Multi-agent architectures as organizational structures

Manuel Kolp · Paolo Giorgini · John Mylopoulos

Abstract A Multi-Agent System (hereafter MAS) is an organization of coordinated autonomous agents that interact in order to achieve common goals. Considering real world organizations as a metaphor, this paper proposes architectural styles for MAS which adopt concepts from organizational theories. The styles are modeled in i*/Tropos, using the notions of actor, goal and actor dependency and are intended to capture needs/wants, delegations and obligations. The proposed architectural styles are evaluated with respect to a set of software quality attributes, such as predictability and adaptability. In addition, we report on a comparative study of organizational and conventional software architectures using a mobile robot control example from the Software Engineering literature. The research reported here was conducted within the scope of the Tropos project, whose objective is to develop a comprehensive agent-oriented software development methodology.

Keywords Multi-agent architectures · Organizational styles · Architectural design

1. Introduction

Software architectures describe a software system at a macroscopic level in terms of a manageable number of subsystems, components and modules inter-related through data and control dependencies.
System architectural design has been the focus of considerable research during the last fifteen years and has produced well-established architectural styles and frameworks for evaluating their effectiveness with respect to particular software qualities. Examples of styles are pipes-and-filters, event-based, layered, control loops and the like [52]. Examples of software qualities include maintainability, modifiability, portability and the like [1]. We are interested in developing a suitable set of architectural styles for multi-agent software systems.

In this paper, we consider that the fundamental concepts of a Multi-Agent System (MAS) are intentional and social, rather than implementation-oriented. We turn to theories which study social structures for motivation and insights since after all, software agents by virtue of their intentional capabilities (e.g., ability to plan and negotiate) are much more than mere software components. Moreover, MAS attempt to emulate or are inspired by our limited understanding of how humans plan and negotiate. In addition, it is precisely these capabilities that social structures take advantage of to enhance flexibility, extensibility, robustness and reliability of the overall system. Finally, social structures have been used in practice for much longer than software architectures. Adopting those among them that have proven most useful makes sense in any attempt to revise our conceptualization of software into one where the basic components are endowed with intentional and social traits.

But, what kind of social theory should we turn to? There are theories that study group psychology, communities (virtual or otherwise) and social networks. Such theories study social structure as an emergent property of a social context. Instead, we are interested in social structures that result from a design process. For this, we turn for guidance to organizational theories, namely *Organization Theory* that describe the internal structure and design of an organization and *Strategic Alliances* that model the strategic cooperation of independent organizational actors that pursue shared goals.

In this paper, we propose a set of architectural styles for MAS, and give an in-depth account of two of them – the structure-in-5 and the joint venture – for multi-agent using the strategic dependency model of $i^*$ [63]. We then analyze these styles with respect to a number of qualities, such as predictability and adaptability. In order to compare our proposal with conventional architectural styles, we also report on the results of a comparative study of organizational and conventional software architectures using a mobile robot control example from the Software Engineering literature.

This research is being conducted within the context of *Tropos* [4,48], a project developing a requirements-driven methodology for software systems. The *Tropos* methodology adopts and integrates ideas from MAS and Requirements Engineering, where agents/actors and goals have been used heavily for early requirements analysis [9,63]. In particular, the *Tropos* project adopts Eric Yu’s $i^*$ model which offers actors (agents, roles, or positions), goals, and actor dependencies as primitive concepts for modeling an application during early requirements analysis.

The key assumption which distinguishes *Tropos* from other software development methodologies is that actors and goals can be used as fundamental concepts during all phases of software development, not just requirements analysis. Using the same concepts and notations to match requirements analysis with system architecture reduces the semantic gap between the organizational environment and the architecture of the future system. It also leads to a streamlined development process founded on a small set of concepts. More details about Tropos can be found in [4].

The present work integrates and extends research in progress about social abstractions for the Tropos methodology. In [35], we have detailed a social ontology for Tropos that treats information systems as social structures throughout the development life cycle. In [36], we