ITC irst

CENTRO PER LA RICERCA
SCIENTIFICA E TECNOLOGICA

38050 Povo (Trento), Italy
Tel.: +39 0461 314312
Fax: +39 0461 302040
e−mail: prdoc@itc.it − url: http://www.itc.it

COORDINATION SPECIFICATION IN MULTI−AGENT SYSTEMS. FROM REQUIREMENTS TO ARCHITECTURE WITH THE TROPOS METHODOLOGY

Perini A., Susi A., Giunchiglia F.

July 2002

Technical Report # 0207−20

Coordination specification in Multi-Agent Systems. From requirements to architecture with the Tropos methodology

Anna Perini
ITC-Irst
Via Sommarive, 18
I-38050 Trento-Povo, Italy
perini@irst.itc.it

Angelo Susi
ITC-Irst
Via Sommarive, 18
I-38050 Trento-Povo, Italy
susi@irst.itc.it

Fausto Giunchiglia
DIT - University of Trento
via Sommarive, 14
I-38050 Trento-Povo, Italy
fausto@dit.unitn.it

ABSTRACT

The goal of this paper is to propose a new methodology for designing coordination between human agents and software agents and, ultimately, among software agents. The methodology is based on two key ideas. The first is that coordination should be designed in steps, according to a precise software engineering methodology, and starting from the specification of early requirements. The second is that coordination should be modeled as dependency between actors. Two actors may depend on one another because they want to achieve goals, acquire resources or execute a plan. The methodology used is based on *Tropos*, an agent oriented software engineering methodology presented in earlier papers. The methodology is presented with the help of a case study.

1. INTRODUCTION

Agents, either human or software, are social entities that interact, by nature. Coordination among agents is considered a form of interaction devoted to goal attainment and to task completion. Moreover, coordination processes support contrasting agents behaviors, from cooperation to competition. Cooperating agents work together to achieve a common goal, trying to accomplish them as a team. Competitive agents work against each other, trying to optimize their own benefit, because their goals are conflicting.

Building effective Multi-Agent Systems (MASs), requires to carefully design and implement coordination processes. This motivates the large interest on this topic, as emerging from the last ten years MAS literature. Agents coordination has been studied from different perspectives, i.e. considering (software) agents coordination at run-time [4, 2], (software) agents coordination at the detailed design level [7, 8] (i.e. designing the *micro* level) and agents coordination at the social level [3, 8] (i.e. designing the *macro* level). At the macro level, both human and software agents can be considered.

SEKE 2002 , Ischia, Italy IT

We think that, in order to build effective MAS that operate into human communities, interacting both with software and human agents, we first need to model coordination processes taking place into the social organizational setting where the MAS has to be introduced. Then, we have to analyze how these coordination processes will be affected by introducing a MAS (analogously to what is done during the *macro* level analysis for heterogeneous systems). Only in the following steps we keep designing coordination processes among software agents and detailing interaction and communication mechanisms which support the required coordination processes. This multi-steps process allow us to keep a trace of the *why* (i.e. the needs) of the coordination processes modeled at the *micro* level.

Coordination specifications should rest on a deep analysis of the agent intentional dependencies which can be modeled in terms of goal, plan or resource dependencies between pair of agents.

In our approach we adopt the *Tropos* methodology which offers concepts and techniques at support of this idea. The *Tropos* methodology [14, 10] is an agent oriented software development methodology that can be characterized by the following features, namely: the notion of agent, goal, plan and various other knowledge level notions are used in all phases of software development, defining a *knowledge level* [11] approach to requirement specification and design activities; a crucial role is given to the very early phase of requirements specification when the environment and the system-to-be are analyzed, according to a *requirement driven* approach [10].

The methodology rests on the idea of building a conceptual model that is incrementally refined and extended from an early requirement model, in which the organizational setting where the MAS will be introduced is analyzed, to executable artifacts, along five main development phases, called respectively: *early requirement analysis*, *late requirements analysis*, *architectural design*, *detailed design*, *implementation*.

The paper is structured as follows. Section 2 briefly describes the Tropos methodology and specifies the concept of dependency in Tropos. Sections 3.1 and 3.2 describe the results of modeling actor coordination during early and late requirements analysis in Tropos, as applied in the context of a real application devoted to develop a decision support for the technicians of the agricultural advisory service when managing plant diseases. Sections 3.3 describes the initial steps in the architectural design. The related work are con-

sidered in Section 4. Finally, conclusions and the future work are presented in Section 5.

2. THE METHODOLOGY

The core process of the *Tropos* methodology consists in performing conceptual modeling using a visual language that provides: intentional and social concepts such as actor, goal and dependency; a graphical notation which allows to visualize a model; a set of diagrams that allow to build views on different properties of the model: *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. Examples are given in Section 3. In actor diagrams, actors are depicted as circles, their goals as ovals and their softgoal as cloud shapes. A dependency relationship between two actors is depicted as two arrowed lines connected by a graphical symbol varying according to the dependum: a goal, a plan or a resource. A goal diagram depicts goal and plan analyses performed from the perspective of a specific actor in a balloon that contains graphs whose nodes are goals (ovals) and /or plans (hexagonal shape) and whose arcs represents the different types of relationships that can be identified between its nodes.

Conceptual modeling drives the software development process in Tropos along five main phases: *Early Requirements*, *Late requirements*, *Architectural design*, *Detailed design*, *Implementation*.

A few remarks about actor dependency are worth to be mentioned. An actor called the dependee, may depend on another actor, the depender in order to attain some goal, execute some plan, exploit a resource. Through *goal dependency* a goal is delegated by the depender actor to the dependee who will decide autonomously how to satisfy the goal. As a consequence, the depender becomes vulnerable respect to the dependee for the goal satisfaction. *Plan dependency* specifies that the depender requests the dependee to execute a specific plan. The execution control is delegated to the dependee, the depender is still vulnerable because he/she rests on the plan outcomes for reaching (avoiding) desirable (undesirable) states of affairs. *Resource dependency* model the request of the depender to the dependee of a specific entity (resource). The way this resource should be delivered by the dependee is not specified. (It can eventually be specified by additional softgoal dependencies).

3. AN EXAMPLE

The example in this paper refers to the agricultural domain, in particular to the problem of favoring the application of Integrated Production (IP) practices, which exploit low environmental impact techniques, by the apple producers [13]. In the rest of the section we shall focus on the early requirement model of the IP domain and on the late requirements model. A sketch of the architectural design will be then presented.

3.1 Early Requirements

> *Early Requirements focuses on the coordination processes between domain stakeholders; they are modeled as a set of dependencies between pairs of actors.*

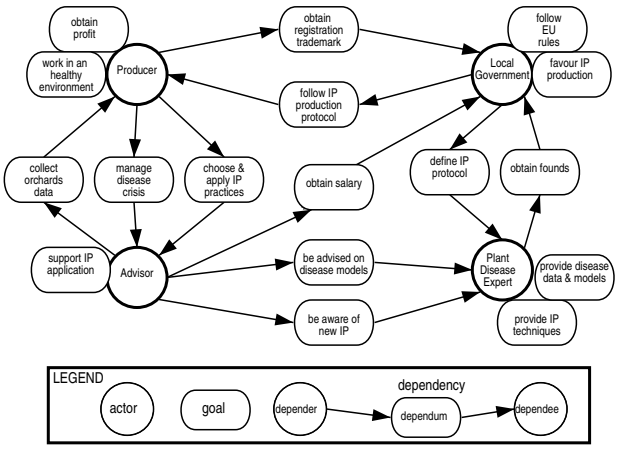

Figure 1: Early requirements model. A portion of the actor diagram modeling the IP organizational setting.

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 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 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. Coordination is modeled in terms of goal dependencies between the actors.

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 car-

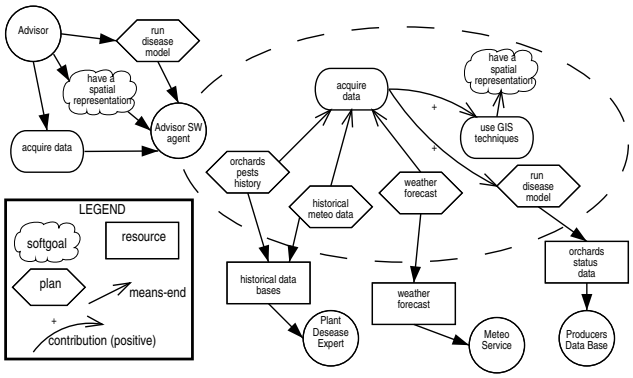

Figure 2: Late requirements model. The Advisor SW Agent goal diagram.

rying up apple production according to the IP rule. So the actors Advisor and Producer closely coordinates: 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). 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 all its actors and by analyzing their goals. In particular the analysis of the actor Advisor proceedes exploiting its goal diagram[1], that describes the advisors internal goals generated from the dependencies shown in the actors diagram in Figure 1. During this phase new coordination processes can be discovered in the domain and new actors and dependences can be added in the model.

3.2 Late Requirements

> *Late Requirements analysis focuses on the system-to-be actor and on the coordination processes between it and the human actors.*

During late requirements analysis the system-to-be, that is the decision support system at use of the advisors when dealing plant disease management, is introduced as a new actor into the conceptual model. Figure 2 depicts a fragment of the late requirements model where the actor Advisor SW Agent models the system-to-be. In particular, the actor Advisor delegates the system-to-be for the fulfillment of the

[1]For a complete description of the Advisor goal diagram see also [15].

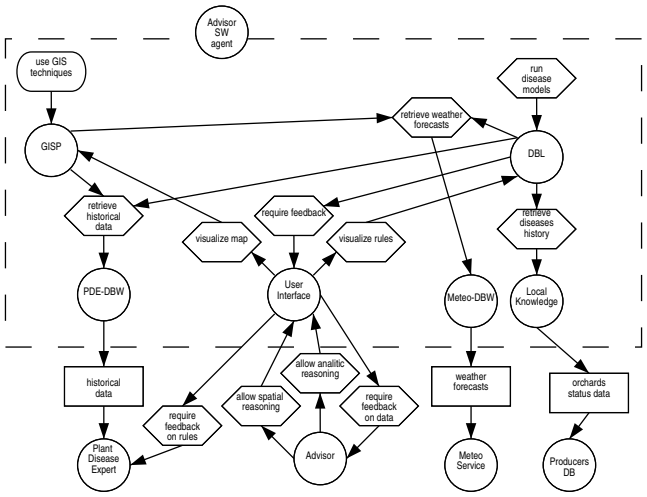

Figure 3: Architectural Design - step1. The actor diagram refined upon system sub-goals delegation.

goal acquire data and of the softgoal have a spatial representation, and for the execution of the plan run disease models, found during Early Requirements analysis. 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.

3.3 Architectural Design

> *Architectural design defines the system's global architecture in terms of subsystems represented as actors. Dependencies between subactors describe the coordination processes, between the system components.*

A portion of the system architecture model is illustrated in the actor diagram depicted in Figure 3. 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 2). 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), 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 Experts 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 3). The actor User Interface which manages the interaction between the user of the application (the actor Advisor) and the other specialized sub-actors of the Advisor SW agent.

The system architecture model can be further enriched with other system actors according to design patterns (as

explained in [6]) 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 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 [12] 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.

4. RELATED WORK

As already mentioned, agents coordination has been largely recognized as a critical issue in both MAS and Distributed Artificial Intelligence [17]. As a consequence, several interesting approaches for studying this topic, from different perspectives, can be found in the literature on MAS.

Work dealing with agents coordination at run-time focus basically on interaction and communication protocols that effectively support coordination processes see for instance [16, 7]. At a more theoretical level, the coordination problem at run-time has been faced using dynamic programming strategies, see for instance [4, 2, 1]. A relevant approach to the design of agents coordination processes is [8], where commitments and conventions mechanisms are used to specify coordination processes. Analyses of agents coordination based on an explicit model of the dependencies among agents actions have been proposed in [3].

Finally, work in Computer Supported Cooperative Work [5], provides useful ideas for dealing with the problem of designing MAS coordination, especially when heterogeneous agents, human and software, are considered. In particular, the work of Malone [9] is worth to be mentioned. Malone considers coordination as a phenomenon that occurs in different kinds of systems (e.g. human, computational, biological) and this allows to set up a framework for studying coordination which exploits analysis techniques provided by different disciplines, such as economics, computer science organizational theory. A definition of coordination, as the process of *managing dependencies between activities* has been defined and a research agenda on this topic is proposed.

5. CONCLUSION AND FUTURE WORK

This paper describes a new methodology for designing coordination between human agents and software agents based on *Tropos*, an agent oriented software engineering methodology. The approach rests on the basic idea that coordination can be modeled as dependencies between actors.

Examples of coordination modeling have been given, with reference to a real application for the agriculture domain which is currently being developed in our group (early requirements, late requirements and a brief sketch of the architectural design has been presented). Next step consist in focus on detailed design and implementation phases combining our methodology with others, for instance those discussed in Section 4.

6. REFERENCES

[1] R. A. Bourne, C. Excelente-Toledo, and N. R. Jennings. Run-time selection of coordination mechanisms in multi-agent systems. In *Proc. of ECAI'2000*, pages 348–352. IOS Press, 2000.

[2] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, pages 478–485, 1999.

[3] C. Castelfranchi. Modeling Social Action for AI Agents. In *IJCAI*, pages 1567–1576, 1997.

[4] E. H. Durfee. Practically Coordinating. *AI Magazine*, 20(1), 1999.

[5] C. Ellis and J. Wainer. *Groupware and Computer Supported Cooperative Work*, chapter 10. In Weiss [17], 1999.

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

[7] M. N. Huhns and L. M. Stephens. *Multiagent systems and Societies of agents*, chapter 2. In Weiss [17], 1999.

[8] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[9] Thomas W. Malone and Kevin Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[10] J. Mylopoulos and J. Castro. Tropos: A framework for requirements-driven software development. In J. Brinkkemper and A. Solvberg, editors, *Information System Engineering: State of the Art and Research Themes*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[11] A. Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.

[12] J. Odell, H. Parunak, and B. Bauer. Extending UML for agents. In G. Wagner, Y. Lesperance, and E. Yu, editors, *Proc. of the AOIS 2000 workshop at the 17th National conference on Artificial Intelligence*, pages 3–17, Austin, TX, 2000.

[13] A. Perini. A web advisor for integrated protection. In *ECAI2000 WS AI in Agriculture and Nat.Res.*, 2000.

[14] A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, and J. Mylopoulos. A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. In *Proceedings of Agents 2001*, Montreal CA, May 2001. ACM.

[15] A. Perini, A. Susi, and F. Giunchiglia. Coordination specification in Multi-Agent Systems. from requirements to architecture with the Tropos methodology. Technical report, ITC-IRST, April 2002.

[16] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In *IEEE Transactions on Computers*, volume C-29, pages 1104–1113, 1980.

[17] G. Weiss, editor. *Multiagent System: a modern approach to Distributed AI*. MIT Press, 1999.