Evaluating the potential for integrating the OPEN and Tropos metamodels

B. Henderson-Sellers
Faculty of Information Technology
University of Technology, Sydney
Broadway, NSW, Australia

P. Giorgini
Department of Information and Communication Technology
University of Trento, Trento, Italy

P. Bresciani
ITS-Irst, Povo (Trento), Italy

Abstract *Methodologies involves both process elements and product elements. The OPEN Process Framework (OPF) focusses largely on process elements in the context of object-oriented systems development. The Tropos metamodel, on the other hand, focusses on early requirements engineering support for agent-oriented development. In a project to extend the OPF to support agent-oriented developments, we investigate the potential for amalgamation of these two methodological approaches and their metamodels, describing how the metamodel integration has been initiated.*

Keywords: methodology, agent orientation, object technology

1 Introduction

Although there is a wide range of usage of the term "methodology", here we will use it as a comprehensive term to describe all elements of the support needed for people to develop software. When one turns to identifying an appropriate *metamodel* to underpin the methodology, there are many similarities, independent of the development paradigm (e.g. agents or objects) being used. Details, such as modelling notations, of course differ significantly but the process elements are remarkably similar. Consequently, we can investigate the efficacy of combining the metamodelling elements from various paradigms — here we restrict ourselves to an amalgamation of metamodel elements from one specific OO-focussed methodological framework (namely OPEN [3]) and one specific agent-oriented methodology and its (partial) metamodel (namely Tropos [1]). This provides a first step in creating a unification of object-oriented and agent-oriented methodologies. The challenge to do so is both technical and political. In addition, it is critical that the result has industry appeal and is not just an "academic exercise".

2 The OPEN Metamodel

OPEN (Object-oriented Process, Environment and Notation: [3]) is a third-generation object-oriented methodology which has the characterising attribute that it supports method engineering (e.g., [2]) rather than being, *per se*, a usable methodology. This support is provided by means of a process metamodel that defines all the process elements needed to constitute a workable methodology, primarily for software development, together with generated instances of all the classes in this metamodel. Also included are rules for constructing

project-specific processes/methodologies from these process/method "chunks" using the principles of method engineering [2, 4].

The underpinning metamodel is described [3] in object-oriented terms i.e. classes and relationships between these classes. The classes in the metamodel describe process elements such as Phases, all Work Products and various Work Units, such as Activity and Task. Activity is at the highest level in the sense that a process consists of a number of Activities: largescale definitions of *what* must be done, used to configure the overall methodology. They are not used for project management or enactment because they are at too high an abstraction level. Instead, OPEN offers the concept of Task (in agreement with the Project Managers' Body of Knowledge) which is defined as being the smallest unit of work that can be project managed.

3 The Tropos Metamodel

Tropos [1] was designed to support agent-oriented systems development with a particular emphasis on the early requirements engineering phase. The stated aim was to use agent concepts in the description and definition of the methodology rather than using OO concepts in a minor extension to existing OO approaches. Tropos takes the BDI (Beliefs, Desires and Intentions) model [6], formulated to describe the *internal* view of a single agent, and applies those concepts to the *external* view in terms of problem modelling as part of requirements engineering. In other words, it uses AI-derived mentalistic notions such as *actors*, *goals*, *softgoals*, *plans*, *resources*, and *intentional dependencies* in all the phases of software development, from the first phases of early analysis down to the actual implementation. A crucial role is given to the earlier analysis of requirements that precedes prescriptive requirements specification. In particular, aside from the understanding of *how* the intended system will fit into the organizational setting, and *what* the system requirements are, Tropos addresses also the analysis of the *why* the system requirements are as they are, by performing an in-depth justification with respect to the organizational goals.

The published Tropos metamodel [1] focusses on a metamodel fragment to support early requirements engineering modelling. It uses agent notions (of plans, goals etc.) as its basis for describing the business requirements, the people involved and the overall problem. It does not address the scope of process elements supported in the OPF metamodel.

4 Integrating the Two Metamodels

The possible points of contact and/or intersection between the OPF and Tropos metamodels can now be identified based on study of the two metamodels (as outlined in the previous two sections). This can be summarized as follows:

- The need for an additional subclass of Language in the OPF
- The Tropos Actor and Role classes cf. the OPF Role and Producer classes
- Tropos Goals and OPF Milestones
- Tropos Means and OPF Tasks
- Tropos Plan and OPF Plan (name clash)

At this stage, these points of intersection are tentative. We hypothesize here that the above five issues might give us coupling points between the two approaches. We now analyze each of these hypotheses in turn, discussing each of them in more detail below.

4.1 Overall de¯nition and the Language metaclass

Our analysis of Tropos clearly identifies a missing class in the original OPF metamodel. The assumption originally was that only one modelling language kind was necessary for all modelling applications. Tropos [1] has shown how

the main modelling language used in object-oriented applications (UML [5]) is seriously deficient when applied to early requirements engineering. Consequently, at least with present-day technology, a different metamodel and notation is required beyond the UML to cater for the issues addressed by Tropos in early requirements engineering, particularly in terms of modelling the problem domain — UML is highly solution domain biased.

4.2 Actors, roles and producers

An Actor in Tropos is defined as an entity with strategic goals and intentionality within some context [1] that is a generalization of the concrete classes of Agent, Role and Position. A role typically characterizes the behaviour of an agent. This would therefore appear to be an identical definition to that in the OPF, wherein a Role is defined as "a functionally cohesive part that may be played by a Person on a Project" ([3], p50). Thus, the hypothesis that Role is a point of contact between the two metamodels appears to be well justified.

4.3 Goals and milestones

A goal in agent-oriented methodologies is usually associated with the internals of an individual agent. However, in Tropos, those agent-oriented ideas are also applied to the real world modelling undertaken in early requirements engineering, so that the notion of Goal is introduced as being the goal of an Actor (usually a person) or the goal of a process as exercised by a person. Tropos also supports the notion of subgoals in the sense of goals being achieved en route towards the final goal. There is thus some ambiguity at present in Tropos because goals are said to be both the final state we are aiming to achieve (the normal meaning of the word goal) but also a means by which we get there. In the light of this statement, we must evaluate the Tropos goal both in terms of the OPEN's milestone, which is a final state to be achieved, and also (in the next subsection) in the context of it, alternatively, as being the means to an end such that perhaps an OPEN Technique more closely models it.

The OPEN Milestone is a point in time at which something has been achieved or some task completed. Milestone is said also to be a kind of "Stage without Duration" [3]: an instance in time at which something is celebrated as having been achieved. Similarly, goals, when they are achieved, mark a point in time associated with the event of success (or failure). Goals can thus be represented as states e.g. in a UML State Transition Diagram. They are, according to the dictionary definition as "destination". Attaining a goal requires a plan which can describe a set of techniques ("how to"), zero or more subgoals that need to be achieved en route to the final goal and, possibly, a set of resources. This suggests that at least the more usual meaning of goal in the agent-oriented literature and as applied in Tropos to requirements engineering equates to the OPF's Milestone and the means to achieve the goal is equated to a more process-like component such as Task or Technique.

4.4 Goals and Techniques

The previous subsection suggested that Tropos goals and OPF Tasks are very similar. However, it also raised the thorny issue of a second definition in which a goal is said to be a *means* to achieve a(nother) goal. If goal, in this contextual meaning, is to represent a "means" of achieving, then it is a Stage With Duration (in OPF parlance) or possibly equated with an OPF Technique, a Technique being a statement of *how* something is achieved. However, this meaning is unlikely to be correct since in the English language there is no semantic link between the words "goal" and "technique". A typical dictionary definition of goal is that of "object of effort; destination" whereas technique is defined as "manner of (artistic) execution". Thus, the hypothesis that a Goal is equivalent to a Technique seems false.

4.5 Plans

In the OPF metamodel, a Plan is a class which inherits from Document. It is intended to represent those planning documents needed most in project management. Its peers are Requirements specification, Schedules, Architecture document and so on. All of these inherit ultimately from the Work Product metaclass.

In Tropos, a plan is a way of achieving a goal. It is not dissimilar to the OPF Plan except that it is more dynamic, representing a means to achieve a goal rather than being an end product of some process, activity or task (depending on your preferred terminology). Since a Tropos Plan metaclass is intended to represent not only plans in the business world, but also the agent-oriented enactment of these, it appears that there is no equivalence with the OPF Plan. A renaming of one or other class is therefore required in any merger of these two metamodels.

5 Goals, Tasks and Plans

An interesting outcome of this analysis is the need for agent-oriented methodologies (and indeed those elements describing internal design of agents) to be clear about the terminology being used with respect to the technical terms of Goal, Task and Plan. In our context, we have examined Tropos and OPF and find both to be poorly defined.

An agreed meaning for goal (or milestone in OPF) is clearly an end point — something to be achieved e.g. I want to be in Beijing for the Olympics. It defines a desired state. The question is how to achieve that state.

The subgoals and actions/activities are part of a plan to achieve the goal. They recommend decomposition of a state (the goal) into substates (acceptable) and actions (inappropriate since the actions effect a change of state; they are not components *of* that state). Tropos follows this BDI approach but depicts Plans and Goals as peers along with Resources.

A Plan is said [1] to provide an abstract description of a way of doing something, the execution of which is one means of achieving the goal. Attainment of subgoals can also be viewed using the concept of a *pre-condition*.

It is thus clear that there is some ambiguity in both the object-oriented and agent-oriented literature on the differences between goals as end states and the means to achieve those goals and what these means should be named in order to be unambiguous. Such a conclusion only surfaces when one is forced (by choosing to use a metamodelling approach) to create a metalevel description and tight definition of the ontology to be used for (in Tropos) agent-oriented requirements engineering.

References

[1] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopolous, J. and Perini, A.: Tropos: an agent-oriented software development methodology. Journal of Autonomous Multi-Agent Systems (2003) in press

[2] Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Inf. Software Technol. **38(4)** (1996) 275–280

[3] Firesmith, D.G. and Henderson-Sellers, B.: The OPEN Process Framework. An Introduction, Addison-Wesley, Harlow, UK (2002) 330pp

[4] Henderson-Sellers, B.: Process metamodelling and process construction: examples using the OPEN Process Framework (OPF), Annals of Software Engineering, **14** (2002) 341–362

[5] OMG: OMG Unified Modeling Language Specification, Version 1.4, September 2001, OMG document formal/01-09-68 through 80 (13 documents) [Online]. Available http://www.omg.org (2001)

[6] Rao, A.S. and Georgeff, M.P., BDI agents: from theory to practice, Technical Note 56, Australian Artificial Intelligence Institute (1995)