Understanding the Requirements of a Decision Support System for Agriculture. An Agent-Oriented approach*

Anna Perini and Angelo Susi

ITC-irst, Trento-Povo – Italy

Email: {perini,susi}@itc.it

Abstract

Recent experiences in building environmental Decision Support Systems (DSS) point out the need of a deep understanding of the application domain before starting system design. The application domain under consideration has to be characterized in terms of stakeholders roles and of their objectives and in terms of the decision making processes these stakeholders are involved in for environmental management.

In this chapter we will focus on the domain and requirements analysis faced during the development of a DSS for Integrated Production (IP) in agriculture. The DSS supports the technicians of the agriculture advisory service while helping producers to apply IP guidelines in growing apple in Trentino, a region in Northern Italy.

Examples taken from the analysis that has been performed adopting the *Tropos* methodology, an agent-oriented software engineering methodology, will be illustrated, with the aim of proving the effectiveness of the methodology in understanding strategic dependencies between domain stakeholders and in identifying the DSS requirements in terms of goal and plan delegation from stakeholders (the users) to the system-to-be. A short overview of the DSS will also be given.

1. Introduction

Recent approaches in building decision support systems (DSS) for environmental management tend to adopt a "systemic" approach. That is to say a problem domain is analyzed in terms of its stakeholders roles and objectives and in terms of the decision making processes these stakeholders are involved in for the management of the environmental problem under consideration.

* PICO project Technical Report. The "Decision Support Systems for Integrated Protection in Agriculture" (PICO) project has been funded by the Italian Ministry for Research (MIUR), 2000-2003. The aim of the project was to develop a decision support system for agronomists and producers dealing with fruit disease management, according to Integrated Production.

Environmental management requires typically to exploit various information sources which may be managed by different organizations dealing to the problem of integrating heterogeneous, dynamic and distributed information sources.
Examples come from previous experiences in developing DSS for forest fire fighting [3][4] and in works that analyzed a knowledge management perspective in environmental management [11][12].
Moreover, environmental management has to be performed also when data on environmental phenomena are not completely available or trustable and this requires understanding alternative problem solving strategies (see for example [7][8][3] where experiences in integrating different automated reasoning techniques in developing environmental DSS are presented).

We refer here to a project devoted to the development of a DSS for Integrated Production (IP) in agriculture. IP consists of a set of practices aimed at favoring the set up of a production model characterized by a reduced environmental impact. The application of IP practices requires a deep knowledge of natural phenomena, of agronomic techniques and on modern chemicals, as well as updated information on the rules that the local government adopts according to the European Union directives.
A closer look to this domain points out a complex network of dependencies between the agriculture organization stakeholders which need to be deeply understood in order to identify the requirements of an effective DSS for IP.

We adopted an agent-oriented methodology for software development, based on the *Tropos* methodology [9] which 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.

The rest of this chapter is structured as follows: section 2 presents the IP domain in agriculture and gives examples referring to disease management in growing apples; section 3 briefly recalls the *Tropos* methodology; section 4 illustrates the analysis of the agriculture domain and how this analysis supported us in eliciting DSS requirements;

sections 5 and 6 describe respectively the system requirements analysis and how it drove an architectural design which has been performed accordingly with well accepted design principles, such as decoupling and cohesion. A short overview of the resulting DSS will be presented in section 7. Related work is presented in section 8 and conclusion given in section 9.

2. Integrated Production in Agriculture

Integrated Production (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 apple pest management, for instance, requires to use low-impact and/or natural techniques to maintain the disease damage of a crop under a specific tolerance threshold, that is an economically acceptable number of damaged fruits.

The European Union proposes norms, guidelines and funds to favor the adoption of IP and delegates the local governments of the European countries to support and monitor IP application. The research on agronomic techniques compliant with the IP principles plays a crucial role in this context, as well as the dissemination of its result to the producers.

More generally, the acquisition by the producers of information and knowledge on updated norms and on new research findings can be an hard task without specific help. On the other side, favoring the adoption of IP practices and monitoring for their correct application, will become an impossible task for the local government, without the collaboration of the producers themselves.

Yearly, the local government of Trentino, defines a set of guidelines for the application of IP in the region, in agreement with the main actors of the local agricultural production system which include a list of admissible chemicals and quantity limits, according to the European Union agreements (these guidelines are also called *production protocol*).

Moreover the local government has set up an Advisory service whose technicians help the producers in applying these guidelines. Finally, specific committee monitor the producers and will allow to certify or not their product.

In our project we focused on IP pest management. For sake of simplicity, in exemplifying our work we will consider the case of the Codling Moth (*Cydia Pomonella*), an important pest for apple orchards in Trentino, which tend to be resistant to pesticides. Figure 1 shows a Codling Moth at larvae and adult stage and the type of damage it produces on the fruit (left side). Typically two generations develops in a season, as shown by the data plotted in the right part of Figure 1.

According to IP guidelines, an acceptable tolerance threshold for this pest in June can be expressed as 0.2% of damaged small-fruits and as 1% of damaged fruit by the end of August. Figure 2. A set of remedial actions should be performed in order to approach the latter threshold if the number of damaged fruits in June is greater than 0.2%. This requires monitoring the orchard about every 15 days (following the pest life-cycle in the). Moreover, natural plant protection techniques must be known - for example the use of antagonists or low impact pesticides - in order to determine the actions capable of reaching the 1% damage threshold goal.

Figure 1: The Cydia Pomonella pest: larva, adult insect and an example of the damage it produces in the fruits (left); its development cycle in a season (right).

IP management can be characterized along the following three main tasks: the diagnosis of a disease from its observable manifestations; the assessment of the severity degree of an orchard infection and the choice of remedial actions.

The first task rests on domain knowledge, such as a classification of disease manifestations respect to a set of candidate diseases, and historical data on previous infections in a given area [14].

A deeper analysis of the second and third tasks shows that the domain stakeholders currently address them in different ways at two different moments, namely, at the beginning of the year, when strategic decisions for the following crop are taken and during the season, in reaction to unforeseen events which may prevent the correct implementation of the selected growing strategy.

Preventive strategy made by growers at the beginning of the year rest on an estimate of the plant disease risk, which is derived from a careful analysis of historical data, and on economical goals. The technicians of the advisory center support the growers in making this assessment and in deciding which preventive strategy to adopt. For example, for the Codling Moth, a preventive strategy consists in using pheromone trapping (PT), a techniques which allows to lower pest mating. Designing an effective PT system requires to decide where to put pheromone bars in the orchard and with which density, taking into account the geometrical features of the orchard and the presence of possible infection sources,

During the season, that is during the implementation of the selected IP management strategy (or in the absence of a preventive strategy), unforeseen disease could menace the crop and this requires interleaving actions aimed at assessing the extent of the disease with remedial actions. This task requires expert knowledge on the development models of the disease and on specialized remedial techniques. This expert knowledge is provided by research studies on disease biology and on agronomic techniques. Assessment activity requires the periodical evaluation of the data gathered on the field taking into account the meteo data and the weather forecast. In particular, deciding which are the most relevant data for an effective assessment and how to use them for evaluating the infection risks are both decision making tasks that need support. The aim is to minimize the number of observations to be performed by growers and technicians in the field and to alert them for critical events that need a direct measurement of the disease evolution in the field. Having an effective model for predicting the development of a plant disease during the season, in terms of a minimal set of environmental parameters, could be a solution to both problems.

3. The modelling approach

In order to identify the requirements of a DSS which could enable the application of IP in pest management we adopted the *Tropos* methodology [9][15], an agent-oriented software engineering methodology which recognize a crucial role to the early phases of system requirements.

Agent-oriented software engineering is being proposed as a general framework for analysis and design, in spite of having adopted Multi-Agent System (MAS) as the target development platform, especially when complex, distributed systems are concerned [10][16][27].

The idea is that agent-oriented methodologies, which are founded on notions such as those of agent, goal, plan, are inherently *intentional*, and allow to model explicitly reasons behind the needs of the application domain stakeholder, as well as reasons behind system requirements.

Tropos provides a conceptual modeling language, derived from the *i* framework* [28], which uses a small set of intentional notions, such as actor, goal and dependency, and a graphical notation to build views on the models in terms of actor and goal diagrams. This approach proved to be suitable for modeling the organization where the system-to-be has to be introduced. In particular, the *Tropos* methodology covers five software development phases: *early requirements analysis*, *late requirements analysis*, *architectural design*, *detailed design,* and *implementation*. *Early Requirements* analysis focuses on the understanding of a problem domain by studying an *existing 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 introduction of the system-to-be 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. *Detailed design* which aims at specifying the agent micro-level and code generation follow.
In this chapter we will illustrate how we applied an agent-oriented approach based on the *Tropos* methodology to understand the IP agricultural domain and to derive the requirements of a DSS, moreover we will show how general strategies for software design and architecting have been addressed.

Our analysis of the domain of IP in agriculture aimed at identifying the actors who performs the plant disease management activities and their main goals; at describing the organizational processes these disease management activities rest on; at pointing out strategic dependencies between these actors, that is to day "who depends on whom for what"; at identifying the most critical information and knowledge required by disease management processes; at characterizing the critical decisional steps in these processes, together with alternative ways to perform a decision making step; and finally at identifying which decisions could be supported by a DSS and who have to be its user.

4. The IP domain analysis

The domain analysis started identifying the stakeholders, that is social actors and software systems that are already present in the domain, of the agriculture production system of our region. They are modeled as actors, depicted by circles, in Figure 2.
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 special 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).

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 infection data and disease simulation models, as well as new effective pest management techniques (see the goals provide disease data & models, provide IP techniques). The actor diagram in Figure 2 shows some of the critical dependencies between the domain stakeholders which, at a macroscopic level, result in a joint effort to disseminate IP.

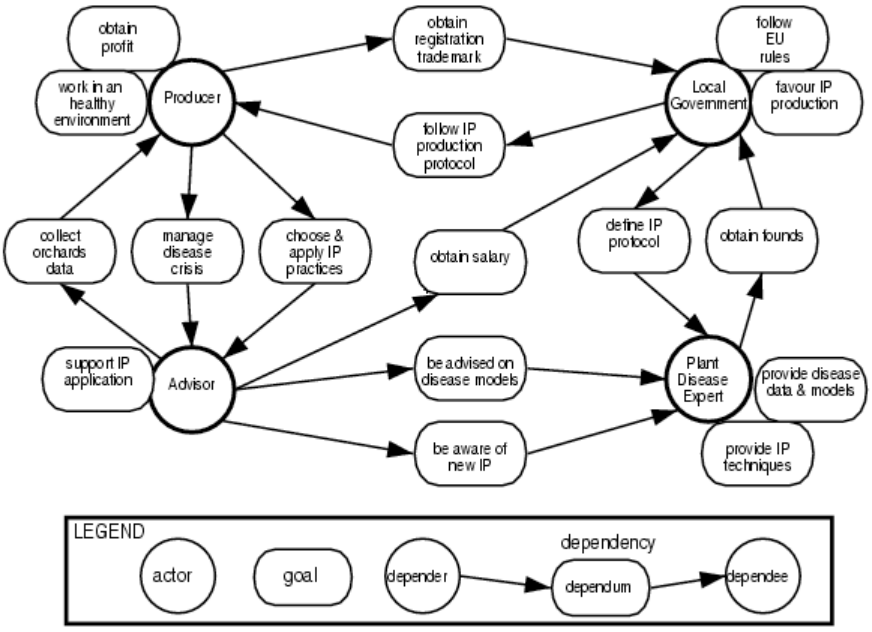

Figure 2: Description of the Agricultural domain in Tropos

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 follow 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. So the actors Advisor and Producer closely depends: 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 their 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). 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 all its actors and by analyzing their goals. New actors and dependences can be added in the model. The goal diagram depicted in Figure 3 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 & apply IP practices and manage disease crisis for which the actor Producer depends from 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. Moreover, the softgoal have a spatial representation that is being able to visualize the data on a map of the whole area under control by the advisor shall allow him/her to perform in a more effective way both the data acquisition activity and the assessment of an infection risk (see the two positive contribution links in Figure 3).

In the following we consider the plans that the advisor performs in order to satisfy them according to current practices. Means to satisfy the goal acquire data consists in getting data resulting from

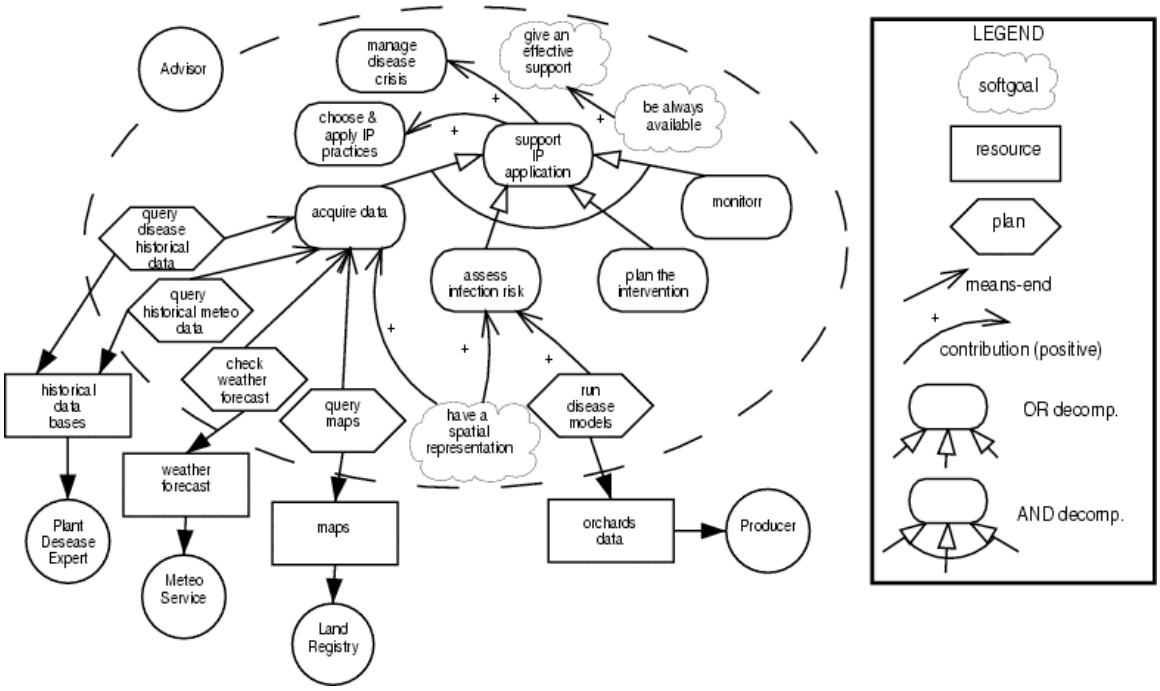

Figure 3: Advisor goal model for the goal support IP application.

observations and measurements activities performed, each season, in the orchards, as well as in getting current meteo data. This is modeled in Figure 3 with a set of plans, depicted as hexagonal shapes, which are related to the goal acquire data trough specific means-end relationships, i.e. query disease historical data, which refers to historical data on the presence of the disease in the area, query historical meteo data which refers to historical climate and check weather forecast (the current meteo data). The analysis points out a set of interaction processes related to the execution of these plans, they are modeled in terms of resource dependencies. For instance, the dependency between the actor Advisor and the actor Plant Disease Expert for the resource historical data

bases models the fact that the advisors usually perform searches into the data bases on disease data held by the experts, as well as on climate data relative to the area under their control. The plan run disease models is a mean to attain the goal assess infection risk. In current IP practices, the advisors exploit phenology and/or epidemiological models which help them in analyzing the behavior of a plant disease. For instance, they allow them to estimate both the disease stage and the infection extent. These models require specific data from the orchard in order to produce updated estimates. Analogously, the remaining subgoals can be analyzed with the aim of identifying advisor plans and dependencies with the other actors.

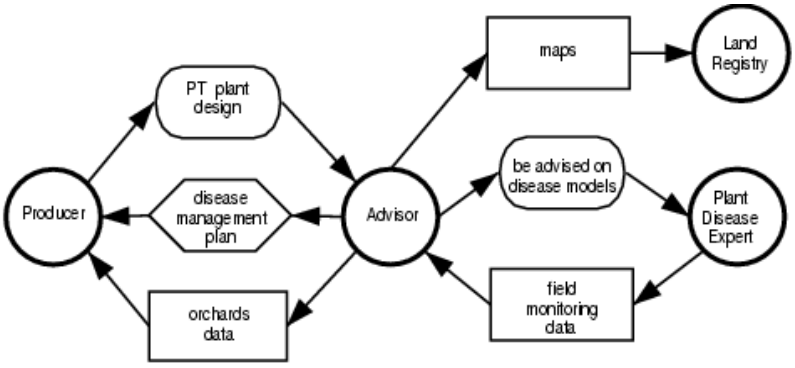

Figure 4: IP domain's strategic dependencies (BEFORE the introduction of the system).

In Figure 4 is shown an excerpt of the domain strategic dependencies for the *Codlyng Moth* pest management problem. Here the Producer delegates to the Advisor the goal of designing an appropriate pheromone trapping (PT) system for her/his orchard (see the goal PT plant design) to the other side the Advisor depends on the Producer to get the necessary orchard data and to have the plan disease management plan executed. The Advisor depends on the Plant Disease Experts to be advised on disease models and from the Land Registry (the region maps archive) for the use of the maps. The Plant Disease Expert will depend on Advisor for acquiring data collected through monitoring activities in the field.

All the three stakeholders modeled in Figure 4, namely the producer, the advisor and the expert could be supported by specific DSS functions. In the following, we will focus on

the advisor to show how intentional analysis, according to the *Tropos* methodology, could support the elicitation and the analysis of the requirements o a DSS.

5. System requirements elicitation and analysis

During late requirements analysis the system-to-be, that is the decision support system devoted to the advisors when dealing plant disease management, is introduced as a new actor into the conceptual model, it is represented by the actor Advisor SW Agent.

The main questions which drive the analysis in this phase are the following. What the system can do for its user and how? And how the adoption of a system will affects the IP domain strategic dependencies?

Figure 5, depicts an example of late requirements analysis. The advisor (i.e. the system's user) rests on the system (actor Advisor SW Agent) for the fulfillment of some goals and plans discovered during the goal analysis in the ER model depicted in Figure 3, such as goals related to the acquisition of data, to the use of a spatial representation of the territory. Figure 5, shows a sketch of the resulting LR goal diagram for the Advisor SW Agent, which allows to analyze how the system can support the user.

According to domain knowledge, the goal acquire data can be operationalized in three plans: orchards pests history, historical meteo data, weather forecast, devoted to the acquisition of data from external resources (historical data bases and weather forecast) that generating a new set of dependencies with the domain actors. The acquisition of data positively contributes to the achievement of the plan run disease model and to the goal use GIS techniques (that represents a possible means to satisfy the softgoal have a spatial representation).

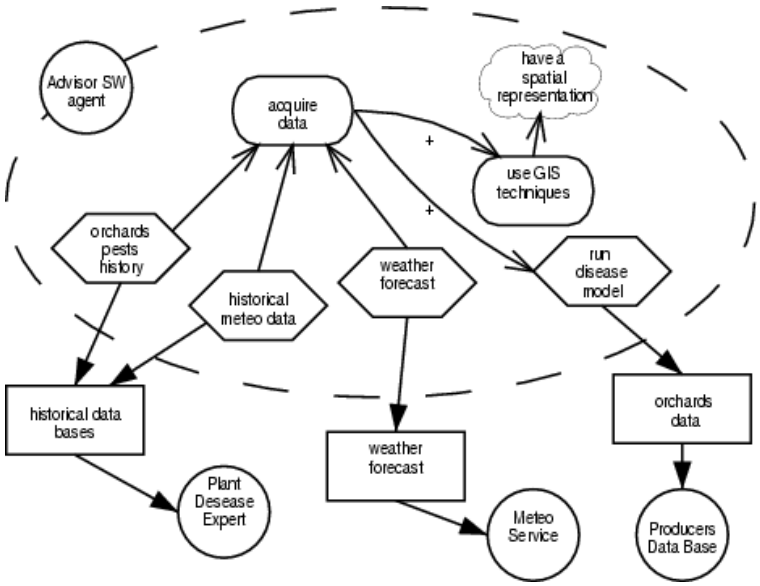

Figure 5: Late Requirement Goal diagram for the Advisor SW Agent.

Figure 6, depicts a fragment of late requirements actor diagram which points out how of the stakeholders strategic dependencies will be affected by the adoption of a DSS. Here we refer to the dependencies modeled in Figure 4, with reference to the example of the *Codlyng Moth* pest.

The new dependencies are depicted in bold. In particular, the actor Advisor delegates the system-to-be for the fulfillment of the goal acquire data and of the softgoal have a spatial representation.

The system actor is now in charge of consulting the maps owned by the Land Registry and to acquire the orchards data from the Producer; these two relationships are modeled with two dependencies ---also in bold---. As a consequence the direct dependencies between the Advisor, the Producer and Land Registry, for the orchards data and for the maps --- in gray in the figure--- are no more critical dependencies.

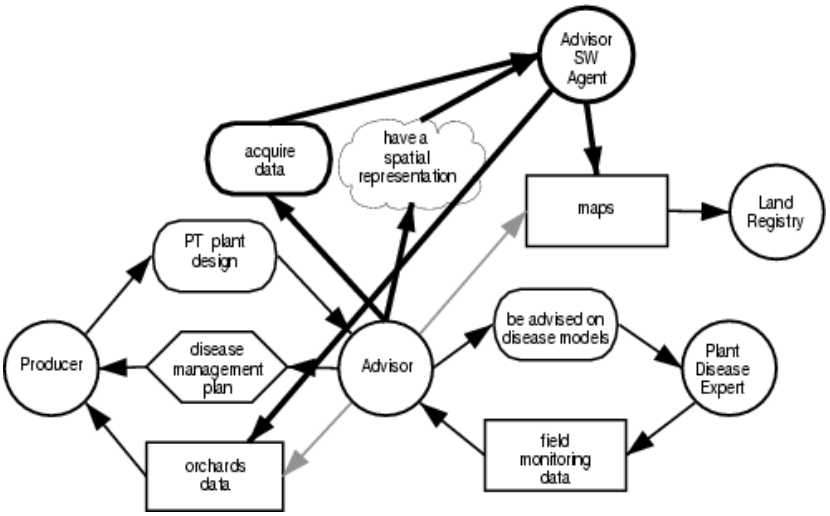

Figure 6: Late requirements actor diagram showing how the IP domain strategic dependencies will be affected by the introduction of the DSS (bold dependencies).

6. The transition from requirements to design

Approaching system design we face several questions, which are typical of the design of complex systems [19]. Among them: how can we identify system components which ensure an appropriate level of cohesion? How can we reduce coupling among components? Is there any architectural styles that we can exploit in an effective way? This last issue has been deeply analyzed, in the context of *Tropos*, in [21][22]. Here we focus on the first questions.

In our approach, the late requirements goal analysis provides the basis for system architecting and designing. In fact, Architectural Design (AD) in *Tropos* aims at defining a macro-level view of the system architecture, in terms of components (modeled as sub-actors), and interfaces between components (specified in terms of dependencies), which results following to a top-down decomposition strategy.

In a sense, goal analysis applied to the system-to-be actor of the LR model provides a method for implementing a *divide-and-conquer* strategy in software architecting. The resulting AD model, which is depicted in Figure 7 includes a set of sub-actors which will take care of goals and plans resulting from the goal analysis of the system's goals.

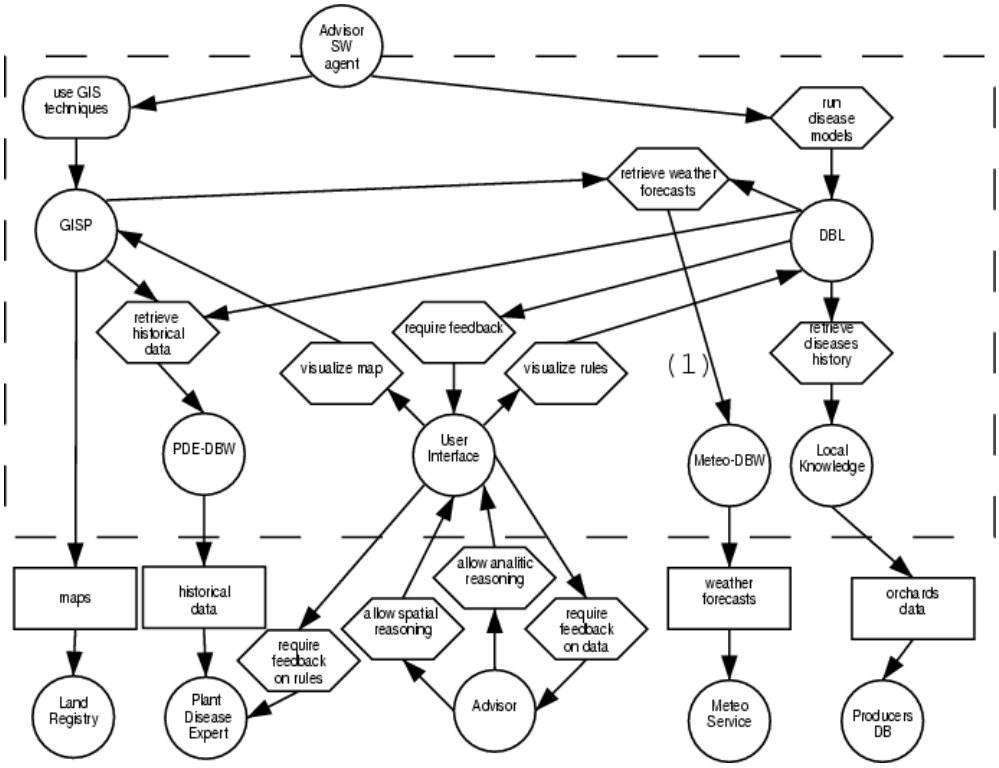

Figure 7: Architectural Design. The actor diagram refined upon system subgoals.

In particular, the actor GISP (Geographic Information Services Provider) to which the Advisor SW agent delegates the goal use GIS techniques; the actor DBL (Disease Behavior Learner), which performs the plan run disease models on the basis of information extracted from the seasonal data on the disease; three wrapper actors, namely, the PDE-DBW (Plant Diseases Expert DB Wrapper) which takes care of retrieving meteo and orchard historical data; the wrapper of the database of the meteo service, called Meteo-DBW (Meteo Service DataBase Wrapper) which retrieves weather forecast; the Local Knowledge actor, which is the wrapper of the local data base containing data relative to the orchards belonging to the area under the advisor control (represented by the actor Producers DB in Figure 7); the actor User Interface which manages the interaction between the user of the application (represented by the actor Advisor in the ER model) and the other specialized system actors. Relationships between subsystems are specified in terms of plan dependencies. For instance, the advisor

involved in the study of a disease that needs to run the model describing the population dynamic, requires the actor User Interface for the execution of the plan visualize rules; as a consequence a new interaction between the User Interface and DBL is needed, devoted to the running of the disease model in order to obtain the set of rules induced by the meteo and orchards status data; this data can be retrieved by the DBL by means of the plan dependencies retrieve weather forecasts and retrieve disease history among DBL and the actors Meteo-DBW and Local Knowledge respectively. The resulting architecture satisfies also principles of *cohesion*, in the sense that similar services have been grouped into a single actor. For instance, the actor User Interface collects all the user interface functionalities both for DBL actor and GISP actor. Moreover, *coupling* among components, here represented in terms of the binary dependencies between actors, is also minimized.

7. The DSS system

A prototype of the DSS system has been developed and evaluated by the technicians and researchers of the Advisory Service providing useful suggestions for its improvement[1].
The system has been implemented as a set of JavaScript and HTML pages (client side), while the server components, implemented in Java, exploit a Tomcat servlet container and a PostGreSQL data base.
In particular, the GISP component is based on a set of Geographic Information System (GIS) functionalities that allow to visualize territorial data and to perform spatial queries relatively to the apple orchards in the Trentino area. In particular, we have developed a set of functions supporting the design of a pheromones sex trapping (PT) plant on a multi-orchard basis. Figure 8, depicts the browser based Graphical User Interface of this component. The visualization area is subdivided in three major areas: in the center is depicted a map of the area of interest showing the organizational setting of the orchards;

[1] See [23] for a description of the requirement analysis of the DSS and [24] for a complete description of the DSS, its detailed design and the users it supports.

on the left, a set of functions allow the user to interact with the map; on the right, the user can find a set of functions related to the management of a PT plant.

The DSS supports the advisor while performing the following decision-making tasks:

- assessing the risk of a disease infestation on a given orchard, taking into account the environmental parameters of the area the orchard belongs to, the features of the culture and the historical data;
- designing a pheromone trapping system using geometrical features of a given orchard and information on the presence of infection sources, diffusion barriers, etc.;
- identifying alerting events that need to be communicated to the producers in order to advise them during the execution of a disease management plan.

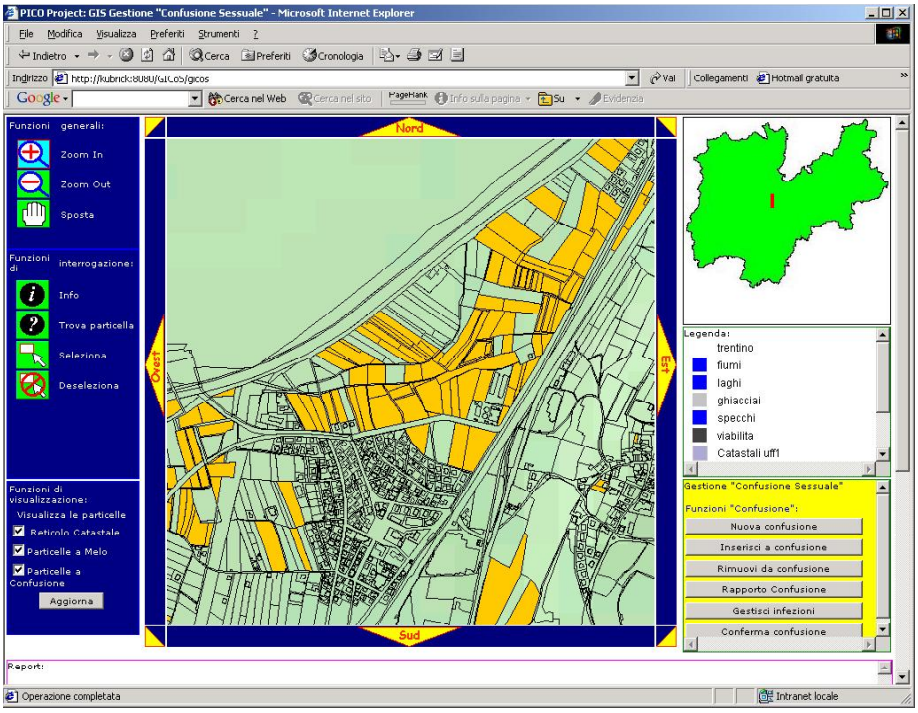

Figure 8: A snapshot of the GUI of the system composed of three major panels: in the center is depicted a map of the area of interest showing the production; on the left, a set of interactive functions for browsing the map; on the right, a set of functions related to PT system design.

8. Related work

Several research lines are interesting for the work presented in this chapter. Here we will mention two of them: first, works that discuss basic issues in building effective DSS for environmental problems; second, approaches aiming at narrowing the semantic gap between a requirement specification and the software architecture to be produced from it.

Along the first line, the need of a deep domain analysis aiming at pointing out the organizational dimension in environmental management has been largely discussed together with the need for involving domain stakeholders in requirements identification and system design (see work proposing participatory design for environmental DSS [17]). Previous experiences in developing DSS for environmental management problems, mentioned in section 1, supported analogous considerations [3][4][5][7][11][12].

Among the works pertaining to the second research line, we shall mention the proposal of a design approach based on a family of pattern (architectural style), in the context of the *Tropos* effort [21][22]. This work is rooted in organizational theory which provides business organization models. The authors propose to use these models as software architectural styles (specified in i* notation [28]) for MAS and evaluate them respect to software quality attributes.

An approach to component-based design that resembles several analogies with our work, is that presented in [20]. Here, design issues related to component identification and specification are put in relation with a parallel analysis of both business goals and processes, which provides with the identification of enterprise components. Component specification is given in UML notation (class diagram). In a sense, Responsibility-driven design [26] can be considered an approach which bears analogies with our, but adopting an object-oriented point of view. Instead of considering actors' goals, here, class responsibilities are analyzed, together with collaborations among classes, which are devoted to fulfill responsibilities.

9. Conclusion

This chapter described the early phases of the development of a decision support system, for the agriculture Advisory Service of our region which have been performed using *Tropos*, an agent oriented software engineering methodology that allows to model explicitly the domain stakeholders with their goals and the mutual dependencies.

We discussed the early requirement and late requirement analysis specifying the reasons for dependencies between social and system actors. In particular, the DSS requirements have been derived in terms of goal and plan delegations from social actors (the users) to system actors.

A sketch of the architectural design, according to the *Tropos* methodology has been also given. The architecture includes a set of software components (agents) wrapping existing information systems and interacting with agents providing estimates on the evolution of a specific plant disease. The resulting DSS has been briefly described.

References

[1] David Aha and Jody J. Daniels, editors. *Case-Based Reasoning Integrations*, number WS-98-15. The AAAI Press, Jul 1998. Madison Wisconsin.

[2] W. Allen, O. Bosch, M. Kilvington, J. Oliver, and M. Gilbert. Benefits of collaborative learning for environmental management: Applying the integrated systems for knowledge management approach to support animal pest control. *Environmental Management*, 27(2):215–223, 2001.

[3] 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.

[4] P. Avesani F. Ricci and A. Perini. Interactive case-based planning for forest fire management. *Applied Intelligence Journal*, 4(13):41–57, 2000.

[5] Paolo Avesani and Emanuele Olivetti. Active Sampling for Data Mining. In Prastacos P., Cortes U., Diaz de Leon J. L., and Murillo M., editors, *e-Environment: Progress and Challenges*, Ist. Politecnico National, Mexico, 2004.

[6] C.A.L. Bailer-Jones and D.J.C. MacKay. A Recurrent Neural Network for Modelling Dynamical Systems. *Computation in Neural System*, 9:531–547, 1998.

[7] L. K. Branting, J. D. Hastings, and J. A. Lockwood. Integrating Cases and Models for Prediction in Biological Systems. AI Applications, 11(1):29–48, 1997.

[8] L. K. Branting, J. D. Hastings, and J. A. Lockwood. CARMA: A Case-Based Range Management Advisor. In *Proceedings of The Thirteenth Innovative Applications of Artificial Intelligence Conference (IAAI-2001)*, Seattle, Washington, USA, August 2001.

[9] P. Bresciani, P. Giorgini, and F. Giunchiglia, J. Mylopoulos, A. Perini. Tropos: An Agent-Oriented Software Development Methodology. In *International Journal of Autonomous Agents and Multi Agent Systems,* 8(3):203–236, May 2004.

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

[11] U. Cortes, I. Rodrıguez-Roda, M. Sanchez-Marre, J. Comas, C. Cortes, and M. Poch. DAI-DEPUR: An Environmental Decision Support System for control and supervision of Municipal Waste Water Treatment Plants. In IOS Press, editor, *Proceedings of European Conference on Artificial Intelligence (ECAI 2002)*, Lyon, France, July 2002.

[12] U. Cortes, M. Sanchez-Marre, R. Sanguesa, J. Comas, I. Rodrıguez-Roda, M. Poch, and D. Riano. Knowledge management in environmental decision support systems. *AI Commun.*, 14(3), IOS press, 2001.

[13] R. Denzer. Generic integration in environmental information and decision support systems. In Andrea E. Rizzoli and Anthony J. Jakeman, editors, *Integrated Assessment and Decision Support*, pages 53 – 60. iEMSs, June 2002.

[14] A. Gerevini, A. Perini, F. Ricci, D. Forti, C. Ioriatti, and L.Mattedi. Pomi: An expert system for integrated pest management of apple orchards. *AI Applications*, (6):51–62, 1992.

[15] F. Giunchiglia, J. Mylopoulos, and A. Perini. The Tropos Software Development Methodology: Processes, Models and Diagrams. In Giunchiglia et al. [21].

[16] F. Giunchiglia, J. Odell, and G. Weiß, editors. *Agent-Oriented Software Engineering III*. LNCS. Springer-Verlag, Bologna, Italy, Third International Workshop, AOSE2002 edition, 2002.

[17] M. Hare, R. A. Lechter, and A. I. Jakeman. Participatory natural resource management: A comparison of four case studies. In Andrea E. Rizzoli and Anthony J. Jakeman, editors, *Integrated Assessment and Decision Support, Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society*, volume 1, pages 73–79. iEMSs, June 2002.

[18] David C.-L. Lam and David A. Swayne. Issues of eis software design: some lessons learned in the past decade. *Environmental Modelling and Software*, 16(5):419–425, 2001

[19] T. C. Lethbridge and R. Laganiere. Object-Oriented Software Engineering. McGraw-Hill, 2001.

[20] K. Levi and A. Arsanjani. A goal driven approach to enterprise component identification and specification. *Communications of the ACM*, 45(10), October 2002.

[21] J. Mylopoulos, M. Kolp, and P. Giorgini. Agent-Oriented Software Engineering. In *Proceedings of the 2nd Hellenic Conference on Artificial Intelligence (SETN-02)*. 2002.

[22] M. Kolp and J. Mylopoulos. Software Architecture as OrganizationalStructures. In *Proceedings of ASERC workshop*. Edmonton, Canada, August 2001.

[23] A. Perini and A. Susi. Designing a Decision Support System or Integrated Production in Agriculture. An Agent-Oriented approach. *Environmental Modelling and Software Journal*, 19(9), September 2004

[24] Anna Perini and Angelo Susi. AI in support of Plant Disease Management. AICommun. 18(4), pp. 281–291, IOS press, 2005.

[25] D.N. Stones R.E. Plant. Knowledge-Based Systems in Agriculture. McGraw-Hill Inc., 1991.

[26] R. Wirfs-Brock and A. McKean. Object Design: Roles, Responsibilities, and Collaborations. Pearson Education, 2002.

[27] M.J. Wooldridge, G. Wei , and P. Ciancarini, editors. Agent-Oriented Software Engineering II. LNCS 2222. Springer-Verlag, Montreal, Canada, Second International Workshop, AOSE2001 edition, May 2001.

[28] E. Yu. Modeling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto, 1995.