engagements between agents using terms expressed by the communication protocol.
- *Model PSMs:* structured knowledge about how to model information that an agent observes. They directly relate to interactions because agency requires autonomous assessment of itself, external agents and the environment. An interaction protocol may constrain the types of information an agent may observe.
- *Strategy PSMs:* structured knowledge about how interactive behaviour is derived from output from the Model PSMs and the Interaction Protocol PSM.

The above three types are derived from classifications of agent design components used for interaction-dependent problem-solving (such as described in (Sandholm, 1999; Jennings, Faratin et al., 2001; Lomuscio et al., 2001)). For example, (Sandholm, 1999) describes variations of interaction protocols where particular strategies depend on models of utility for cooperative distributed problem-solving. (Lomuscio, Wooldridge et al., 2001) describes interaction protocols and strategies as the two basic types of components for agent-based negotiation. (Jennings, Faratin et al., 2001) describes areas of negotiation research concerned with protocols, negotiation objects, and decision making models.

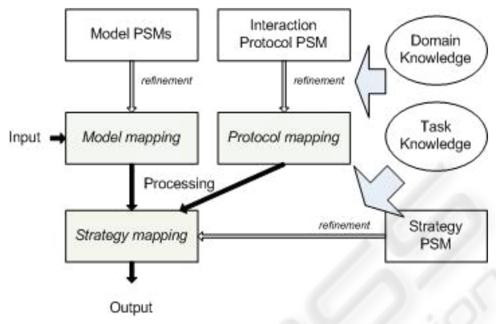

Figure 4: Knowledge level PSM composition.

Interaction Protocol PSM definitions are refined by domain and task knowledge to produce specific Interaction Protocol mappings (Figure 4). Model PSM definitions are refined by domain knowledge to produce Model PSM mappings, whereby inputs to these mappings are provided by the agent. Multiple Model PSMs may be selected or refined, depending on agent design. The output from the Interaction Protocol and Model PSM mappings are then used to select the Strategy PSM. The strategy PSM is then refined by task knowledge to produce the Strategy PSM mapping. The output of the Strategy PSM mapping is then used by the software engineer to design the agent's next interactive move. By distinguishing between Interaction Protocol PSMs, Model PSMs, and Strategy PSMs, interaction specific problem-solving knowledge may be reused by software engineers to design agent-oriented solutions to complex problems.

5 APPLICATION OF PSMS FOR MAS DEVELOPMENT

This section describes an application of interaction dependent PSMs and interaction-specific PSMs to designing agents for negotiation. The methodology follows from Section 2, Figure 2 (from (Beydoun, Tran et al., 2006)), where ontology-based development of MASs from PSMs was described. The scenario is negotiation for e-commerce trade.

Example: *An agent oriented system is required to automate negotiation in an electronic market place for buying and selling fish. Autonomous, self-interested agents act on behalf of people. Agents* *determine when and how to negotiate in order to satisfy the needs of people. Agents are responsible for collecting relevant information, and negotiating the best possible utility-based outcome given the information context* (Cuní et al., 2004).

Suppose a software engineer aims to design an agent that buys fish. At the conclusion of a domain ontology and goal analysis, the software engineer establishes a set of goals and task requirements to be satisfied by an agent. The engineer selects the task *"Buy fish"* and identifies *"Buy Item PSM"* as an appropriate possible solution. *"Buy Item PSM"* is identified as having an interaction dependency with another PSM, *"Sell Item PSM"*. Figure 5 illustrates a PSM approach to designing the agent solution.

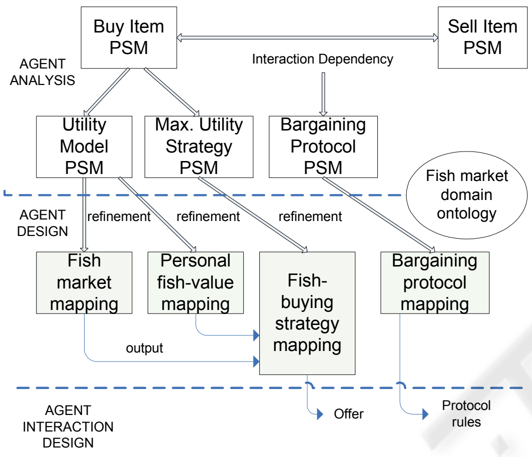

Figure 5: Agent-oriented modelling for fish market place derived from PSMs with an interaction dependency.

The software engineer determines that negotiation is the interaction type necessary for the domain. Since negotiation is pervasive in many domains, the engineer consults libraries for existing negotiation-specific problem-solving knowledge. Task and goal analysis revealed that agents are also required to maximise a utility, where a utility is defined by the domain ontology. Appropriate interaction-specific PSMs need to be selected – a type of Interaction Protocol, Model, and Strategy PSM. The software engineer attempts to find interaction-specific PSMs (within the repositories) oriented towards *utility* modelling and strategy. "Utility Modeling PSM" and "Maximise Utility Strategy PSM" are identified. For defining the interaction, a "Bargaining Protocol PSM" is suitable.

To complete the development, the software engineer now needs to design the fish buying agent for the market place. PSMs are *refined* by task and domain knowledge, resulting in *mappings* that are task and domain specific methods that can directly be used to design agent plans. Firstly, the domain ontology is used to refine the "Utility Model PSMs" to produce a *fish market mapping*, and a *personal fish-value mapping* (the inputs for these mappings might come from the person for whom the agent is acting). Secondly, refinement of the "Maximise Utility Strategy PSM" is made towards a specific communication protocol ontology, producing a *fish-buying strategy mapping*. The inputs for the *fish-buying strategy mapping* are the outputs from the *fish market mapping* and *personal fish-value mapping*. Thirdly, refinement of the "Bargaining Protocol PSM" is made towards the domain ontology to produce a *bargaining protocol mapping* which restricts interactions defined in terms of the communication protocol ontology. At the conclusion of this design, the software engineer may decide to engage in a similar process for designing the fish selling agent, with a view to (possibly) re-using PSMs and mappings identified for the buying fish agent. In addition to defining methods for interaction-dependent problem-solving, interaction-specific PSMs might also have dependencies with other PSMs. For instance, suppose the *Utility Model PSM* required information from external market agents – further analysis of interactions (albeit simple enquiries) may be necessary to design the agent to acquire this information.

Interaction dependencies between PSMs and interaction-specific PSMs drive the agent-oriented development of fish auction agents by using ontology-based domain, task, and problem-solving knowledge engineering where re-use and extensibility are supported.

6 CONCLUSIONS

The use of domain ontologies in AOSE has recently been investigated e.g. (Iglesias & Garijo, 2005). That work has been limited to the early phases of system development. It is our contention that the potentially knowledge intensive nature of the analysis involved towards creating the software components in a Multi Agent System suggests that a knowledge centred approach throughout the whole software development cycle is effective. This approach is underpinned by reusable knowledge models concomitant with an appropriate set of reusable domain problem solving processes (aka methods) that operationalise corresponding chunks of knowledge as required by the requirements of the system. (Beydoun, Tran et al., 2006) presented a methodological model underpinned by the presence of PSM repositories 'appropriately' represented. This paper bridges the gap between that work and the representation required to formulate the PSMs for interaction-dependent problem solving. We introduce new constructs to model the interaction dependencies of PSMs, and these are used by software engineers in the analysis of solutions to complex problems where interaction is required. We illustrated these constructs in a simplified development of a negotiation-based system.

Further work is required to create a formal underpinning of interaction-dependent PSMs We are in the process of developing a PSM library containing interaction-specific PSMs for supporting the development of negotiation agents in a variety of real-world domains. Future work will also identify and integrate software process steps required within an agent-oriented methodology.

ACKNOWLEDGMENTS

This work is supported by the Australian Research Council.

REFERENCES

Beydoun G., Krishna A.K., Ghose A. & Low G.C. 2007. Towards Ontology-Based MAS Methodologies: Ontology Based Early Requirements *Information Systems Development Conference*. Galway.

Beydoun G., Tran N., Low G. & Henderson-Sellers B. 2006. Foundations of Ontology-Based Methodologies for Multi-agent Systems: Springer LNCS, pp. 111-123.

Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J. & Perini A. 2004. TROPOS:An Agent Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8, 203-236.

Breuker J. 1999. Indexing Problem Solving Methods for Reuse. *Knowledge Acquisition, Modeling and Management*, 1621.

Bruaux S., Kassel G. & Morel G. 2005. An ontological approach to the construction of problem-solving models. *K-CAP '05: Proceedings of the 3rd international conference on Knowledge capture*.

Cattoni R., Potrich A., Charlton P. & Mamdani E. 1999. Evaluating the FIPA standards on the field: an audio video entertainment applications. *First Asia-Pacific Conference on Intelligent Agent Technology*.

Cuní G., Esteva M., Garcia P., Puertas E., Sierra C., et al. 2004. MASFIT: Multi-Agent System for Fish Trading. *Proceedings of the 16th European Conference on Artificial Intelligence*.

Decker S., Erdmann M., Fensel D. & Studer R. 1999. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. *DS*, 8, 351-369.

Deloach S.A., Oyenan W.H. & Matson E.T. 2008. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16, 13-56.

Fatima S.S., Wooldridge M. & Jennings N. 2004. Optimal negotiation of multiple issues in incomplete information settings *AAMAS04*. New York: ACM.

Fensel D., Motta E., Benjamins V.R., Crubezy M., Decker S., et al. 2002. The Unified Problem-solving Method Development Language UPML. *Knowledge and Information Systems*, 5, 83-131.

Fensel D., Motta E., Decker S. & Zdrahal Z. 1997. Using Ontologies for Defining Tasks, Problem-Solving Methods and Their Mappings. *In:* E. Plaza & R. Benjamin (eds.) *10th European Knowledge Acquisition Workshop (EKAW97)*. Spain: Springer, pp. 113-128.

Gómez M. & Plaza E. 2007. The ORCAS e-Institution: a Platform to Develop Open, Reusable and Configurable Multi-Agent Systems. *International Journal of Intelligent Control and Systems*, 12, 130-141.

Iglesias C.A. & Garijo M. 2005. The Agent-Oriented Methodology MAS-CommonKADS. *In:* B. Henderson-Sellers & P. Giorgini (eds.) *Agent-Oriented Methodologies*: IDEA Group Publishing, p. 46.78.

Jennings N., Faratin P., Lomuscio A., Parsons S., Sierra C., et al. 2001. Automated negotiation: prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10, 199-215.

Lomuscio A., Wooldridge M. & Jennings N. 2001. A Classification Scheme for Negotiation in Electronic Commerce. *Agent Mediated Electronic Commerce, The European AgentLink Perspective*, 19-33.

Müller J.-P. 2002. Emergence of Collective Behaviour and Problem Solving. *Engineering Societies in the Agents World*, 1-21.

Pavon J., Gomez-Sanz J. & Fuentest R. 2005. The INGENIAS Methodology and Tools. *In:* B. Henderson-Sellers & P. Giorgini (eds.) *Agent-Oriented Methodologies*: IDEA Group Publishing, pp. 236-276.

Sandholm T. 1999. Distributed rational decision making. *Multiagent systems: a modern approach to distributed artificial intelligence*, 201 - 258.

Schreiber G., Akkermans H., Anjewierden A., Hoog R.d., Shadbolt N., et al. 2001. *Knowledge Engineering And Management: The CommonKADS Methodology*. London: The MIT Press.

Studer R., Benjamins V.R. & Fensel D. 1998. Knowledge engineering: principles and methods. *Data and Knowledge Engineering*, 25, 161-197.

Tran Q.-N.N. & Low G. 2008. MOBMAS: A Methodology for Ontology-Based Multi-Agent Systems Development. *Information and Software Technology*, 50, 697-722.