Extending Prometheus with Agent Cooperation

Ebrahim AlHashel, Bala M. Balachandran, and Dharmendra Sharma

Faculty of Information Sciences and Engineering
University of Canberra - Australia

{ebrahim.al.hashel, bala.balachandran, dharmendra.sharma}@canberra.edu.au

Abstract

Agent cooperation is a central component of many multi-agent systems. To be successful in open, multi-agent environments, agents must be capable of dynamically forming a team and solving a problem by cooperating with each other. Such an autonomous agent cooperation process has the potential to execute a plan to achieve a common goal. To date numerous methodologies for agent-oriented software engineering (AOSE) have been proposed in the literature. However, those methodologies lack the ability to model agent cooperation strategies effectively.

This research work proposes a set of artefacts to extend Prometheus development methodology in order to support agent cooperation. We will illustrate the extended Prometheus methodology using an example from the domain of e-commerce. The results are presented and our future work is discussed

Keywords: Multi-agent systems, agent-oriented software engineering, software autonomous agents, software agent cooperation

1. Introduction

A multi-agent system is designed to be capable of reaching goals that are difficult to achieve by an individual agent or a monolithic system. In multi-agent systems, an agent usually cooperates with other agents, so it should have some social and communicative abilities. Designing a team of agents that can cooperate and work together toward a common goal is one of the challenges in multi-agent systems research. There are many theories and attempts directed to formulate the team formation process or what is more commonly known as agent cooperation [1], [2], [3], [4], [5], [6], [7]. The existing agent-oriented software engineering methodologies, including Prometheus [8], Gaia [9], MaSE [10], PASSI [11], Tropos [12], are not directed to support the engineering of multi-agent systems that are capable of dynamically forming a team and solving a problem by cooperating with each other. Further, due to the growing interest in the development of multi-agent systems, many software development environment and software runtime platforms were created, for example, JADE [13], Zeus [14], RETSINA [15], IMPACT [16], JACK [17], and Aglet [18]. Currently available tools are designed to support variant agent services and approaches except the team formation concepts. The lack of ability of the agents to cooperate within a multi-agent environment means such systems cannot function effectively to their full potential.

We have examined the strengths and weaknesses of the Prometheus methodology for designing a multi-agent travel support system where autonomous software agents, cooperate to satisfy a customer's needs in terms of hotel, flight, and rental car. The Prometheus development methodology provides mechanisms for analysing and

978-0-7695-3514-2/08 $25.00 © 2008 IEEE
DOI 10.1109/CIMCA.2008.222

designing Belief-Desire-Intention (BDI) agents. Our experience with the Prometheus methodology has shown that it is an excellent initiation in the field of AOSE, but still needs further improvements in order to be used for real-world complex applications where agent cooperation is essential [19]. In this paper we propose a theoretical framework that extends the Prometheus development methodology to incorporate agent organisation and team formation concepts.

The proposed cooperation concept has been developed based on the notion that an agent controls a particular professional domain which embodies a set of skill agents, as well as sub-skills, depending on the system requirements. This cooperation concept deploys a four-level model: user, coordination, professional, and skill levels.

The extending process has been carried out with consideration to the Prometheus analysis structure as a descriptive methodology. For this reason we have designed a new set of additional descriptive forms and tables to extend Prometheus's three phases as follows: a) the architecture design phase is extended by the user-goal diagram and user-goal descriptor; b) the system specification phase is extended by professional agent registry and the execution plan; c) the detailed design phase is extended by the skill agent registry and its execution plan. The implementation phase remains unchanged and retained for future work to design and develop a software cooperation runtime platform.

The remainder of this paper is structured as follows. Section 2 overviews the Prometheus methodology. Section 3 describes our agent cooperation model and explains how our model can be integrated into the current Prometheus framework. Section 4 illustrates the proposed model using a practical example. Section 5 summarises our conclusions and future work.

2. Prometheus Methodology Overview

Prometheus [8], consists of three phases, as shown in Figure 1: system specification, architectural design, and detailed design. The first phase, system specification, deals with determining the system's environment and establishing its goals and functionalities. The environment is defined in terms of percept and actions while the system's functionality is identified in terms of goals and plans. The outcome of this phase are system goals, scenario and functionality descriptions. The second phase, architectural design, uses and processes the system specification artefacts to develop a high–level designed agent system. The outcomes of this phase are agent types that will be used by the application, the interactions between agents, and overall system structure. The third phase, detailed design, will produce three main artefacts: agent capabilities, plans, and events. The implementation process is kept open for the developer to choose a preferred platform but the use of the JACK development environment is strongly recommended because both Prometheus and JACK are based on BDI architecture. Prometheus development tool PDT provides an option to the developer to automatically generate JACK skeleton java code.

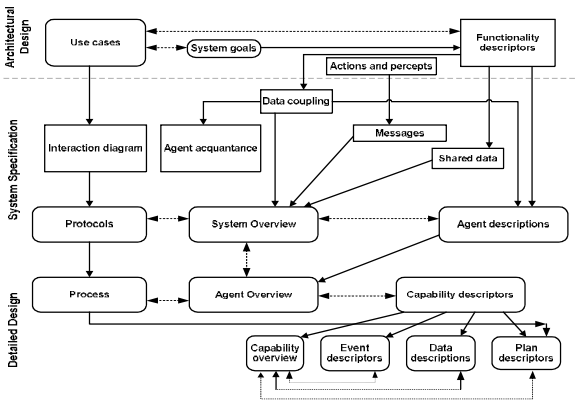

Figure 1: Prometheus methodology architecture

3. A Conceptual Model for Agent Cooperation

We believe that the agent cooperation process must be facilitated by a multi-agent systems software runtime platform (Agent Cooperation Model) to manage and support the team formation process in terms of agent coordination, communication, and execution plan. Figure 2 shows the relationship between the enhanced Prometheus methodology and the proposed agent cooperation model.

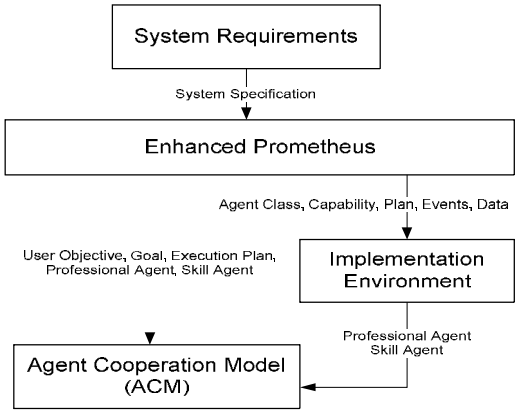

Figure 2: Project overview diagram

The proposed cooperation concept is based on profession and skill types that are possessed by agents. Each agent is characterised by particular profession domain and each profession domain own a set of skill agents. Each skill agent has its own unique task, which collectively forms a particular profession.

The cooperation concept can be classified into four levels, namely user level, coordination level, professional level, and skill level, as shown in Figure 3. The components of the proposed cooperation model are correlated to each other but it is not necessary to implement the idea as one body, so we can treat each level as an independent plug-in component. Therefore, the user level will be allocated in the requirement phase. The coordination level will be merged with the specification phase while both the professional and skill level will be incorporated in the detailed design phase.

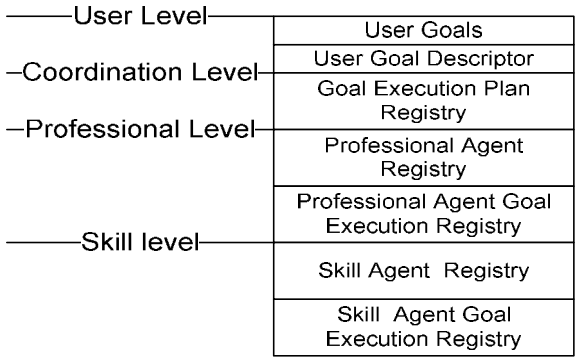

Figure 3: Agent cooperation levels

4. A Case Study: A Travel Agency System

Over several years, travel agency systems have been used to illustrate various aspects of agent technology for e-commerce [10]. A Travel Agency System (TAS) is a multi-agent system designed to obtain optimum travel packages including hotel, flight, and rental car for the customer, depending on their preferences. Figure 4 shows the multi-agent architecture of the travel agency system [20].

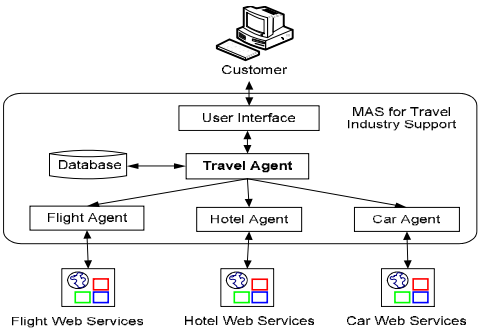

Figure 4: TAS system architecture

Below we illustrate our agent cooperation model using the above case study.

4.1. User Level

The scenario approach is an effective method to handle the user-goal requirement and can be used in a straightforward manner. The first main scenario will be retrieved from the system specification document by asking what the user wants the system to do on his/her behalf. After developing the first level scenario, the second refinement starts by finding what happens if the first level fails. This process continues until the exact user goal requirements are captured. The idea is like reversing the use-cases model in UML. Instead of identifying the actor to system functionality, it identifies the system to user requirements. The following steps represent the approach:

1. Read system specification document and describe the typical interaction between the user and the system.
2. Develop the high level scenario.
3. Refine the high-level scenario to capture the user's goal.
4. Use the user-goal diagram (see example in Figure 5) to represent the user goals.
5. Develop the goal descriptor (see Figure 6) for each user goal.

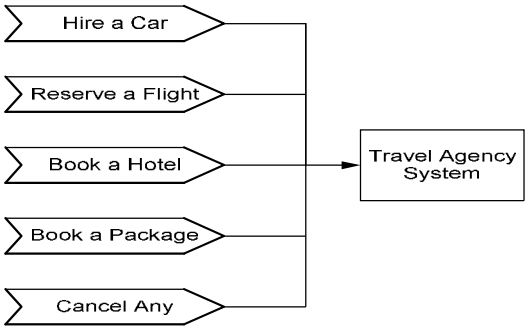

Figure 5: The user-goal diagram

User-Goal Descriptor

User-Goal Name: Hire Car
Description: the user can activate this goal to trigger the system to hire a car according to the user selection.
Verification: valid payment method, valid driving licence
Input Data: Hiring date, period, credit card no, licence validity date, personal data, arrival time
Option1: Booking can be cancelled one day prior to the trip
Action1: Payment refunds available within five days
Option 2: Car not available
Action2: Cancel the request, notify the user
Priority: 1

Figure 6: Example of a user-goal descriptor

The main difference between percept and user-goal is that the percept is linked to a particular action or event in the system, whereas the user-goal can be linked to many precepts. In fact, the user-goal is at a higher level than the percept.

4.2. The Coordination Level

The coordination levels is a logical component in the cooperation platform, but at this stage of detailed design, we would like to define the elements which will drive the process, for example, professional agent capabilities, goal execution, and protocols. Figure 7 depicts the goal execution plan descriptor that provides information to the professional level agent's capabilities and responsibilities. The codes in the goal execution plan descriptor are related to the main descriptors of each activity where the name of each activity is presented to maintain Prometheus analysis structure.

Goal Execution Plan
Goal Id: GOL-23-01
Goal Name: Rent car
Goal Description: Rent a car according to the customer selection
Professional Agent Id: PAG-01-01
Professional Agent Name: Car Rental

Skill Agent (SAG)	Car agent	Flight agent	Hotel agent
Skill Agent Id	SAG-23-05		
SAG Goal Id	SGL-23-12		
SAG Sub-Goal Id	SSG-23-181		
SAG Plan Id	SPL-23-03		

Figure 7: Goal execution plan descriptor

4.3. The Professional Level

The professional level consists of high-level agents that own the strategy of executing particular tasks using a set of skills agents. Each professional agent is loaded with the one execution plan corresponding to its objective, including its relevant skill agent's team, capabilities, and communication protocols. The agent designed by Prometheus will be linked to this agent level and contains additional information related to the logical professional level platform. Figures 8 and 9 depict the two independent descriptors that will be developed in the detailed design phase, in sequence with capabilities, plans, and events.

Professional Agent
Professional Id : PAG-01-01
Profession Name : Car Rental
Description : Rent a car from different service provider

Goal Id	**Goal Name**
GOL-23-01	Rent car

Figure 8: Professional agent registry

Professional Agent Goal Execution Plan
Goal Id:GOL-23-01
Goal Name: Rent car
Goal Description: Rent a car according to the customer selection

Execution Sequence	**Plan Id**	**Plan Name**	**Skill Agent Id**	**Skill Agent Name**
1	CRP-01	Rent Car P-01	SAG-23-05	Silver Car Rental
2	CRP-02	Rent Car P-02	SAG-23-06	Black Car Rental

Figure 9: Professional agent goal execution plans registry

4.4. The Skill Level

The skill level is the factory that the system depends on. At this level all the skill agents are located in a tree configuration, as described earlier. Any system must contain a skill agent regardless how small it is, but a sub-skill level does not necessarily exist. This depends on the size and the design of the application. The agents that design using Prometheus development methodology can be directly attached to the skill level along with its enhancement descriptor. The additional descriptor elements will enable the skill agent to coordinate and communicate with the professional level and with each other at the same time. Figures 10 and 11 show how the current agent descriptors have been redesigned.

Skill Agent
Skill Agent Id: SAG-23-05
Skill Agent Name: Silver Car Rental
Description: Rent a car according to customer needs

Goal Id	**Goal Name**
GOL-23-01	Rent car

Figure 10: Skill agent registry

Skill Agent Goal Execution Plan

Goal Id:GOL-23-01
Goal Name: Car Rental
Goal Description: Rent a car from Silver Car Rental

Execution Sequence	Plan Id	Plan Name	Skill Agent Id	Skill Agent Name
1	SPL-23-03	Rent Car	SAG-23-05	Silver Car Rental
2				

Figure 11: Skill agent goal execution plan registry

4.5. The Sub-Skill Level

The sub-skill level will emerge if the system designer finds it suitable for the system. It leads to an efficient design, however, efficiency depends on the nature and the size of the system. If there is any need to extend to this stage, then the relationship between the skill level and sub-skill level takes the same structure as that of the professional to skill levels. Regardless of the number of nodes in the skill trees, the agent in the node acquires the professional agent role and the descriptors will also flow in parent–child structure.

5. Summary and Future Work

Agent cooperation is a central component of many multi-agent systems. We have indicated that the major weakness of the Prometheus methodology is its lack of ability to support agent cooperation. We have presented and illustrated a theoretical framework for an agent cooperation model, one which supports agent cooperation at four different levels, namely user level, coordination level, professional level, and skill level. The team formation process proposed in this paper is designed to match the goal so that the appropriate composition of the professions and skills of the agent team is achieved. This research paper provides the first step in the multi-agent cooperation process. Step two will be developed in future work which will focus on designing and developing a software cooperation model that can operate as a runtime environment for agent cooperation. The results will be reported in our future publications.

References

[1] P. Cohen, H. Levesque, and I. Smith, "On Team Formation", in Contemporary Action Theory, Synthesis, J. Hinitkka and R. Tuomela, Eds., 1997.

[2] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman, "On Cooperation in Multi-Agent Systems", presented at Foundations of Multi-Agent Systems, University of Warwick UK, 1996.

[3] M. Luck and M. d'Inverno, "Engagement and Cooperation in Motivated Agent Modelling", presented at Distributed Artificial Intelligence Architecture and Modelling, Australia, 1996.

[4] L. Changhong, L. Minqiang, and K. Jisong, "Cooperation Structure of Multi-agent and Algorithms", presented at 2002 IEEE International Conference on Artificial Intelligence Systems, China, 2002.

[5] K. P. Sycara and G. Sukthankar, "Literature Review of Teamwork Models", Robotics Institute, Carnegie Mellon University 31, Pittsburgh, Pennsylvania, Technical CMU-RI-TR-06-50, November 2006.

[6] M. Wooldridge and N. R. Jennings, "Towards a Theory of Cooperative Problem Solving", presented at 6th. European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-94), Odense, Denmark, 1994.

[7] B. Wilsker, "A Study of Multi-Agent Collaboration Theories", University of Southern California 24, Pasadena, Research ISI/RR-96-449, November, 1996.

[8] L. Padgham and W. Michael, Developing Intelligent Agent Systems, vol. 1. England: John Wiley & Sons, Ltd 23 82, 2004.

[9] M. Wooldridge, N. Jennings, and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design", presented at Autonomous Agents and Multi-Agent Systems, Netherland, 2000.

[10] M. Fasli, "*Agent Technology for e-Commerce*", vol. 1, 1 ed: John Wiley 453, 2007.

[11] C. P. M. Cossentino, "PASSI: a Process for Specifying and Implementing Multi-Agent Systems Using UML". Roam-Italy: Engineering Ingegneria Informatica S.P.A, 2001.

[12] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, "Tropos: An Agent-Oriented Software Development Methodology", presented at Autonomous Agents and Multi-Agent Systems, Netherlands, 2004.

[13] F. Bellifemine, G. Caire, and D. Greenwood, Developing Multi-agent Systems with JADE. West Sussex England: Wiley, 2007.

[14] Zeuse, development toolkit, http://labs.bt.com/projects/agents/zeus/, 12/09/2008

[15] RETSINA, development toolkit, http://www.ri.cmu.edu/projects/project_76.html, 24/06/2008

[16] IMPACT, development toolkit, http://www.cs.umd.edu/projects/impact/Docs/IMPACT_Manual_Ch2.pdf, 16/07/2008

[17] JACK, development toolkit, http://www.agent-software.com/, 07/07/2008

[18] Aglet, development toolkit, http://aglets.sourceforge.net/, 23/07/2008

[19] E. AlHashel, B. Balachandren, and D. Sharma, "Comparison of Three Agent-Oriented Software Development Methodologies: ROADMAP, Prometheus, and MaSE", in KES2007. ITALY: Springer, 2007.