In Proceedings of The Eighteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'06).
*5-7 July, 2006. San Francisco, California, USA.*

# A Comparative Analysis of *i**Agent-Oriented Modelling Techniques

Gemma Grau, Carlos Cares, Xavier Franch, Fredy J. Navarrete
*Universitat Politècnica de Catalunya (UPC)*
*C/Jordi Girona 1-3, UPC-Campus Nord (C6), Barcelona (Spain)*
*{ggrau, ccares, fjnavarrete, franch}@lsi.upc.edu*

## Abstract

*Agent-oriented software engineering has become an extended practice. The autonomy and flexibility provided by agents makes it appropriate either for the development of agent-based systems and for the development of complex and distributed software systems. Due to this wide applicability there are many modelling languages and methodologies for representing and developing systems using the agent-oriented paradigm. Among them, the i* language is very appropriate, not only for the agent concepts that it models, but also for its capabilities in the disciplines of requirements engineering and organizational process modelling. Nowadays, i* is one of the most widespread notations used for these purposes, and agent-oriented methodologies may take advantage of its existence in the requirements phase. However, due to the degree of freedom inherent to the i* language, the construction of the models may not be an easy task. To give light to this subject, we present several i* agent-oriented modelling techniques and we compare them according to a certain set of criteria.*

## 1. Introduction

The difficulty involved in the design and development of a software system increases with the complexity of such a system. To solve this problem, several software engineering paradigms have been proposed, being one of them agent-oriented software engineering [1]. The agent-oriented paradigm provides the means for representing the organizational structure of any kind of system and the interactions between the entities (computational or human) involved. Agents exhibit properties such as autonomy, reactivity, pro-activeness and social ability, which allow representing, analysing and designing software solutions for agents and multi-agents systems, but also for other kinds of complex systems that involves distribution of data and control, legacy systems or open systems [2].

The success in using the agent-oriented paradigm is enforced by the great amount of existing agent-oriented methodologies. As most of them address different issues, a crucial step is to understand the relationships between these various methodologies and to understand the main properties addressed by them [3]. With this aim, there is a great amount of literature for comparing and evaluating agent-oriented methodologies [3, 4, 5, 6, 7, 8, 9, 10, 11].

Agent-oriented methodologies are mainly composed of four different phases, namely the analysis, design, implementation and verification phases [2], which are done at the level of abstraction more adequate to the problem to be solved. It is during the analysis phase that the acting entities of the problem domain are identified and modelled as agents. On the other hand, the agents and their actions (or behaviour) are refined and specified in the design phase. In agent-oriented methodologies these two phases have to include a requirements gathering stage that is not always supported by the proposed processes [12]. For instance, GAIA [1] views the requirements capture phase as being independent of the paradigm used for analysis and design and, so, it only focus on the late requirements phase. A similar problem is detected in Prometheus [13], where a simple version of KAOS is used to describe the goals of the system, but description of business models is not supported. On the other hand, TROPOS [12] uses the *i** framework [14] as its modelling technique, and because of this, it is one of the methodologies that better supports early and late requirements analysis [12, 7]. Further work on Prometheus [15], proposes to use the facilities provided by *i** for refining the methodology and improve its early requirements analysis phase. As *i** is an agent-oriented modelling language, most of the constructs that it defines can be adapted to other agent-oriented modelling languages. This is the reason why it is possible to use it for obtaining and modelling early requirements in agent-oriented methodologies and, then, defining transformation guidelines between *i** and other agent-oriented modelling languages such as proposed in [15].

Despite the usefulness of *i** models, their construction may be difficult because of the degree of freedom provided by the language. In order to address this issue, we have studied those *i** methods that directly address the construction of *i** models and we have compared them following a certain set of criteria. More precisely, the only methods that provide precise guidelines for constructing *i** models are, as far as we know:

- The TROPOS methodology [12];
- The *goal-based business modelling oriented towards late requirements generation* method [16];
- A methodology for mapping activity theory diagrams into organizational models based on *i** [17];

- A methodology for building *i** models from BPEL process descriptions [18];
- The R*i*SD methodology [19]; and
- The PR*i*M methodology [20].

The objective of this paper is to present and compare the existing set of *i** modelling methods. As similar work has already been done for agent-oriented methodologies, related work is presented in section 2. Section 3 presents the *i** framework in more detail. In section 4, an overview of the different *i** model construction methods is provided. Section 5 contains the comparison criteria, which are used in section 6 for analysing and comparing the different methods. Finally, section 7 presents the conclusions.

## 2. Related work

In order to successfully apply the agent-oriented paradigm we have first to decide if an agent-oriented approach is suitable and, second, to choose the most appropriate agent-oriented methodology. The adequacy of the agent-oriented paradigm against other paradigms is discussed in [1] and [21]. In [1], the agent-oriented paradigm is balanced against the object-oriented paradigm, whilst [21] provides guidelines for evaluating the benefits of an agent-oriented approach against the adoption of other methodologies such as the object-oriented ones. Once the decision of adopting an agent-oriented paradigm is assumed, the already existing agent-oriented methodologies have to be analysed and, as most of them address different issues, the most appropriate has to be chosen by analysing, evaluating and comparing them.

Agent-oriented methodologies are generally built by extending other existing methodologies such as object-oriented or knowledge oriented methodologies in order to include the relevant aspects for agents [4]. As all the methodologies can be categorized according to its origin, it is possible to establish a genealogy of agent-oriented methodologies, such the one presented in [11]. Therein, the TROPOS methodology [12] comes from *i**; and *i** inherits both from the field of Requirements Engineering and the techniques related to the Object-Oriented patterns for agents.

There exist several proposals whose only aim is to compare and evaluate agent-oriented methodologies. For instance, in [4] several methodologies are described, stating its origin, the models they propose and the processes or phases undertaken to construct the models. Although this work does not explicitly compares the methodologies, there are other proposals that do it according to a set of established criteria [3, 5, 6, 7, 9, 10, 11].

The most common categorization of the criteria includes concepts, notation, process and pragmatics [9, 10, 11]. The **concepts** deal with those aspects related on how a methodology adheres to the basic notions of agents-based systems and, so, includes criteria such as *proactiveness*, *reactiveness*, and *adaptability*. Those criteria can be grouped into those that are internal to the agent such as *autonomy* or *mental notions*, and the ones that refer to its interaction with other agents such as *social ability*, *protocols* or *capability to deal with multiple interests*. The **notation** category is related to the modelling techniques that the methodologies exhibit. It includes criteria for evaluating the *syntax* and *semantics* of the formalism used; the *adequacy* and *expressiveness* of the language; and the *refinement*, *modularity* it provides. The **process** category groups all those aspects dealing with the process development aspect of a methodology, including criteria such as the *lifecycle coverage*, the *development context* and the *activities* proposed in the methodology. Finally, the **pragmatics** refers to the management criteria and technical issues that are provided to the users when adopting it. It contains criteria such as *target domain*, *quality*, *cost estimation*, *resources* or *tool support*.

Although not all the comparison frameworks propose the same criteria, overlapping exist in most of the categories because all them share an interest for evaluating both the agent concepts and their representation, and the methodological process itself. For instance, [6] establishes two categories of concepts: *internal attributes* and *interaction attributes*; and [5] groups the criteria of notation, process and pragmatics into the category *software engineering attributes*. On the other hand, the evaluation frameworks in [7, 11] consider the platform in their evaluation. In particular, [7] considers a *technology dimension* in order to include aspects such as *mode of processing*, *human-machine interface* and *development environment*; whilst [11] distinguishes between platform independent and platform dependent attributes and evaluates the methodologies according to a concrete platform.

The evaluations are completed by applying the criteria individually in each methodology [6, 10] or by applying a certain attribute over a set of methodologies [5, 7, 9, 3]. Most of the evaluations provide a qualitative evaluation of the proposed criteria, but there are also some different approaches. For example: [3] compares the agent-oriented methodologies by using the NFR approach; [6] and [10] adopt a quantitative approach by evaluating the methodologies with numerical values and, subsequently, obtaining quantitative results; and, finally, the proposals [3, 7, 8] define an exemplar as well as attributes for analysing, comparing and evaluating the methodologies. The agent-oriented methodologies that are evaluated more often in the proposals are: MaSE, evaluated in [7, 9, 11]; Agent Modelling Technique for Systems of BDI Agents, evaluated in [4, 6, 7]; GAIA, evaluated in [4, 7, 10]; and TROPOS, evaluated in [3, 9, 11].

As we have already mentioned, agent-oriented and goal-oriented requirements engineering are approaching because goals help to abstract the intentionality of agents, such as it is revealed by *i**. Regarding comparison frameworks on goal-oriented software engineering, most of the proposed methods introduce and analyze other proposals in their related work. However, as far as we know, only [22] specifically compare goal-oriented methods. The criteria used include the usage of the methods, the subject addressed, the kind of representation and the development support. Among the compared methods, there are *i**, GRAM and KAOS.

## 3. The *i** framework

The *i** framework proposes the use of two types of models for representing systems, each one corresponding to a different abstraction level: a Strategic Dependency (SD) model represents the intentional level and the Strategic Rationale (SR) model represents the rational level.

A SD model consists of a set of nodes that represent actors and a set of dependencies that represent the relationships among them. Dependencies expresses that an actor (*depender*) depends on some other (*dependee*) in order to obtain some objective (*dependum*). Thus, the *depender* depends on the *dependee* to bring about a certain state in the world (goal dependency), to attain a goal in a particular way (task dependency), for the availability of a physical or informational entity (resource dependency) or to meet some non-functional requirement (softgoal dependency).

A SR model allows visualizing the intentional elements into the boundary of an actor in order to refine the SD model with reasoning capabilities. The dependencies of the SD model are linked to intentional elements inside the actor boundary. The elements inside the SR model are decomposed accordingly to two types of links:

- *Means-end* links establish that one or more intentional elements are the means that contribute to the achievement of an end. The "end" can be a goal, task, resource, or softgoal, whereas the "means" is usually a task. There is a relation OR when there are many means, which indicate the different ways to obtain the end. *Contribution links* are *Means-end links* with a softgoal as an *end*, which allows stating explicitly if the contribution of the *means* towards the *end* is negative or positive.
- *Task-decomposition* links state the decomposition of a task into different intentional elements. There is a relation AND when a task is decomposed into more than one intentional element.

For more details about *i**, we refer to Yu's seminal work [14] and also to [23] for an *i** reference model in UML.

## 4. Overview of *i** modelling techniques

The **TROPOS methodology** [12] addresses agent-oriented development and it is intended to support all analysis and design activities in the software development process. TROPOS models information systems as social structures by means of a collection of social actors and the social dependencies among them. The modelling of these concepts are associated to different activities. Therefore, the TROPOS methodology consists of five phases: early requirements analysis, late requirements analysis, architectural design, detailed design, and implementation. We remark that requirements analysis is split into two phases: early requirements and late requirements analysis, both sharing the same conceptual and methodological approach. However, in the early requirements phase the domain stakeholders are identified and modelled as social actors, stating the *why* behind the system functionalities. As a last result in the overall process, the stated early requirements support the verification of how the final implementation matches the initial needs. In the late requirements phase the conceptual model is extended including the system as a new actor and its dependencies with the other actors of the environment. These dependencies define all the functional and non-functional requirements of the system-to-be.

The objective of the ***goal-based business modelling oriented towards late requirements generation*** method [16] is to use a business model for constructing a software requirements specification. The method proposes to use a goal-based elicitation method in order to capture the organizational context in a so-called Goal-Refinement Tree, which contains the successive decomposition of goals into subgoals. Such decomposition is done by means of classification and elicitation strategies. Once the Goal-Refinement Tree is constructed, several steps are applied in order to map its goals into the *i** SD elements. The *i** SR model is created afterwards by representing the actors internal goals. The final model is used to perform business improvement analysis by means of inserting the system actor into the system. This analysis leads to strategic models that can be used to derive functional (use case) specifications with their corresponding scenarios, which can be done by using the guidelines provided in [24].

The methodology presented in [17] proposes the **mapping of activity theory diagrams into organizational models based on *i****. Thus, it seeks to build *i** models based on activities theory in order to document functional and non-functional requirements and, so, provide a better understanding of the process. Taking the activity theory models as a starting point, the activities and their actions are analysed in order to construct the model. This analysis is done by decomposing the activities into actions and the actions into sub-actions until the actions are kept simple enough. The method provides concrete guidelines for mapping the activity theory concepts to an SD *i** model, providing rules for distinguish the *depender* and the *dependee*, as well as the different kinds of dependencies. SD models are then used to build the SR by following analogous guidelines.

The **methodology for building *i** models from BPEL process descriptions** [18] provides several guidelines to guide the mapping between BPEL Web Services descriptions into *i** diagrams. The SD and SR models are developed simultaneously by applying the guidelines, which map the concepts provided in the BPEL descriptions to the *i** constructs. Once the models are generated, they can be automatically translated into the action language ConGolog in order to run simulations and analyse the model properties. This method relies on a very formal basis and its main goal is to represent and evaluated agent-based designs for inter-organizational networks.

The **R*i*SD methodology** [19] is aimed at building Reduced *i** SD models for software systems. R*i*SD is defined as a set of activities structured in two phases, one for constructing the social system and the other for constructing the socio-technical system. The social system model does not include the software system and focuses on the stakeholder needs. Its construction is iterative and begins with the identification of the initial set of social actors involved, their main goals and their strategic dependencies. The process iterates until all the actors and dependencies are obtained. The socio-technical system model incorporates the software system and the SD model dependencies are reassigned around this new actor. The system may be further decomposed into subsystems which are modelled as new actors and, therefore, the existing dependencies are reassigned again. New subsystems may depend on each other and these dependencies are also established.

The **PR*i*M methodology** [20] addresses *i** modelling from the business process reengineering perspective, where the specification of the system-to-be starts from the observation of the current system and uses the *i** framework for modelling this process. The *i** model is used for generating several process alternatives that can be evaluated using structural metrics [25]. The final model can be used for the specification of the new system, taking advantage of the possibility of connecting strategic reasoning with information system development. Based upon this business process reengineering point of view the PR*i*M methodology is composed of five phases. In the first one, the current process is analysed and the information obtained is summarized into Detailed Interaction Scripts (DIS). Every activity produces a DIS, which contains the activity preconditions, postconditions and triggering events, and a description of the actions and the resources involved in the activity. In the second phase, an *i** model is constructed by using a set of prescriptive guidelines in order to obtain the process view of the current system and its rationale. In the third phase the systematic generation of process alternatives is done by means of the addition of new actors and the reallocation of responsibilities between them. Those different alternatives are evaluated in the fourth phase and one of them is selected as the solution. Finally, the specification of the new system is generated based on the chosen process alternative.

There are some other approaches that address the construction of *i** models. However, as their main goal is not the construction of the models but their use they are not included in this analysis, and we just briefly summarise the three most widespread. First, the seminal proposal of *i** [14], which clearly introduces the *i** constructors and their uses, but provides small guidance on how models are constructed. Second, the organizational patterns provided in [26] which are based on organizational structures than can be used as a basis to generate *i** models. This approach is suitable when a specific organizational architecture is modelled, but there is little guidance on how to use the patterns and no description on how exactly this patterns are defined. Finally, the RESCUE process [27] provides several guidelines for applying *i** modelling at the requirements specification stage with the aim of discovering and eliciting requirements. However, its main goal is to get the system specification rather than the *i** model.

## 5. Comparison criteria

The analysis and comparison of agent-oriented methodologies has been widely studied [3, 4, 5, 6, 7, 8, 9, 10, 11]. As we have already mentioned in section 2, some of these proposals state a set of comparison criteria for comparing agent-oriented methodologies according to the concepts, the notation, the process and the pragmatics they exhibit. However, in the context of the *i** modelling techniques, all the methodologies use the same concepts and graphical constructors, which are the ones provided by *i**. Consequently, the concepts and notation are not relevant for comparing them and, so, the focus is on the methodological process. The criteria used for evaluating the i* modelling methods are based upon the ones proposed on the agent-oriented methodology comparison frameworks and have been grouped into the following categories: the **development process**, the **prescriptiveness** of its steps, the **resources** involved and the ***i** issues** addressed.

According to [10], a **development process** is a series of actions, changes, and functions that, when performed, result in a working computerized system. Thus, for evaluating the process we take into account the **development context** where the method is applied (e.g., creating new software, reengineering of existing software, prototyping, designing for reusing components). For establishing the **lifecycle coverage** of the method we consider the phases proposed in TROPOS [12]: early requirements, late requirements, architectural design, detailed design and implementation. As mentioned in [1], there are three different approaches for identifying the behaviour of the agents: top-down approaches, which are based on progressive decomposition of behaviour; bottom-up approaches, which begin by identifying elementary agent behaviours; and mixed approaches. Thus, the **development approach** adopted when building the *i** models is also analysed. Some of the methods present **process restrictions** that constraint the situations where they can be applied.

**Prescriptiveness** is concerned with guidelines and rules for a correct and unambiguous use of the *i** constructors. To deal with these issues, we compare the degree of detail provided when describing each of the stages proposed by the methods. This includes the **method decomposition** in smaller parts (e.g. steps) in order to tackle the problem complexity, the existence of precise **guidelines or heuristics** for constructing the models and the existence of **examples**.

**Resources** are crucial when using a method, in the analysis two different kinds of resources are studied: resources generated by the method and resources available for applying the method. The **resources generated** by the methodology are mainly SD and SR models which can be developed at different levels of detail. If the application of the method generates **intermediate artefacts**, they are also considered. On the other hand, the **resources provided** are those that are available for applying the methodology such as templates, patterns or software tools.

The nature and objectives of the seminal proposal of *i** [14] has provided the language with a certain degree of freedom (see

[23] for a comparative analysis on *i**-based agent-oriented languages). This flexibility when applying the language results into new constructors that appear on behalf of the methods needs. In order to evaluate those ***i* issues*** we consider how each method addresses the *i** framework by means of the addition of new *i** constructors. As *i** models become difficult to manage when the complexity of the system grows, we also analyse how the **scalability** of the models is tackled.

## 6. Applying the comparison criteria

Based on the criteria proposed in the previous section, we have compared the methodologies and we have obtained the analysis results that are summarized in Table 1. In order to facilitate to reference them, some of the methods names are abbreviated as follows: GBM stands for the *Goal-based Business Modelling oriented towards late requirements generation method* [16]; ATM stands for the methodology for mapping Activity Theory diagrams into organizational Models based on *i** [17]; and, BPD stands for the methodology for building *i** models from BPEL Process Descriptions [18].

**Context:** Addressing the development context, the methods GBM, ATM and R*i*SD are aimed at supporting the specification of a new software system. BPD addresses the representation and dynamic evaluation of agent-based designs and, thus, aims at prototyping. PR*i*M adopts a process reengineering perspective which includes the option of designing the new system by reusing software components. TROPOS is the only method that aims at developing new software and, thus, its lifecycle coverage goes from early requirements to implementation. Both early requirements and late requirements are addressed by GBM, R*i*SD and PR*i*M; whilst ATM main focus is on early requirements and BPD main focus on the detailed design. The development approaches adopted by TROPOS, GBM and R*i*SD are top-down; ATM and BPD are bottom-up; whilst PR*i*M has a mixed approach. The restrictions applied over the methods deal with the starting situation from where they have to be applied: TROPOS, GBM and R*i*SD do not need auxiliary information and, so, they can be built from the scratch. On the other hand, activity diagrams, BPEL descriptions and the complete documentation of the existing process are needed as a starting point for ATM, BPD and PR*i*M, respectively.

**Prescriptiveness:** All the methods decompose the problem into steps or stages for a better application of their guidelines.

| Comparison Criteria | | **TROPOS** [12] | **GBM** [16] | **ATM** [17] | **BPD** [18] | **R*i*SD** [19] | **PR*i*M** [20] |
|---|---|---|---|---|---|---|---|
| **Process** | **Development context** | - Creating new software | - Specifying new software | - Specifying new software | - Prototyping | - Specifying new software | - Reengineering Design for components reuse |
| | **Lifecycle coverage** | - From early requirements to implementation | - Early Requirements<br>- Late Requirements | - Early Requirements | - Detailed Design | - Early Requirements<br>- Late Requirements | -Early Requirements<br>- Late Requirements<br>- Arch. Design |
| | **Development approach** | - Top-down | - Top-down | - Bottom-up | - Bottom-up | - Top-down | - Mixed |
| | **Restrictions** | - | - | - Activity Diagrams as starting point | - BPEL descriptions as starting point | - | - Current process as starting point |
| **Prescriptiveness** | **Problem decomposition** | - Different steps for early requirements and late requirements | - Different steps for the construction of the SD and the SR | - Different steps for the construction of the SD and the SR | - SD and SR models are build in parallel in different steps | - Different steps for early requirements and late requirements | - SD and SR models are build in parallel in different steps |
| | **Guidelines and heuristics** | - Heuristics for the *i** element identification<br>- Heuristics to decide the *dependum* kind | - Mapping guidelines from the Goal-Refinement Tree to *i** | - Mapping guidelines from activity diagram<br>- Heuristics to decide *dependum* kind | - Mapping Guidelines from BPEL | - Heuristics for the *i** element identification<br>- Heuristics to decide the *dependum* kind | - Mapping Guidelines from the DIS<br>- Consistency checks |
| | **Example** | - eCulture System | - Conference Review Process | - Project-based Learning activities | - Loan Service | - Information Reliability | - Meeting Scheduler |
| **Involved Resources** | **Resources produced** | - Complete SD and SR models | - Complete SD and SR models<br>- Considers alternative paths | - Complete SD and SR models<br>- Do not considers alternative paths | - Complete SD and SR models<br>- Do not considers alternative paths | - Complete SD model<br>- Partial SR model<br>- Considers alternative paths | - Complete SD model<br>- Partial SR model |
| | **Intermediate Artifacts** | - Capability diagrams<br>- Plan diagrams<br>- Agent interaction diagrams | - Goal-Refinement Tree | - Decomposition of the activities in actions<br>- Decomposition of actions into subactions | - | - | - DIS: Detailed Interaction Scripts for each activity |
| | **Method Resources** | - JACK tool [12] | - | - | - SNet Tool [28] | - | - REDEPEND-REACT [29] |
| ***i* Issues*** | **Constructors Used** | - Basic *i** [14] | - Basic *i** [14] Without softgoals and means-end link | - Basic *i** [14] | - Basic *i** [14]<br>- Without means-end<br>- Adds the constructor *task precondition* | - Basic *i** [14]<br>- Adds the constructor *support* | - Basic *i** [14]<br>- Restricts the SR decomposition |
| | **Scalability** | - By goal-decomposition | - By goal-decomposition | - By decomposing the problem in activities | - Not addressed | - Using the supports decomposition | - By decomposing the problem in activities |

**Table 1.** Comparison of *i** methodologies according to the established set of criteria

This decomposition divides the process into late requirements and early requirements specification in TROPOS and R*i*SD, into the construction of SD or SR models in GBM and ATM and in several steps for a parallel SD and SR construction in BPD and PR*i*M. All the methods propose concrete guidelines for creating the models. For instance, in ATM and BPD, guidelines determine the mapping between the elements of the starting artefacts and the final *i** model elements. GBM and PR*i*M generate intermediate artefacts and provides mapping guidelines between their elements and the final *i** model elements. TROPOS and R*i*SD provide identification heuristics in order to facilitate the identification the *i** elements without any artefact as a starting point. On the other hand, in GBM, BPD and PR*i*M, the guidelines strongly determine the kind of the dependum and other strategic elements used. More freedom is provided in TROPOS, ATM, and R*i*SD, where additional heuristics are provided to help to decide the dependency kind (goal, resource, task or softgoal). Additionally, PR*i*M provides consistency checks in order to ensure the correct application of the guidelines. Every method is demonstrated by means of an illustrative example in order to clarify the explanations: TROPOS models an eCulture System; GBM a Conference Review Process; ATM models Project-based Learning Activities; BPD a Loan Service; R*i*SD models information reliability; and PR*i*M a Meeting Scheduler.

**Involved Resources:** TROPOS is the only method that builds complete SD and SR models. GBM, ATM, BPD, and PR*i*M develop a complete SD model but the SR does not allow alternative analysis because only task-decomposition is used. Alternatives are developed in PR*i*M as alternative models. R*i*SD SD models are also complete and the SR model contains both *task-decomposition* and *means-end* links for allowing the representation of alternatives. However, R*i*SD's SR model is not considered complete because it is built as an auxiliary model in order to support the development of the SD model. According to the order in which SD and SR models are developed, GBM and ATM fully develop the SD model before developing the SR model, whilst in the other methods the construction of the two can be intertwined. The intermediate artefacts produced by each of the methodologies are the following : TROPOS permits to generate capability diagrams, plan diagrams and agent interaction diagrams; GBM generates an intermediate Goal-Refinement Tree; ATM provides the decomposition of activities into actions, and the decomposition of these actions into subactions; and PRiM produces Detailed Interaction Scripts (DIS). We remark that some of the methods provide specific tools for supporting its processes. In particular, TROPOS proposes the JACK tool [12], BPD proposes the SNet Tool [28] and PR*i*M is partially supported by REDEPEND-REACT [29].

***i** issues:** All the methods are based on the seminal proposal of *i** [14], however some of them do not use all the constructors, whilst others have added their own constructors. As we have already mentioned, GBM, ATM and BPD do not use means-end links in the SR models and, so, do not consider alternative analysis. We also remark that GBM does not use the softgoal construct in any of the produced models. On the other hand, BPD extends *i** with the constructor *Task precondition* in order to describe task preconditions and effects. R*i*SD also adds a new constructor, *supports*, which establish a relation between related dependencies. This constructor tackles scalability in R*i*SD, because it helps to address the problem incrementally by decomposing the already existing *i** elements. A similar approach is adopted in TROPOS and GBM where scalability can also be handled by goal-decomposition. In BPD this issue is not mentioned. Finally, ATM and PR*i*M deal with scalability by decomposing the problem in activities and the activities into actions.

## 7. Conclusions

The *i** framework is agent-oriented and thus, is suitable in the context of agent-oriented software engineering. As it is also successfully applied in requirements engineering and organizational modelling, *i** can also support the requirements engineering phase in those agent-oriented methodologies where requirements gathering or organizational rational is not explicitly addressed. In order to construct *i** models a specific methodology has to be applied. We have presented six different methodologies and we have analysed them in order to inform their use. The comparison focuses on the process, the prescriptiveness of the provided explanation, the resources involved and the *i** issues addressed by each method. The criteria used in each of these categories had been defined upon the ones already defined in other agent-oriented methodologies comparison frameworks. As a result, and this is the main contribution of our work, these criteria show under which circumstances one method may be more valuable than the others and therefore their selection may now rely on objective criteria rather than on a subjective belief.

Further work includes a deeper analysis of the *i** modelling methods by using an exemplar such as proposed in [7, 9, 11] and by evaluating the comparison framework from different points of view (e.g., the authors, students, industry practitioners). Also a quantitative approach would be adopted by refining the evaluation criteria into new ones in order to allow a numerical evaluation and, then, applying a quantitative approach [6, 9]. Case studies on using and selecting the methodologies would also be completed.

## Acknowledgements

This work has been done in the framework of the research project UPIC, ref. TIN2004-07461-C02-01, supported by the Spanish Ministerio de Ciencia y Tecnología. Some authors have grants that partially support their work: C. Cares, by the MECE-SUP FRO0105 Project of the Chilean government and Universidad de la Frontera; and G. Grau, by a UPC research scholarship.

## 8. References

[1] M. Wooldridge, N. R. Jennings, and D. Kinny. "The Gaia methodology for agent-oriented analysis and design". Journal on Autonomous Agents Multi-Agents Systems, Vol. 3, No. 3, 2000.

[2] N.R. Jennings. "On Agent-Based Software Engineering". Artificial Intelligence, Vol. 117, No. 2, 2000. pp: 277–296.

[3] C.T.L. Silva, P.C.A.R. Tedesco, J.F.B. Castro, R.C. Silva. "Comparing Agent-Oriented Methodologies Using the NFR Approach". In *Proceedings of the Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*, SELMAS 2004. Vol. 1, pp: 1-9.

[4] C.A.. Iglesias, M. Garijo, J.C. González. "A Survey of Agent-Oriented Methodologies". In *Proceedings of the Fifth International Workshop on Intelligent Agents V. Agent Theories, Architectures, and Languages*, ATAL 1998. pp: 317 – 330.

[5] O. Shehory and A. Sturm. "Evaluation of modeling techniques for agent-based systems". In *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.

[6] L. Cernuzzi, G. Rossi. "On the evaluation of agent oriented modeling methods". In *Proceedings of the First International Workshop on Agent-Oriented Methodologies*, held at OOPSLA 2002. pp. 21-30.

[7] A. Sabas, S. Delisle, M. Badri. "A Comparative Analysis of Multiagent System Development Methodologies: Towards a Unified Approach", In *Proceedings of the Third International Symposium "From Agent Theory to Agent Implementation"*, held at the *16th European Meeting on Cybernetics and Systems Research*, 2002.

[8] E.S.K. Yu, L.M. Cysneiros. "Agent-Oriented Methodologies - Towards a Challenge Exemplar". In *Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems*, AOIS 2002.

[9] K.H. Dam, M. Wnikoff. "Comparing Agent-Oriented Methodologies". In *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems*, AAMAS 2003.

[10] A. Sturm, O. Shehory. "A Framework for Evaluating Agent-Oriented Methodologies". In *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems,* , AOIS 2003. pp: 94-109.

[11] J. Sudeikat, L. Braubach, A. Pokahr, W. Lamersdorf. "Evaluation of Agent - Oriented Software Methodologies - Examination of the Gap Between Modeling and Platform". In *Proceedings of the Fifth International Agent-Oriented Software Engineering Workshop*, AOSE 2004. pp: 126-141.

[12] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos. "TROPOS: An Agent-Oriented Software Development Methodology". *Journal of Autonomous Agents and Multi-Agent Systems*. May 2004. Volume 8, Issue 3.

[13] L. Padgham, M. Winikoff. *Developping Intelligent Agent Systems - A practical Guide*, John Wiley & Sons, 2004.

[14] E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD. thesis, University of Toronto, 1995.

[15] G. Cysneiros, A. Zisman. "Refining the Prometheus Methodology with *i**". In *Proceedings of the Third International Workshop on Agent-Oriented Methodologies*, held at OOPSLA 2004.

[16] H. Estrada, A. Martínez, O. Pastor. "Goal-based business modeling oriented towards late requirements generation". In *Proceedings of the 22nd International Conference on Conceptual Modeling*, 2003. Springer-Verlag, LNCS 2813. pp: 277-290.

[17] G.C. Neto, A.S. Gomes, J.B. Castro. "Mapeando Diagramas da Teoria da Atividade em Modelos Organizacionais Baseados em *i**". In *Proceedings of the 7th Workshop em Engenharia de Requisitos*, WER 2004. pp: 39-50.

[18] D. Schmitz, G. Lakemeyer, G. Gans, M. Jarke. "Using PBEL Process Descriptions for Building up Strategic Models for Inter-Organizational Networks". In *Proceedings of the Workshop on Modeling Inter-Organizational Systems*, MIOS 2004. Springer-Verlag, LNCS 3294. pp: 520-532.

[19] G. Grau, X. Franch, E. Mayol, C.P. Ayala, C. Cares, M. Haya, F. Navarrete, P. Botella, C. Quer. "R*i*SD: A Methodology for Building *i** Strategic Dependency Models". In *Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering*, SEKE 2005. pp: 259-266.

[20] G. Grau, X. Franch, N.A.M. Maiden. "A Goal Based Round-Trip Method for System Development". In *Proceedings of the 11th International Workshop on Requirements Engineering: Foundation For Software Quality*, REFSQ 2005. Essener Infromatik Beiträge. pp: 71-86.

[21] S.A. O'Malley, S.A. DeLoach. "Determining when to use agent-oriented software engineering". In *Proceedings of the Second International Workshop on Agent-Oriented Software Engineering*, AOSE 2001. pp. 188-205.

[22] E. Kavakli, P. Loucopoulos. "Goal Driven Requirements Engineering: Evaluation of Current Methods". In *Proceedings of the 8th CAiSE/IFIP8.1 Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, EMMSAD 2003.

[23] C.P. Ayala, C. Cares, J.P. Carvallo, G. Grau, M. Haya, G. Salazar, X. Franch, E. Mayol, C. Quer. "A Comparative Analysis of *i**-Based Goal-Oriented Modelling Languages". In *Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering,* SEKE 2005. pp: 43-50.

[24] V.F.A. Santander, J.F.B. Castro. "Deriving Use Cases from Organizational Modeling". In *Proceedings of the 10th IEEE International Requirements Engineering Conference*, RE 2002. pp: 32-39.

[25] X. Franch, G. Grau, C. Quer. "A Framework for the Definition of Metrics for Actor-Dependency Models". In *Proceedings of the 12th IEEE International Requirements Engineering Conference*, RE 2004. pp: 348-349.

[26] M. Kolp, P. Giorgini, J. Mylopoulos. "Organizational Patterns for Early Requirements Analysis". In *Proceedings of the 15th International Conference on Advanced Information Systems Engineering*, CAISE 2003. Springer-Verlag, LNCS 2681. pp: 617-632.

[27] S. Jones, N.A.M. Maiden. "RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems". Book chapter in *Requirements Engineering for Sociotechnical Systems*, Idea Group Inc., 2004.

[28] G. Gans, D. Schmitz, M. Jarke, and G. Lakemeyer. "SNet Reloaded: Roles, Monitoring, and Agent Evolution." In *Proceedings of the 6th Workshop on Agent-Oriented Information Systems*, AOIS 2004. pp: 2-16.

[29] G. Grau, X. Franch, N.A.M. Maiden. "REDEPEND-REACT: an Architecture Analysis Tool". In *Proceedings of the 13th IEEE International Requirements Engineering Conference*, RE 2005. pp: 455 - 456.