International Congress on Environmental Modelling and Software

Jul 1st, 12:00 AM

Designing a Multi-Agent System for Integrated Protection in Agriculture

A. Perini

A. Susi

Perini, A. and Susi, A., "Designing a Multi-Agent System for Integrated Protection in Agriculture" (2002). *International Congress on Environmental Modelling and Software*. 165.
https://scholarsarchive.byu.edu/iemssconference/2002/all/165

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]).

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 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.
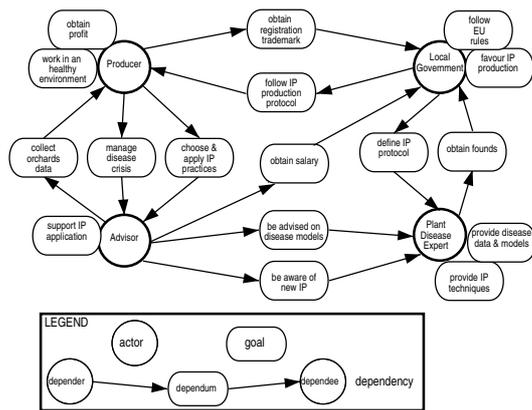
Figure 1: The actor diagram showing a portion of the IP organizational setting model. Early requirements model.

3 EARLY REQUIREMENTS

The analysis starts identifying the stakeholders of the agriculture production system of our region and modeling them as actors, depicted by circles in Figure 1:

- The actor Producer represents the apple grower who pursues objectives such as to obtain a profit following acceptable market strategies, and to work in a healthy environment.

- The actor Advisor models the technician of the advisory service that has been set up by the local government in order to provide a support to producers in choosing and applying the best agricultural practices and techniques (see the goal support IP application). The advisor plays a key role in our area since the majority of producers are not professional farmers, they lack specific skills and/or are not confident enough of adopting an IP approach.

- The actor Local Government plays both an institutional and a practical role in promoting IP diffusion in our region (see the goals favor IP production, follow EU rules). It sets up a list of admissible chemicals and quantity limits, according to the European Union agreements. These rules are yearly updated and coded into a production protocol.

- The actor Plant Disease Expert represents the researcher in biological phenomena and in agronomical techniques. Among his/her objectives that of transferring research results

directly to the production level, for instance providing disease data and models and new effective pest management techniques (see the goals provide disease data & models, provide IP techniques).

The actor diagram in Figure 1 shows some of the critical coordination processes between the domain stakeholders which, at a macroscopic level, result in a joint effort to disseminate IP.

In particular, the actor Producer depends on the actor Local Government for obtaining a product certification (i.e. obtain registration trademark) that states that he/she follows IP practices, as required by specific market sectors. The local government sets up the yearly IP production protocol and issues the desired certification only to the producers that follows it. So, the actor Local Government depends on the actor Producer in order to have its goal follow IP production protocol satisfied. As already noticed, the actor Advisor plays the role of mentor, with respect to the producer, in carrying up apple production according to the IP rule. The actor Producer depends on the actor Advisor in order to choose & apply IP practices according to the production protocol and in order to manage disease crisis which may occur in case of unforeseen events and that requires to adopt an appropriate remedy action, still IP compliant. Viceversa, the actor Advisor depends on the actor Producer for satisfying his/her goal to collect orchards data in order to maintain an updated picture of the disease presence and evolution in the area under control. Moreover, the Advisor depends on the actor Plant Disease Expert in order to use effective disease models (i.e. to attain the goal be advised on disease models and to get information on new IP techniques (be aware of new IP). Both actors, the Advisor and the Plant Disease Expert are funded by Local Government). The goal dependency define the IP protocol between the Local Government and the Plant Disease Expert closes the loop. It models the contribution of the expert in providing the technical skills necessary for defining a production protocol that follows the European Union strategic directives.

The Early Requirements model is further refined by considering each actor and by analyzing its goals. New actors and dependences can be added in the model. The goal diagram depicted in Figure 2 shows the analysis of the goal support IP application, from the point of view of the actor Advisor.

The goal support IP application contributes positively to the fulfillment of both goals choose &
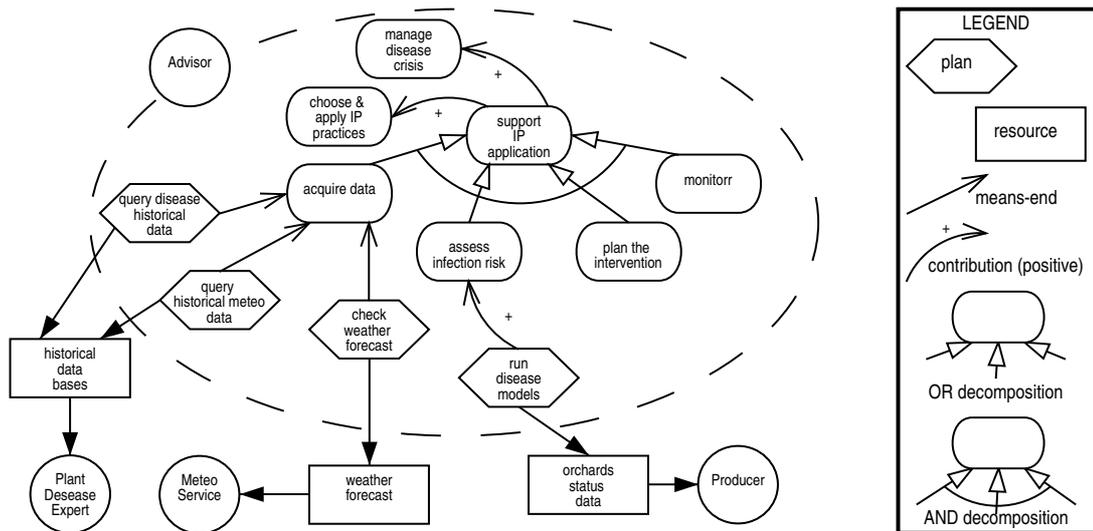
Figure 2: The goal diagram of the goal support IP application analyzed from the point of view of the actor Advisor.

apply IP practices for which the actor Producer needs to coordinate with the actor Advisor. The goal can be AND decomposed into a set of more specific subgoals, i.e. acquire data, assess infection risk, plan the intervention and monitor the situation after the intervention. The analysis follows considering the plans that the advisor performs in order to satisfy them. Given the goal acquire data the following plans are means to satisfy it getting data that are relevant in disease management, i.e. historical data on the presence of the disease in the area, historical meteo data and weather forecast (see the plans, depicted as hexagonal shapes in Figure 2, query disease historical data, query historical data and check weather forecast). The dependency between the actor Advisor and the actor Plant Diseases Expert for the resource disease historical data models the fact that the advisors usually perform searches into the data bases on disease data held by the experts. Analogously, historical meteo data and weather forecast are data to be obtained from the actors that institutionally held them. The plan run disease model is a means to attain the goal assess infection risk. Specific simulation models exist for the most critical plant diseases. They require data from the orchard in order to produce updated estimates.

Analogously, the remaining subgoals can be analyzed with the aim of identifying advisors plans and coordination processes with the other actors that allow for the execution of these plans.
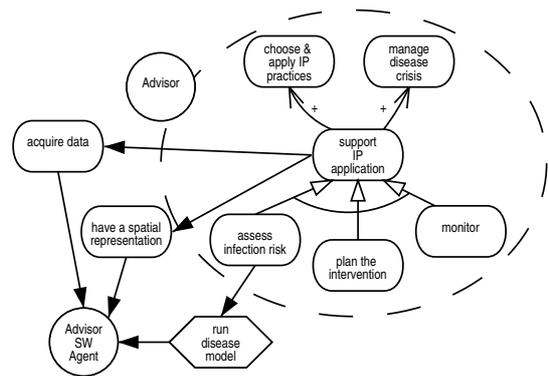

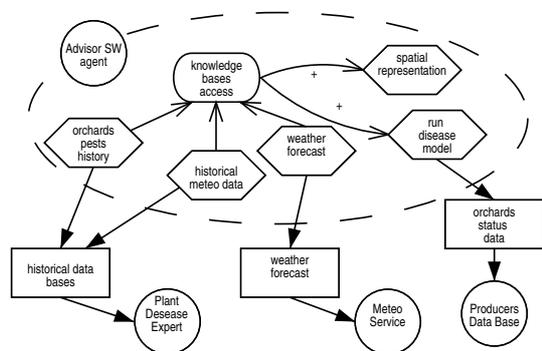Figure 3: The advisor goal diagram for plant disease management. Late requirements model.

Figure 4: The Advisor SW Agent goal diagram. Late requirements model.
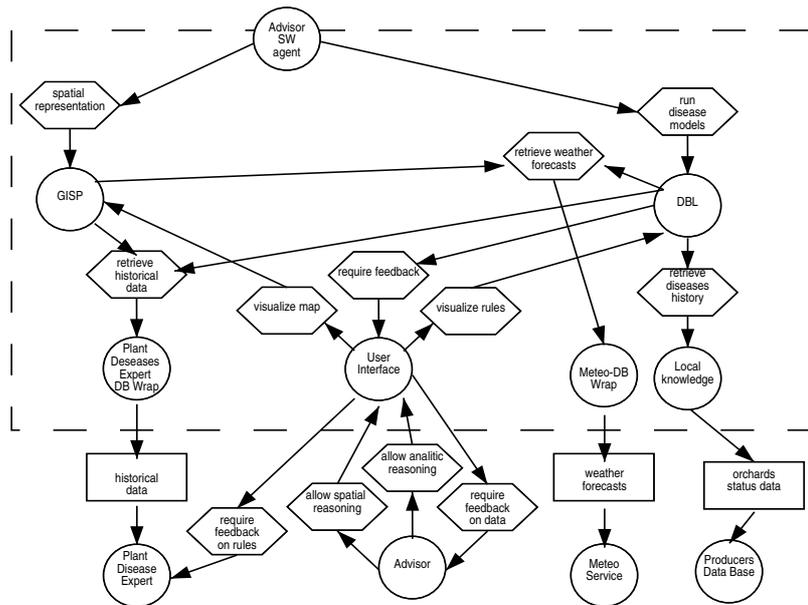
411

Figure 5: The Architectural design.

4 LATE REQUIREMENTS

During late requirements the system-to-be is introduced as a new actor into the conceptual model and analyzed in relation with the social actors. Figure 3 depicts a fragment of the late requirements model where the actor Advisor SW Agent models the decision support system at use of the advisors when managing plant diseases. In particular, the actor Advisor delegates the system-to-be for the fulfillment of the goals acquire data and have a spatial representation of the data on an electronic map. Moreover, it depends on the system for the execution of the plan run disease model. This implies that also the dependencies to the other social actors related with these model elements have to be appropriately revised. For instance all the coordination processes with actors holding data relevant for disease management have been delegated to the system-to-be actor.

The goal diagrams of the actor Advisor is modified accordingly. The resulting diagram is shown in Figure 4 [2].

5 ARCHITECTURAL DESIGN

A portion of the system architecture model is illustrated in the actor diagram depicted in Figure 5. Six

sub-system actors have been introduced to which the Advisor SW agent delegates the sub-goals and the plans that were found during the goal analysis performed from the point of view of the system actor (see Figure 4). They are:

- the actor GISP (Geographic Information Services Provider) to which the Advisor SW agent delegates the plan spatial representation;

- the actor DBL (Disease Behavior Learner), which performs the plan run disease models on the basis of information extracted from the data concerning the disease using machine learning techniques (Mitchell [1997]), like decision tree, that, given a set of parameteres and their values for a particular territory in a certain period of time, are able to produce sets of rules induced by these data an describing the behavior of the insects populations;

- three wrapper actors, namely, the PDE-DBW (Plant Diseases Experts DB Wrapper) which executes the plan retrieve historical data returning both, meteo and orchard historical data from an existing data base hold by the actor Plant Disease Expert; the wrapper of the database of the meteo service, called Meteo-DBW (Meteo Service DataBase Wrapper) which performs the plan retrieve weather forecast; the wrapper of the existing data base contains data relative to the orchards belonging to the area under the

---

control of the advisor (see the actor Produc-ers DB);

- the actor User Interface which takes care of a the execution of a set of plans guiding the interaction between the user of the application (the actor advisor and the other specialized actors of Advisor SW agent. For instance, it executes the plan allow spatial reasoning committed upon user request and during its execution may commit the actor GISP to execute the plan visualize map.

The system architecture model can be further enriched with other system actors according to design patterns (as explained in Hayden et al. [1999]) that provide solutions to heterogeneous agents communication and to non-functional requirements.Further steps are required in *Tropos* to complete the architectural design, such as that aimed at identifying actor capabilities from the analysis of the dependencies going-in and -out from the actor and from the goals and plans that the system actors will carry on. The architectural design ends with a mapping of the system subactors to a set of agents. Each agent is characterized by a set of the capabilities identifyied in the actor diagram.

The next phase in the Tropos development process is detailed design, where the agent micro-level is specified in terms of agent capabilities and plans. Here a set of diagrams proposed in Agent UML can be used. The detailed design specifies the interaction between agents, that will allow to realize the coordination processes designed at the architectural design level. At this stage we exploit specification of agent communication protocols available in the MAS literature.

6 CONCLUSION AND FUTURE WORK

This paper describes the requirements analysis and the design of a multi-agent system at support of apple growers and technicians of the advisory service performed using *Tropos*, an agent oriented software engineering methodology.

We discuss the early requirement and late requirement analysis specifying the reasons for dependencies between social and system actors. In particular, goal and plan delegation from social actors (the users) to system actors is pointed out.

A sketch of the architectural design, according to the *Tropos* methodology is also given. The architecture includes a set of actors (agents) wrapping existing information systems and interacting with new

actors providing estimates on the evolution of a specific plant disease.

We are currently developing some of the agents of the Advisor SW Agent system. In particular, the GISP agent and the DBL agent with reference to a critical pest for apple, called the Codling Moth (Cydia Pomonella).

REFERENCES

A. Avesani, A. Perini, and F. Ricci. The Twofold Integration of CBR in Decision Support Systems. In *AAAI98 Workshop on Case-Based Reasoning Integrations*. Madison, July 1998.

P. Ciancarini and M. Wooldridge, editors. *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in AI*. Springer-Verlag, March 2001.

F. Giunchiglia, J. Mylopoulos, and A. Perini. The Tropos Software Development Methodology: Processes, Models and Diagrams. Technical Report 0111-20, Istituto Trentino di Cultura - IRST, November 2001a.

F. Giunchiglia, A. Perini, and F. Sannicolò. Knowledge level software engineering. In *Proceedings of ATAL 2001, Seattle*. Springer Verlag, December 2001b. Also IRST Technical Report 0112-22, Istituto Trentino di Cultura, Trento,Italy.

S. Hayden, C. Carrick, and Q. Yang. Architectural design patterns for multi-agent coordination, 1999.

T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

A. Perini. A web advisor for integrated protection. In *ECAI2000 Workshop AI in Agriculture and Natural resources*, 2000.

A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, and J. Mylopoulos. A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. In *Proceedings of the Fifth International Conference on Autonomous Agents*, Montreal CA, May 2001. ACM.

A.S. Rao and M.P. Georgeff. Modelling rational agents within a BDI-architecture. In *Proceedings of Knowledge Representation and Reasoning (KRR-91) Conference*, San Mateo CA, 1991.

M. Wooldridge, P. Ciancarini, and G. Weiss, editors. *Proceedings of the Second International Workshop on Agent-Oriented Software Engineering (AOSE-2001)*, Montreal, Canada, May 2001.