Developing and Evaluating a Generic Metamodel for MAS Work Products

Ghassan Beydoun\(^{1,2}\), César Gonzalez-Perez\(^{1,2}\), Brian Henderson-Sellers\(^{2}\), and G. Low\(^{1}\)

\(^{1}\) School of Information Systems, Technology and Management, University of New South Wales, Sydney, Australia (g.beydoun, g.low)@unsw.edu.au
\(^{2}\) Faculty of Information Technology, University of Technology of Sydney, Sydney, Australia (brian, cesargon)@it.uts.edu.au

Abstract. MAS development requires an appropriate methodology. Rather than seek a single, ideal methodology, we investigate the applicability of method engineering, which focuses on project-specific methodology construction from existing method fragments and provides an appealing approach to organize, appropriately access and effectively harness the software engineering knowledge of MAS methodologies. In this context, we introduce a generic metamodel to serve as a representational infrastructure to unify the work product component of MAS methodologies. The resultant metamodel does not focus on any class of MAS, nor does it impose any restrictions on the format of the system requirements; rather, it is an abstraction of how the work product elements in any MAS are structured and behave both at design time and run-time. Furthermore, in this paper we validate this representational infrastructure by analysing two well-known existing MAS metamodels. We sketch how they can be seen as subtypes of our generic metamodel, providing early evidence to support the use of our metamodel towards the construction of situated MAS methodologies.

1 Introduction

There is an increasing software engineering interest in the use of multi-agent systems (MAS), a new class of distributed parallel software applications, that have already proved effective in the core tasks of automating management of information within businesses (e.g. computer network management applications [23]), building computational models of human societies to study emergent behaviour [14, 17, 34] and building cooperative distributed problem solving [22, 24]. The building blocks of a MAS are intelligent, autonomous and situated software entities: agents. The agent, the concept of agency and the full range of MAS abstractions offer the promise of making software systems easier to embed within our daily lives as suggested in [10].

In order to develop a MAS, some appropriate methodological approach is needed. Indeed, a significant number of such MAS methodologies already exist [20]. Notable examples are Gaia [36], Adelfe [2], Prometheus [28], PASSI [8]. However, since it is generally agreed [7, 18] that no single methodology is sufficient, regardless as to how well thought out it might be, any one of these individual methodologies will, by definition, have limited applicability e.g. to a specific domain or a specific type of software application.
We argue in this paper that attempting to simply combine existing methodologies into one large, high quality methodology, as suggested in e.g. [1] will prove to be impossible, because the sets of assumptions underlying each methodology are likely to be inconsistent and irreconcilable. We propose instead using method\(^1\) engineering [5, 26] to empower software developers to create methodologies from existing (method) fragments (i.e. self-contained components). Method engineering approaches have been successful in object-oriented development due to widely accepted modelling languages and constructs of OO software systems and development processes [19, 21, 30]. For method engineering to be equally successful in the context of MASs, a suitable representation of any potential agent-oriented methodology is required. The goal of such representation is to capture knowledge about methodologies. This includes concepts (plus their properties) related to products of the software development process, as well as concepts and their properties related to the software development process itself. These collections of concepts are often known respectively as product metamodel and process metamodel [30].

In this paper, we present a generic product metamodel\(^2\) for any MAS methodology. In this context, “product metamodel” is synonymous with “modelling language specification”. Our generic metamodel comprises the abstract syntax and semantics of such a modelling language. It does not make any assumptions about the kinds of MAS that it describes. It only makes assumptions about what are the essential properties of an agent. Our metamodel is the first to focus on conceptual and ontological underpinnings rather than being constrained for use in a single methodological approach. Moreover, in this paper, we reinforce our case for method engineering in the context of MAS development by validating our generic metamodel against two well known and applied MAS metamodels: TAO [32] and Islander\(^3\) [13]. We sketch how our metamodel can generate both of them. This constitutes early evidence that our method engineering proposal for MAS development is plausible.

The rest of this paper is organised as follows: In Section 2, we justify our method engineering endeavour and describe our metamodel and its synthesis. In Section 3, we present a comparative analysis of this metamodel and two prominent (although not explicit) metamodels: those of Islander [13] and TAO [32]. We indicate that these two metamodels can be viewed as particular refinements of our metamodel. In Section 4, we conclude with a description of future work.

## 2 Generic MAS Metamodelling

Edmonds et al. [10] rightly point out that we do not currently know all possible features of any MAS. They compare the science of MAS to the science of zoology, where we have a lot to discover about how the internals of a MAS change dynamically, and how this alters the overall behaviour of the system. They use this to formulate a theoretical argument against the possibility of having a one-size-fits-all methodology. In this section, we point to other current limitations in pursuing an

---

1 Both terms, method and methodology are considered synonymous in this paper.
2 Henceforth, we use the term “metamodel” and “product metamodel” interchangeably.
3 Islander is a specification language. We compare our metamodel to its underlying model.