

Designing a Multi-Agent System for Integrated Protection in Agriculture *

A. Perini^a and A. Susi^a

^a*ITC – IRST, Via Sommarive 18 - Loc. Pantè
I-38050 Povo, Trento - Italy
{perini,susi}@irst.itc.it*

Abstract: Integrated Protection (IP) in agriculture consists of a set of practices aimed at favoring the set up of a development model characterized by a reduced environmental impact. The application of IP practices in plant disease management by growers and agronomists requires both specialistic skills, historical data and information on chemicals and on low impact techniques for pest management. These sources of information and knowledge are distributed among different actors in the agriculture production system.

Recent approaches in developing decision support systems for agriculture, and more generally for environmental problems management, tend to adopt a “systemic” approach. That is to say a problem is considered for its dependencies to different skills and responsibilities, and the proposed applications aim to be integrated in larger information systems. So basically, two main dimensions of complexity have to be considered while analyzing the problem: the organizational dimension dealing with all the dependencies between the domain stakeholders, and the technical dimension concerning the study of natural plant protection techniques.

These considerations motivates our choice of using an agent-oriented methodology for software development in designing a multi-agent system at support of apple growers and technicians of the advisory service. The methodology, called Tropos, gives a central role to early requirements analysis and allows to derive system functional and non-functional requirements from a deep understanding of the domain stakeholders goals and of their dependencies.

Keywords: Agent Oriented Software Engineering; Integrated Pest Management.

1 INTRODUCTION

Plant disease control, according to Integrated Protection (IP) directives, is to maintain the damage on crop under a tolerance threshold which can be economically acceptable. The application of IP practices by growers and agronomists requires specialistic skills, historical data and information on low impact techniques for plant disease management and on the chemicals authorized by the International Organization for Biological Control and by the local government. These sources of information and knowledge are distributed among different actors in the agriculture production system.

Recent approaches of AI applications to agriculture, and more generally to environmental problems, tend

to adopt a “systemic” approach. That is to say a problem is analyzed in terms of all the knowledge, the data and the responsibilities it depends on. So, the proposed applications aim to be integrated in larger information systems exploiting the fact that different organizations may manage information sources and resources that are relevant to problem solutions. This basically has a twofold effect in applying AI techniques to environmental problems: first, an organizational analysis becomes a necessary step when specifying application requirements; second, the resulting applications should be designed in terms of a set of specific, interrelated services, such as information providing or reasoning services, that are provided by specialized software agents. Depending on the required capabilities, each software agent will be built using specific AI techniques for reasoning or will wrap existing DBMS, or Geographical Information Systems (Avesani et al. [1998]).

*The work presented in the paper is partially funded by the Italian Ministry of Scientific and Technological Research.

This paper focuses on the requirement analysis and the design of a Multi-Agent System (MAS) devoted to support decision making by the technicians of the agricultural advisory service when managing plant diseases (Perini [2000]). This system is being developed in the context of a project involving experts on IP techniques and agronomists.

We adopt the *Tropos* methodology, described in Perini et al. [2001] and in Giunchiglia et al. [2001b], an agent oriented software development methodology (see for an overview of current approaches in agent oriented software engineering Ciancarini and Wooldridge [2001], Wooldridge et al. [2001]).

The paper is structured as follows. Section 2 recalls the main concepts and the practical steps of the *Tropos* methodology. Sections 3 and 4 describe the results of modeling actor coordination during early and late requirements analysis in Tropos. Section 5 describes the initial phase of the architectural design of the system. Finally, conclusions and the future work are presented in Section 6.

2 THE METHODOLOGY

The *Tropos* methodology is an agent-oriented software development methodology based on two key ideas, namely: (i) the use of knowledge level concepts, such as actor, goal, plan and dependency between actors, along the whole software development process, and (ii) the critical role assigned to the preliminary phase of requirements analysis aimed at understanding the environment in which the system-to-be will operate. Tropos covers five software development phases: *early requirements analysis*, *late requirements analysis*, *architectural design*, *detailed design*, and *implementation*. From a practical point of view the methodology guides the software engineer along the whole process in building conceptual models, with the help of a visual modeling language which provides an ontology including knowledge level concepts. The language provides also a graphical notation for concepts and a set of diagrams for viewing the models properties: *actor diagrams* for describing the network of dependency relationships among actors, as well as *goal diagrams*, for illustrating goal and plan analysis from the point of view of a specific actor. The purpose of conceptual modeling in each phase of the software development process is briefly recalled below.

Early Requirements analysis focuses on the understanding of a problem domain by studying an *exist-*

ing organizational setting where the system-to-be will be introduced. Social actors and software systems that are already present in the domain are modeled as actors with their individual goals and with mutual, intentional dependencies.

Late Requirement analysis focuses on the system-to-be which is introduced as a new actor into the model. The system actor is related to the social actors in terms of dependencies; its goals are analyzed and will eventually lead to revise and add new dependencies involving a subset of the social actors (the users).

Architectural design defines the system's global architecture in terms of subsystems, that are represented as actors. They are assigned subgoals or subplans of the goals and plans assigned to the system. Each actor is characterized by: (i) a set of individual capabilities and (ii) a set of social capabilities required by actor coordination. The result of the architectural design is the mapping of the system subactors (with their capabilities) to a set of agents.

Detailed design aims at specifying the agent microlevel, defining: (i) *capabilities* and *plans* using the AUML activity diagram; (ii) *communication* and *coordination* protocols using the AUML sequence diagrams. A mapping between the Tropos concepts and the constructs of the implementation language and platform is provided. So, for instance, considering a BDI platform (Rao and Georgeff [1991]), each agent capability is defined in terms of beliefs, plans and events and each plan in terms of atomic actions. *Interaction* and *communication protocols* required by each coordination process involving the agent is also designed at this time.

The *Implementation* activity produces an implementation skeleton according to the detailed design specification. Code is added to the skeleton using the programming language supported by the implementation platform.

Unlike other agent oriented software engineering methodologies, the *Tropos* methodology covers all the software development phases above described (as explained in Giunchiglia et al. [2001a]).

In the following sections we will describe the application of the methodology to the design of the IP decision support system we are developing. We will focus only on the first phases of the development process, due to lack of space.

