An Agent-Oriented Meta-model for Enterprise Modelling

Ivan Jureta and Stéphane Faulkner

Information Management Research Unit,
University of Namur,
8 Rempart de la vierge, B-5000 Namur, Belgium
{ivan.jureta, stephane.faulkner}@fundp.ac.be

Abstract. This paper proposes an agent-oriented meta-model that provides rigorous concepts for conducting enterprise modelling. The aim is to allow analysts to produce an enterprise model that precisely captures the knowledge of an organization and of its business processes so that an agent-oriented requirements specification of the system-to-be and its operational corporate environment can be derived from it. To this end, the model identifies constructs that enable capturing the intrinsic characteristics of an agent system such as autonomy, intentionality, sociality, identity and boundary, or rational self-interest; an agent being an organizational actor and/or a software component. Such an approach of the concept of agent allows the analyst to have a holistic perspective integrating human and organizational aspects to gain better understanding of business system inner and outer modelling issues. The meta-model takes roots in both management theory and requirements engineering. It helps bridging the gap between enterprise and requirements models proposing an integrated framework, comprehensive and expressive to both managers and software (requirements) engineers.

1 Introduction

Business analysts and IT managers have advocated these last fifteen years the use of enterprise models to specify the organizational and operational environment (outer aspects of the system) in which a corporate software will be deployed (inner aspects of the system) [20]. Such a model is a representation of the knowledge an organization has about itself or of what it would like this knowledge to be. This covers knowledge about functional aspects of operations which describe what and how business processes are to be carried and in what order; informational aspects that describe what objects are to be processed; resource aspects that describe what or who performs these processes according to what policy; organizational aspects that describe the organization architecture within which processes are to be carried out ; and finally, strategic aspects that describe why processes must be carried out. The specification of these key aspects of the core business of an enterprise is an effective tool to consider for gathering and eliciting software requirements. It may be used to [1, 4]:

- analyze the current organizational structure and business processes in order to reveal problems and opportunities;
- evaluate and compare alternative processes and structures;

J. Akoka et al. (Eds.): ER Workshops 2005, LNCS 3770, pp. 151 – 161, 2005.

– achieve common understanding and agreement between stakeholders (e.g., managers, owners, workers, etc.) about different aspects of the organization;
– reuse knowledge available in the organization.

This paper proposes an integrated agent-oriented meta-model for enterprise modelling. The agent paradigm is a recent approach in software engineering that allows developers to handle the life cycle of complex distributed and open systems required to offer open and dynamic capabilities in the latest generation enterprise software (see e.g., [22]).

The proposed meta-model takes inspiration from research works in requirements engineering frameworks (see e.g., [3, 5]), management theory concepts found to be relevant for enterprise modelling (see e.g., [11, 12, 13]) and agent oriented software engineering (see e.g., [22]). It leads to reduce the semantic gap between enterprise and requirements representations, providing a modelling tool that integrates the outer specification of the system together with its inner specification. Our proposal implicitly suggests a holistic approach to integrate human and organizational issues and gain better understanding of business processes and organisations representation. To this end, we introduce new concepts to enterprise modelling, related to authority, power and interest.

The rest of this paper is organized as follows. Section 2 describes the main concepts of our meta-model. Sections 3 and 4 detail some elements of the meta-model and discuss their relevance for enterprise modelling. Section 5 gives an overview of related works and Section 6 summarizes the results and points to further work.

2 An Agent-Oriented Enterprise Meta-model

The motivation of our proposal is to understand precisely the semantics of the organizational environment of the system and to produce an agent-oriented requirements specification for the software to build. The framework described in this section provides modelling constructs that enable the representation of the autonomy, intentionality, sociality, identity and boundary, and rational self-interest of actors, i.e., agents in the real world and/or software agents. Actors are autonomous as their behaviour is not prescribed and varies according to their dependencies, personal goals and capabilities. They are intentional since they base their actions and plans on beliefs about the environment, as well as on goals they have to achieve. Being autonomous, actors can exhibit cooperative behaviour, resulting from similar goals and/or reciprocal dependencies concerning organizational roles they assume. The dependencies can either be direct or mediated by other organizational roles. Actors can have competing goals which lead to conflicts that may result from competing use of resources. Actors have varying power and interest in the ways in which organizational goals contribute to their personal ones. Boundary and identity are closely related to power and interest of actors. We model variations in boundary and identity as resulting from changes in power and interest since these vary with respect to the modifications in the roles an actor assumes and the dependencies involving these roles. Actors can act according to their self-interest, as they have personal goals to achieve. They have varying degrees of motivation to assume organizational roles,

according to the degree of contribution to personal goals these roles have in achieving organizational ones. Actors apply plans according to the rationale described in terms of personal, organizational goals, and capabilities. The rationale of our actors is not perfect, but bounded [10, 15], as they can act based on beliefs that are incomplete and/or inconsistent with reality. We provide constructs such as AndOr relationships, non-functional requirements [22]… to evaluate alternative deployments of the software in the organizational environment.

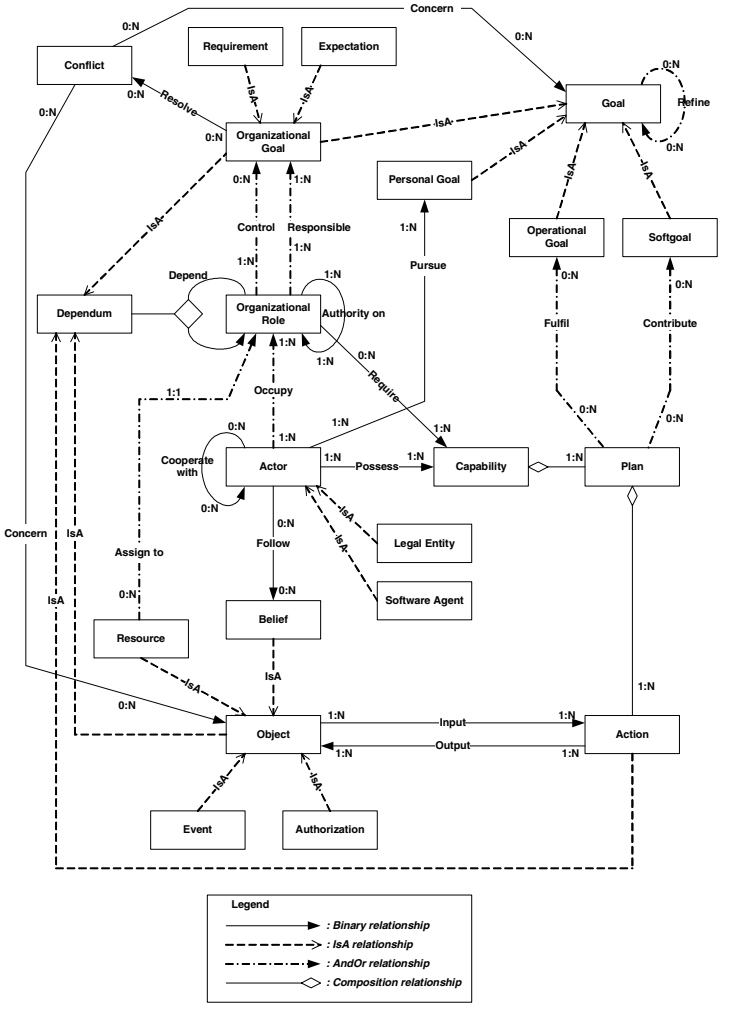

Fig. 1. The agent-oriented meta-model

Fig. 1 introduces the main entities and relationships of our meta-model. For clarity, we have subdivided it into five sub-models:

- *Organizational sub-model*, describing the actors of the organization, their organizational roles, responsibilities and capabilities.
- *Goals sub-model*, describing enterprise and business process purposes, i.e. what the actors are trying to achieve and why.
- *Conflict sub-model*, indicating inconsistencies in the business process.
- *Process sub-model*, describing how actors achieve or intend to achieve goals.
- *Objects sub-model*, describing non-intentional entities and assumptions about the environment of the organization and the business processes.

Due to a lack of place, the paper only details the organizational and goal sub-models, their integration and discusses their relevance for enterprise modelling. We first sketch the meta-model from the point of view of system developers and of organization managers.

2.1 Information System Development Perspective

The meta-model provides widely-used constructs for specifying the architecture of an agent-oriented information system: *Actors* are agents of the system. They *possess Capabilities* composed of *Plans*, each *Plan* representing a sequence of atomic *Actions*. When applying *Plans*, *Actors fulfil* or *contribute* to system *Goals*. *Actors follow Beliefs* which represent assertions about aspects of the organization and/or its environment. *Actions* can take *Objects* as *input* from the system or its environment. New *Objects* can be produced or existing ones modified by carrying out *Actions*, i.e., they can be *output* from *Actions*. *Objects* represent any thing of interest for the system: *Resources*, *Beliefs*, *Authorizations* or *Events.*

2.2 Management Perspective

The meta-model provides common terms used to describe an organization. *Organizational Roles* are *responsible* of *Organizational Goals*, which may be either *Operational* (i.e. can be actually fulfilled) or *Softgoals* (such as e.g., broadly specified business objectives). *Organizational Roles* can *depend* on one another for the provision of *Dependums* - *Actions*, *Objects*, or *Organizational Goals*. An *Actor*, being a *Legal Entity* or a *Software Agent*, can *occupy Organizational Roles*, as long as it *possesses* the *required Capabilities* to do so. *Actors* exhibit intentional behaviour as they act according to *Goals* and *Beliefs* about their environment. As *Beliefs* may be incoherent, and as they pursue *Personal Goals*, *Actors* can exhibit competitive behaviour. They will exhibit cooperative behaviour when they are responsible of identical *Organizational Goals*. *Actors* execute *Plans*, composed of *Actions*, in order to fulfil and contribute to *Goals*. By doing so, they comply with the responsibilities of *Organizational Roles* they *occupy*. As a matter of organizational policy, *Resources* in the organization are assigned to *Organizational Roles*. The allocation of Resources is determined by both *authority* among *Organizational Roles* and *Authorizations* that may be *input* or *output* of specific *Actions*.

Common ground between both points of view resides in the sense that the information system can be developed to automate some (part of) business processes (e.g., administrative tasks) or to radically modify ways in which *Goals* are fulfilled (e.g., reorganizing customer relationship management by deploying e-commerce

facilities). The model provides an unambiguous representation serving both software staff and organization strategic management.

Primitives of our framework are of different types: meta-concepts (*Goal*, *Actor*, *Object*, etc.), meta-relationships (*possess*, *require*, *pursue*, etc.), meta-attributes (*Power*, *Interest*, *Motivation*, etc.), and meta-constraints (e.g., "*an actor occupies a position if that actor possesses all the capabilities required to occupy it*").

All meta-concepts, meta-relationships and meta-constraints have the following mandatory meta-attributes:

- *Name*, which allows unambiguous reference to the instance of the meta-concept (e.g. "European Commission" for the Actor meta-concept).
- *Description*, which is a precise and unambiguous description of the corresponding instance of the meta-concept. The description should contain sufficient information so that a formal specification can be derived for use in requirements specifications for a future information system.

3 Organizational Sub-model

The Organizational sub-model is used to identify the relevant *Actors* of the organization, the *Organizational Roles* they *occupy*, the *Capabilities* they *possess*, and the *Dependums* for which *Actors depend on* one another.

3.1 Actor

An Actor applies *Plans* (which are part of his *Capabilities*) to *fulfil* and/or *contribute* to *Organizational Goals* for which the *Organizational Role* he *occupies* is *responsible*, and *Personal Goals* he *pursues* (i.e. wishes to achieve). As the *Actor* exists in a changing environment, it *follows Beliefs* about the environment in order to adapt its behaviour to environmental circumstances.

An *Actor* is either a *Legal Entity* or a *Software Agent*. A *Legal Entity* is used to represent any person, group of people, organizational units or other organizations that are significant to the organization we are modelling, i.e., that have an influence on its resources, its goals, etc. A *Software Agent* is used to represent a software component of an information system(-to-be). An *Actor* can *cooperate with* another *Actor* to fulfil and/or contribute to *Organizational Goals* common to the *Organizational Roles* that each of these *Actors occupies*.

Besides standard meta-attributes, an *Actor* possesses the *Motivation* meta-attribute, whose values describe the degree of motivation of an *Actor* to *occupy* an *Organizational Role*. Values are functions of the degree of contribution to *Personal Goals* the Actor's *Organizational Role* have in achieving *Organizational Goals* and of functions of the conflicts involving this *Actor*.

A *Legal Entity* is characterized with two specific meta-attributes: *Interest* and *Power* [11]. *Interest* is the degree of satisfaction of an actor to see *Organizational Goals* positively contributing to its *Personal Goals*. *Power* is the degree to which the actor is able to modify the objectives of the organization or its business processes through its *Capabilities*. For instance, when automating a business process, the values of *Interest* and *Power* meta-attributes of *Legal Entities* change: in the new

configuration of the process, some actors will gain decision power while maintaining the same level of interest; others that previously benefited from high power in the initial process structure might become less powerful. It is crucial to take these changes into account when eliciting software requirements. It may lead otherwise to introducing *Goals* not identified during the initial requirements analysis, and/or changing *Priority* of already specified *Goals*. *Interest* and *Power* help to find *Legal Entities* that will play a crucial role in the software-to-be. For example, focus in some business process might shift to *Legal Entities* which were not considered very significant during the inception phase, and whose needs were not specified in depth. This would result in that these now crucial processes would not be fully exploited, and would lead to the overall failure of the requirements specification efforts.

3.2 Organizational Role

Actors occupy Organizational Roles. They can take many forms: a unique functional position (e.g. the Project Manager), a unique functional group (e.g. the Marketing Department), a rank or job title (e.g. the CIO), a class of persons (e.g. Customer), etc. *Organizational Roles* are responsible of *Organizational Goals*. They cannot be responsible of *Personal Goals*. Each *Organizational Role* requires a set of *Capabilities* which can be used to fulfil or contribute to *Organizational Goals* for which it is responsible. *Organizational Roles* can be attributed only to those *Actors* that possess all the capabilities required to occupy these *Organizational Roles*.

Organizational Roles can have different levels of authority. Consequently, an *Organizational Role* can have *authority on* another *Organizational Role*. The *authority on* meta-relationship specifies the hierarchical structure of the organization.

3.3 Capability

A *Capability* is a set of *Plans* an *Actor* can execute. An *Actor possesses Capabilities*.

When exploring possible alternative business processes or organizational structures, newly identified *Organizational Roles* can *require Capabilities* that no *Actor possess*. These *Capabilities* have to be confronted to those available in the organization, in order to evaluate the proposed alternatives with respect to the current *Roles* and the way they use existing *Capabilities*. This is significant to determine which and how proposed *Capabilities* and *Roles* will be finally introduced through the system-to-be.

3.4 Dependum

An *Organizational Role depends* on another *Organizational Role* for a *Dependum*, so that the latter may provide the *Dependum* to the former. A *Dependum* can be an *Organizational Goal*, an *Object*, or an *Action*. In the *depend* meta-relationship, the *Organizational Role* that depends on is called the depender, and the *Organizational Role* being depended upon is called the dependee. We define the following dependency types:

- *Organizatonal Goal*-dependency: the depender *depends* on the dependee to *fulfill* and/or *contribute* to an *Organizational Goal*. The dependee is given the possibility

to choose *Plans* through which it will *fulfill* and/or *contribute* to the *Organizational Goal*.

- *Action*-dependency: The depender *depends* on the dependee to accomplish some specific *Action*.
- *Object*-dependency: The depender *depends* on the dependee for the availability of an *Object*.

The *depend* on relationship differs from the dependency relationship in *i** [3] in several aspects. In our meta-model, dependencies are not among *Actors*, but among *Organizational Roles*. *Organizational Roles* are independent concepts and separated from *Actors*. They only *occupy* them, in order to enable changing of *Actors* in *Organizational Roles* without reviewing the entire process or the organizational structure. For example, an *Actor* a_1 can *occupy* some *Organizational Role r*; but if in some point in the future the organization has access to some *Actor* a_2 which can provide better performance in terms of fulfilling and/or contributing to *Organizational Goals* for which *Organizational Role r* is *responsible*, then *Actor* a_2 might be chosen to *occupy* the *Organizational Role r* instead of *Actor* a_1. This replacement would be done without reviewing the entire business process and/or organizational structure – it is sufficient to replace a_1 by a_2 in the *occupy* relationship of *r*.

The *Object*-dependency allows us to represent any specialization of the *Object* meta-concept as a *Dependum*. For example, an *Organizational Role* r_1 might *depend* on another *Organizational Role* r_2 for an *Authorization*. This has implications on the *authority on* meta-relationship, as this dependency means that *r2* must have *authority on* r_1.

4 Goals Sub-model

A *Goal* describes a desired or undesired state of the environment. A state of the environment can be described through the states of *Objects* (*Beliefs*, *Authorizations*, *Resources*, etc.). In addition to standard meta-attributes, a *Goal* is characterized by the optional *Priority* meta-attribute, which specifies the extent the goal is optional or mandatory.

A *Goal* can be *refined* [7] into alternative sets of other *Goals*. Each such set is identified through goal refinement. Informally, goal refinement consists of asking "how" questions about a *Goal* G in order to find alternative sets of *Goals.* Each alternative set of *Goals* that *refine* G provides an alternative way of *fulfilling* and/or *contributing* to G in such a way that the *fulfilment* and/or *contribution* to all of the *Goals* in the set *fulfils* and/or *contributes* to G. Goal refinement is introduced in the model using the *refine* meta-relationship.

The *refine* meta-relationship is an *AndOr* relationship, making it possible to show (directly in the model) alternative refinements of a *Goal*. The *refine* meta-relationship is characterized with an *Alternative Name* and *Alternative Status* meta-attributes. *Alternative Status* indicates whether the alternative is sufficient or not to *fulfil* the *Operational Goal* it *refines*.

Goal types are defined along two axes: *Operational Goals vs. Softgoals* and *Organizational Goals vs. Personal Goals*. In addition, we use patterns to specify the temporal behaviour of *Goals*.

Operational Goals vs. Softgoals. An *Operational Goal* is a set of *Objects* (*Beliefs*, *Resources*, etc.) describing the environment state that can be achieved by *Plans*. We can always determine whether an *Operational Goal* has been *fulfilled* or not by verifying whether the environment state described by the *Operational Goal* has or has not been achieved.

An *Operational Goal* has *State* and *Status* optional meta-attributes. *State* explicitly describes (in terms of *Objects*) the environment in which the *Operational Goal* is fulfilled. *Status* indicates whether the *State* of the *Operational Goal* has been reached, i.e. whether the *Goal* has been fulfilled or not.

A *Softgoal* describes the environment state which can never be achieved since its achievement criteria are not objective. This makes it impossible to formally verify whether a *Softgoal* has been achieved. *Plans* that are otherwise applied to *fulfil Operational Goals* can only *contribute* (positively or negatively) to *Softgoals*. For example, "increase customer satisfaction", "implement a flexible IS", "improve productivity of the workforce", are *Softgoals*.

Organizational Goals vs. Personal Goals. An *Organizational Goal* describes the state of the environment that should be achieved by cooperative behaviour of *Actors*. An *Organizational Goal* is either a *Requirement* or an *Expectation*. A *Requirement* is an *Organizational Goal* under the responsibility of an *Organizational Role occupied* by a *Software Agent*. An *Expectation* is an *Organizational Goal* under the responsibility of an *Organizational Role occupied* by a *Legal Entity*.

Organizational Goals can solve *Conflicts* by specifying the state of the environment in which the *Conflicts* cannot be true.

A *Personal Goal* describes the state of the environment that an *Actor pursues* (wants to obtain) and which can require competitive behaviour among *Actors*.

Organizational Roles are *responsible* of *Organizational Goals*, and *Actors pursue Personal Goals*, i.e., we distinguish what is expected from an actor's participation in the process (through the *Organizational Role* it *occupies*), from what the *Actor* expects from his participation in the process (*fulfilment* of or *contribution* to its *Personal Goals*). In reality, consistency between the *Organizational Goals* and *Personal Goals* is not necessarily ensured. Consequently, it is important to reason about *Conflict* that may arise between *Personal* and *Organizational Goals*, as well as about the degree to which an *Organizational Goal* assists in the *Actor's* pursuit of *Personal Goals*. We use *fulfil* and *contribute* meta-relationships to show how *Plans fulfil* and *contribute* to both *Personal Goals* that the *Actor pursues* and *Organizational Goals* for which its *Organizational Roles* are *responsible*.

5 Related Works

Process-Oriented Approaches such Activity Diagrams, DFDs, IDEF0, workflows (see e.g., [2, 14, 15, 16]) describe enterprise's business processes as sets of activities. Strong emphasis is put on the activities that take place, the order of activity invocation, invocation conditions, activity synchronization, and information flows. Among these approaches, workflows have received considerable attention in the literature. In such kind of process-oriented approaches, agents have been treated as a

computational paradigm, with focus on the design and implementation of agent systems, not analysis on enterprise models.

Actor-Oriented Approaches emphasize the analysis and specification of the role of the actors that participate in the process [17]. The *i** modelling framework [3] has been proposed for business process modelling and reengineering. Processes, in which information systems are used, are viewed as social systems populated by intentional actors which cooperate to achieve goals. The framework provides two types of dependency models: a strategic dependency model used for describing processes as networks of strategic dependencies among actors, and the strategic rationale model used to describe each actor's reasoning in the process, as well as to explore alternative process structures. The diagrammatic notation of *i** is semi-formal, and proved useful in requirements elicitation (see e.g., [18, 12, 19]). In this context, actor-oriented approaches provide significant advantages over other approaches: agents are autonomous, intentional, social, etc. [21] which is of particular importance for the development of open distributed information systems in which change is ongoing. However, actors have served mostly as requirements engineering modelling constructs for real-world agents, without assuming the use of agent software as the implementation technology nor the use of organizational actors for enterprise modelling.

Goal-Oriented Approaches focus on goals that the information system or a business process should achieve. Framework like KAOS [5, 8] provides a formal specification language for requirements engineering, an elaboration method, and meta-level knowledge used for guidance while the method is applied [6]. The KAOS specification language provides constructs for capturing the various types of concepts that appear during requirements elaboration. The elaboration method describes steps (i.e. goal elaboration, object capture, operation capture, etc. [6]) that may be followed to systematically elaborate KAOS specifications. Finally, the meta-level knowledge provides domain-independent concepts that can be used for guidance and validation in the elaboration process.

Enterprise Knowledge Development (EKD) [17] is used primarily in modelling of business processes of an enterprise. Through goal-orientation, it advocates a closer alignment between intentional and operational aspects of the organization and links re-engineering efforts to strategic business objectives. EKD describes a business enterprise as a network of related business processes which collaboratively realise business goals. This is achieved through several sub-models: enterprise goal sub-model (expressing the causal structure of the enterprise), enterprise process sub-model (representing the organizational and behavioural aspects of the enterprise), and information system component sub-model (showing information system components that support the enterprise processes) [17]. Agents appear in the EKD methodology but without explicit treatment of their autonomy and sociality [21]. In KAOS, agents interact with each other non-intentionally, which reduces the benefits of using agents as modelling constructs.

6 Conclusion

Modelling the organizational and operational context within which a software system will eventually operate has been recognized as an important element of the engineering process (e.g., [20]). Such models are usually founded on primitive

concepts such as those of *actor* and *goal*. Unfortunately, no specific enterprise modelling framework really exists for engineering modern corporate IS. This paper proposes an integrated agent-oriented meta-model for enterprise modelling. Moreover, our approach differs primarily in the fact that it is founded on ideas from in requirements engineering frameworks, management theory concepts found to be relevant for enterprise modelling and agent oriented software engineering.

We have only discussed here the concepts that we consider the most relevant at this stage of our research. Further classification of, for instance, goals is possible and can be introduced optionally into the meta-model. For example, goals could be classified into further goal categories such as Accuracy, Security, Performance, etc. We also intend to define a strategy to guide enterprise modelling using our meta-model as well as to define a modelling tool à la Rational Rose to visually represent the concepts.

References

1. Koubarakis M., Plexousakis D.: A formal framework for business process modelling and design. Information Systems 27, 2002, pp. 299-319.
2. Kamath M., Dalal N.P., Chaugule A., Sivaraman E., Kolarik W.J.: A Review of Enterprise Process Modelling Techniques. In Prabhu V., Kumara S., Kamath M.: Scalable Enterprise Systems: An Introduction to Recent Advances. Kluwer Academic Publishers, Boston, 2003.
3. Yu E.: Modelling Strategic Relationships for Process Reengineering. Ph.D. thesis, Dept. of Computer Science, University of Toronto, 1994.
4. Bernus P.: Enterprise models for enterprise architecture and ISO9000:2000. Annual Reviews in Control 27, 2003, pp. 211–220.
5. Dardenne A., van Lamsweerde A., Ficklas S.: Goal-directed requirements acquisition. Sci. Comput. Programming 20, 1993, pp. 3-50.
6. van Lamsweerde A., Darimont R., Letier E.: Managing Conflicts in Goal-Oriented Requirements Engineering. IEEE Transactions on Software Engineering, Special Issue on Managing Inconsistency in Software Development, 1998.
7. van Lamsweerde A.: Goal-Oriented Requirements Engineering: A Guided Tour. Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering, Toronto, 2001, pp. 249-263.
8. van Lamsweerde A.: The KAOS Meta-Model –Ten Years After. Technical report, (2003).
9. Simon H. A.: Administrative Behavior : A Study of Decision-Making Processes in Administrative Organization. New York: The Free Press 3rd ed., 1976.
10. Johnson G., Scholes K.: Exploring Corporate Strategy, Text and Cases. Prentice Hall, 2002.
11. Brickley J.A., Smith C.W., Zimmerman J.L.: Managerial Economics and Organization Architecture. McGraw-Hill Irwin 2nd ed., 2001.
12. Faulkner S., Kolp M., Coyette A., Tung Do T.: Agent-Oriented Design of E-Commerce System Architecture. Proceedings of the 6th International Conference in Enterprise Information Systems Engineering, Porto, 2004.
13. Simon H. A.: Rational Decision Making in Business Organizations. The American Economic Review vol.69 no.4, 1979, pp. 493-513.
14. Elmagarmid A., Du W.: Workflow Management: State of the Art Versus State of the Products. In Dogac A., Kalinichenko L., Tamer Ozsu M., Sheth A.: Workflow Management Systems and Interoperability. NATO ASI Series, Series F: Computer and Systems Sciences 164, Springer Heidelberg, 1998.

15. Mentzas G., Halaris C., Kavadias S.: Modelling business processes with workflow systems: an evaluation of alternative approaches. International Journal of Information Management 21, 2001, pp. 123-135.
16. Sheth A.P., van der Aalst W., Arpinar I.B.: Processes Driving the Networked Economy. IEEE Concurrency 7, 1999, pp. 18-31.
17. Kavakli V., Loucopoulos P.: Goal-Driven Business Process Analysis Application in Electricity Deregulation. Information Systems 24, 1999, pp. 187-207.
18. Liu L., Yu E.: Designing information systems in social context: a goal and scenario modelling approach. Information Systems 29, 2004, pp. 187–203.
19. Briand L., Melo W., Seaman C., Basili V.: Characterizing and Assessing a Large-Scale Software Maintenance Organization. In Procedings of the 17th International Conference on Software Engineering, Seattle, WA, 1995.
20. Castro J., Kolp M. and Mylopoulos J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. In Information Systems (27), Elsevier, Amsterdam, 2002.
21. Yu E.: Agent-Oriented Modelling: Software Versus the World. Proceedings of the Agent-Oriented Software Engineering AOSE-2001 Workshop, Springer Verlag, 2001.
22. Chung L.K., Nixon B.A., Yu E., Mylopoulos J.: Non-Functional Requirements in Software Engineering. Kluwer Publishing, 2000.