Chapter V

Hermes: Designing Flexible and Robust Agent Interactions

Christopher Cheong¹
RMIT University, Melbourne, Australia

Michael Winikoff
RMIT University, Melbourne, Australia
University of Otago, Dunedin, New Zealand

ABSTRACT

Although intelligent agents individually exhibit a number of characteristics, including social ability, flexibility, and robustness, which make them suitable to operate in complex, dynamic, and error-prone environments, these characteristics are not exhibited in multi-agent interactions. For instance, agent interactions are often not flexible or robust. This is due to the traditional message-centric design processes, notations, and methodologies currently used. To address this issue, we have developed Hermes, a goal-oriented design methodology for agent interactions which is aimed at being pragmatic for practicing software engineers. Hermes focuses on interaction goals, i.e., goals of the interaction which the agents are attempting to achieve, and results in interactions that are more flexible and robust than message-centric approaches. In this chapter, we present the design and implementation aspects of Hermes. This includes an explanation of the Hermes design processes, notations, and design artifacts, along with a detailed description of the implementation process which provides a mapping of design artifacts to goal-plan agent platforms, such as Jadex.

INTRODUCTION

Our ever-evolving and technologically advanced world is a place that is complex, dynamic, and failure-prone. Intelligent agents are steadily accruing purchase as a technology which is intrinsically able to address the aforementioned real world issues (Jennings, 2001). Currently, intelligent agents are used in
a range of real world applications spanning a number of different domains. These include telecommunication systems (Chaib-draa, 1995; Jennings 2001), process control (Sycara 1998; Jennings et al. 1998), air traffic control (Sycara, 1998), business process management (Jennings, 2001), logistics (Benfield et al., 2006), production scheduling (Munroe et al., 2006), and many more.

A key issue in developing and using agents is how to systematically analyse and design multi-agent systems. This issue has resulted in the development of the field of Agent Oriented Software Engineering. This field has seen the development of a number of methodologies which provide the developer with guidance, processes, and notations for the analysis and design of agent systems.

The systems in the previous examples all employ multiple agents as “there is no such thing as a single agent system” (Wooldridge, 2002). In such multi-agent systems, agent interactions are the crux of the matter, as the agents will need to interact in various ways in order to achieve their goals. Consequently, the design of agent interactions is a crucial part of a design methodology.

Current approaches to interaction design are message-centric as the design process is driven by messages that are exchanged during the interaction and is focused on the information passed within the messages. For example, in the Prometheus methodology (Padgham and Winikoff, 2004), as part of its interaction design process, the designers are advised to think about messages and alternatives. This is not restricted to Prometheus, but is also the norm in other methodologies such as Gaia (Zambonelli et al., 2004), MaSE (DeLoach et al., 2001) and Tropos (Bresciani et al., 2004).

Using current message-centric approaches to create interactions results in a number of problems. The main problem is that designs resulting from message-centric approaches tend to be overly, and sometimes unnecessarily, constrained. For example, using the interaction protocol of Figure 1, the interaction must begin with the Customer agent enquiring about the price of a laptop. It cannot, for example, enquire about the availability of a laptop first. Similarly, if a laptop is out of stock, the Vendor cannot proactively send a “Laptop Out of Stock” message to the Customer agent before or after replying with the price.

This lack of flexibility and robustness in interactions is problematic for intelligent agents. By following such limited designs, key intelligent agent characteristics, such as autonomy and proactivity, are greatly subdued and the fundamental concept of goal-orientation is ignored. Thus, current approaches to interaction design are not congruent with the agent paradigm.

More abstractly, the problem with message-centric approaches results from the general design process where the designer begins by creating a desirable but rigid message sequence and then “loosens it”, i.e. improves flexibility and robustness by adding alternatives. The problem with this is that the “default” result is an interaction that has not been sufficiently “loosened”, and is more constrained than it needs to be. A number of alternative approaches for specifying agent interactions have been explored. These alternative approaches avoid overly restricting interactions by starting with completely unconstrained interactions and then adding constraints so that the protocols are restricted and lead only to desirable interactions.

In this chapter we describe one such alternative approach: the Hermes methodology. Hermes provides a way of designing agent interactions that results in flexible and robust interactions. A key design aim of Hermes was that it had to be pragmatic, that is, easily usable by designers. In particular, we wanted Hermes to include processes, notations, and guidance, not just a means for representing interactions. Furthermore, Hermes had to be usable by software engineers, not just agent researchers.

Hermes is intentionally limited to the design and implementation of agent interactions and not of entire agent systems as there are already many good methodologies for such in existence. Instead