Neurocomputing 182 (2016) 304–321

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Business and model-driven development of BDI multi-agent systems

Yves Wautelet [a,*], Manuel Kolp [b]

[a] KU Leuven, Belgium
[b] Université catholique de Louvain, Belgium

ARTICLE INFO

Article history:
Received 19 February 2015
Received in revised form
29 August 2015
Accepted 6 December 2015
Communicated by Ngoc Thanh Nguyen
Available online 21 December 2015

Keywords:
Service modeling
BDI software system
I*
Actor responsibility assignment
Multi-agent system
Model-driven engineering
Agent-based development

ABSTRACT

Model-driven development allows IT professionals to specify the system functionality, organization and behavior in a logical or platform-independent manner. Modeling using services allows domain analysts to focus on the added-value and core business the enterprise offers to its stakeholders. Those services are coarse-grained elements able to encapsulate a composition of business process models. The framework presented in this paper provides models together at strategic, tactical and operational levels to develop an agent-oriented software system. The strategic level is concerned with long-term decisions; this top-level uses a service model to understand the business' high-level (added) values as well as the Quality Expectations and the threats they face. The tactical level is concerned with a broader description of the business processes automated by the system; the i* strategic dependency and rationale models are used here to further document the service behavior. Actors' accountability and responsibility can be determined in the visual representation of these strategic and tactical levels. Finally, i* models are mapped into a set of operational models to document the (multi-agent) system behavior when achieving modeled functionalities. These operational models instantiate the Belief/Desire/Intentions (BDI) paradigm proposing entities – the agents – mapping as closely as possible the real life organization. The paper thus builds a business-driven transformation process leading to a run-time agent-architecture in a single and common framework. It both uses existing models and introduces or refines existing ones to dispose of a method ensuring better alignment and traceability between the business and the IT system.

1. Introduction

1.1. Research context

Many modern software development methodologies are said to be *Model-Driven* in the sense that the whole development process can be traced from or driven by high-level modeled entities. For instance, object-oriented development methodologies such as the *Unified Process* inspired ones (*RUP, OpenUP, EUP, AUP, …* [1–4]) are said to be *use case driven*, meaning that the entire process is driven by the system functionality and behavior identified as use cases at the requirement analysis and/or business modeling stage. Similarly, implementation methodologies for ERP and e-business systems [5] such as *Accelerated SAP* (*ASAP*), *Fast Track*, *Business Integrated Methodology* (*BIM*) or *Sure Steps* or in some case even *PRINCE 2* [6] may be considered business processes driven in the sense that the life cycle is, in this case, driven by business functionality and activity identified during the business (process) modeling step.

Besides, in model-driven development, highest level analysis elements are called *scope elements* and are consequently useful not only to share a common high-level vision with stakeholders, but also to estimate the project effort on a non-redundant basis, for evaluating related risks and opportunities brought by software adoption, etc.

Defining adequate scope elements is a key factor for a successful adoption of a software development methodology. Indeed, such element granularity must be adequate and the focus on a core functionality of the application is crucial to determine one particular aspect of the software to build. Agent-oriented development methodologies such as Tropos [7–11] have proposed various concepts to represent and develop software systems; some are coarse-grained (e.g. goals, tasks) and other fine-grained (e.g. beliefs, desires, intentions). Nevertheless, agent and requirement-driven methodologies such as Tropos still lack to adopt a clear "red-threat" from the strategic to the operational levels with a direct impact on scalability (see [12]).

1.2. Contributions

This paper is an effort to propose a clear model-driven framework to develop agent-oriented software proposing strategic, tactical and operational views. To this extent, it addresses the lacks and deficiencies of classical i* (i-star, see [13–16]) models to furnish adequate scope elements. For this purpose, it refines a proposal

* Corresponding author.
E-mail address: yves.wautelet@kuleuven.be (Y. Wautelet).

http://dx.doi.org/10.1016/j.neucom.2015.12.022
0925-2312/

from [12] to define a strategic analysis model driven by the concept of *Service*. It genuinely introduces elements of quality and risk management at this strategic level and formalizes the proposal. It also proposes to study the actors' responsibility assignment within the strategic and tactical perspectives. Classical i* models are not left out but used for tactical knowledge representation. Finally, the Multi-Agent System (MAS) design – constituting the operational perspective – is represented through three different models introduced in the paper. These are aimed to implement agent software with the cognitive *Belief*, *Desires*, *Intentions* (*BDI*) paradigm in mind (see [17–20]), a simple but efficient reasoning model that allows us to capture human rationale. We thus address the representation of the system-to-be. The implementation of the operational models is nevertheless outside the scope of this paper, but an implementation model for the proposed MAS design has been covered in previous work (see [21]).

In short, the paper formalizes a strategic model for knowledge representation as well as design diagrams following the BDI paradigm; the tactical middle layer constituted by the i* strategic dependency and strategic rationale diagrams is left as-is. This gives a business and model-driven perspective for Tropos allowing full traceability of elements from strategic to operational levels.

1.3. Paper structure

The paper is organized as follows: Section 2 overviews related work and positions the present proposal. Section 3 motivates the need for a high-level vision. Section 4 defines a model-driven framework for business modeling based on services while Section 5 formalizes our service-driven agent modeling approach. Section 6 specifically highlights the transformation process. The transformation process is illustrated on a case study in Section 7. Finally, Section 8 concludes the paper.

2. Related work

Most Agent-Oriented Software Engineering (AOSE) methods focus on the design stage of a MAS and poorly take organizational analysis as an important step into the software development. Furthermore, some AOSE methods claiming to be requirements-driven only rely on UML use-cases as development scenarios (like for example the *Multiagent System Engineering* (*MaSE*) method [22]); Tropos uses advanced organizational analysis through i* for business process analysis as a first step in the agent-software development. Nevertheless, the i*-driven approach from Tropos has some drawbacks (see Section 5.1.1) notably to represent a strategic perspective offering enough scalability abilities for dealing with large projects and to clearly forward engineer elements into a MAS design. With respect to the models included in the Tropos process as presented in [7–11], our framework:

- Includes the *Strategic Services Model* (*SSM*) which through its constituting elements allows to drive the model transformation process; more particularly it allows to:
  - deal with the issues of classical i*/Tropos notably highlighted by [12]. In our case, we are particularly interested by the way it offers to deal with scalability issues. Indeed, huge projects induce huge sets of i* elements (notably goals and tasks) and rapidly become unmanageable. Managing the project on the basis of services (i) addresses this issue particularly well (see [12,23]) and (ii) allows us to develop a software application highly aligned with the business the company is exercising;
  - tackle quality management and risk management basics at a high level of abstraction. Indeed, services can be impacted by *Quality Expectations* and *Threats* the overall company has/is facing with respect to its IT strategy. These are thus identified at strategic level and later forward engineered at tactical level into a set of i* softgoals, goals and/or tasks;
  - allows us to study involved actors' accountability – service governance perspective – and responsibility – service management perspective – in a unified framework.
  - manage the software project and deal with planning issues. The present paper focuses on the transformation issues related to model driven development and this element is thus outside the scope of the paper but it can also be used in an iterative project management perspective as illustrated in [24].
- Includes design models to forward engineer i* models into a BDI Agent-Oriented Design. The implementation of these models is outside the scope of the paper but their implementation using the *Java Agent DEvelopment Framework* (*JADE*) [25] can be found in [21]. JADE is a framework used for implementing MAS which conforms to the FIPA standard (see [26]). JADE simplifies the MAS development while ensuring standard compliance through a comprehensive set of system functions and their related agents.

Next to Tropos, other AOSE methods have been proposed; we position in the rest of this section our contribution with respect to these. Gascueña et al. [27] study model-driven techniques for the development of MAS. It notably compares, on the basis of a set of features, the technological aspects of INGENIAS [28], Prometheus [29] and PIM4Agents [30,31]; it then further studies Prometheus. When compared to Prometheus, our framework offers a strategic and tactical layer to drive the software process. Prometheus indeed directly starts the development with basic system goals and functionalities developed in the form of use-case scenarios. Moreover, Prometheus targets the *JACK intelligent agents* [32] as development platform. Our framework remains independent of any implementation language even if an implementation guidance with JADE is provided in [21]. The framework developed in this paper is business-driven thus tackling a layer that has not been linked to agent-design concepts through a transformation process in previous work.

To the best of our knowledge, no other framework or MAS development method has furnished a complete and consistent solution. For instance, *Multi-Agent Systems Development Methodology* (*MASD*) [33] claims to address the whole life cycle of an AOSE development; it nevertheless only envisages requirements as defined scenarios issued of use-cases.

Finally, we also highlight that Descartes Architect [34] is a CASE-Tool has been developed to support the creation and edition of the diagrams of our framework.

3. The need for a high level vision

Management and organizational theories involve several layers for decision making. Indeed, decisions do not have the same impact – from a marginal short term consequence to a major long-term strategy – so that their time horizon is variable. Traditionally, management sciences identify three levels of decision making in order to differentiate time horizons and resources that should be allocated:

- The Strategic Level in which decisions are top-level non-structured knowledge processes concerning general direction, long-term goals, philosophy and values of the organization.
- The Tactical Level in which more concrete, semi-structured decisions are taken aiming at implementing the strategy defined at the corporate level. The business units adapt this strategy in terms of policies under which the operations will take place.
- The Operational Level in which daily structured decisions are made to support tactical ones. Their impact is immediate, on a short-term, short range, and usually low cost. Operational