Requirements Eng (2010) 15:267–283
DOI 10.1007/s00766-009-0087-7

ORIGINAL ARTICLE

Understanding the human context in requirements elicitation

Rubén Fuentes-Fernández · Jorge J. Gómez-Sanz · Juan Pavón

Received: 7 November 2008 / Accepted: 30 July 2009 / Published online: 14 August 2009

Abstract The human context within which a software system will operate is fundamental for its requirements. It may not appear to be very much related to the system, but it is very relevant in achieving its successful adoption. However, requirements engineers have usually a background in Software Engineering and are not trained to elicit this kind of information. This situation raises the need for analytical tools to deal with these features. These tools should enable collaborative work between requirements engineers, who use them in development, social practitioners, who provide the knowledge and processes underlying these tools, and the customers, who know the domain and intended application of the projects. The framework presented in this paper is based on the socio-psychological Activity Theory and its analysis of human contexts. It includes a repository of social properties and a process to perform this elicitation using it. The paper illustrates its application through a case study on the impact of a new system in the organization of a firm.

Keywords Software psychology · Elicitation methods · Human context · Activity Theory · Activity Checklist · Conflicts

R. Fuentes-Fernández (✉) · J. J. Gómez-Sanz · J. Pavón
Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, Madrid, Spain
e-mail: ruben@fdi.ucm.es
URL: http://grasia.fdi.ucm.es

J. J. Gómez-Sanz
e-mail: jjgomez@fdi.ucm.es

J. Pavón
e-mail: jpavon@fdi.ucm.es

1 Introduction

Requirements Elicitation (RE) is usually considered as one of the most challenging stages of a software process [1]. This is all the more evident when considering the human factor in requirements. For instance, let us see the deployment of an enterprise application in a consultancy firm. After months of development, the system is delivered to the final users, and the complaints begin. Secretaries regret the absence of their usual key shortcuts to introduce data; heads of department find it unacceptable that some reports do not require their approval; and marketing people cannot believe that in order to facilitate accounting they must provide daily details of expenses. This is not uncommon. After all, a common mistake in a development is that the application does not comply with the people's uses, the organization policies, or the implicit norms, though it does satisfy the customers' demands [2].

Although the aforementioned scenario could seem ad-hoc, this kind of problem is typical in RE, and it has proven to be pervasive, widespread, and with high costs [1]. A proper elicitation must not only capture the customers' requirements, but all the aspects of the context that can affect the system or its use in some way. In the hope of providing a suitable method for eliciting information about the human environment, solutions are based on experience gathered from past software projects and/or the incorporation of knowledge extracted from Social Sciences. However, several problems have precluded the adoption and success of these approaches:

- *Lack of expert knowledge*. Pure RE approaches have their origin in Software Engineering and they focus on the functionality the system must implement. This usually leads to the lack of specific processes for the elicitation of the human context and knowledge about what to gather. Even if these approaches consider past experiences to obtain this information [2], these are difficult to extrapolate to different scenarios. Engineers are not experts in social analysis and they are not aware of which relevant general features of the scenario should be incorporated into the elicitation technique or of how to generalize the processes that gathered them.

- *Need for social experts*. Some approaches address the previous problem incorporating social practitioners [3–5]. These practitioners can directly participate in the RE process [4, 5], but this increases the budget of the project and only the larger ones can afford it. As an alternative, these practitioners can provide and tailor the kind of methods they use to study human contexts for their application in the elicitation by requirements engineers [3, 4]. However, applying these methods requires a deep knowledge of the underlying theories, assumptions and working procedures of Social Sciences, so they are hardly applicable by people without this background.
- *Misunderstandings*. They are a key problem in RE due to the different backgrounds of customers and engineers [1, 2]. Adding social practitioners scales up this problem with a new group of specialists. In a meeting, people would be likely to use different terms, the same words with different meanings, and leave aside implicit information that they consider to be common knowledge although it is not really for the others. The representation systems and procedures of these groups are also very different, making collaboration within teams difficult.

Apart from these issues, RE also has to deal with the common gap between the textual form of requirements and their total or partial formalization [6]. Most elicitation techniques begin working with natural language or some structured form of it, like templates or restricted forms of natural language. Engineers must then translate this information to more formal representations, like the models for software development (for instance to the Unified Modelling Language [7, 8] or entity-relationship diagrams). This task implies interpretation work that is time-consuming and prone to error.

A solution to these problems could be to provide languages, tools and techniques that bridge the background gaps between the actors participating in RE, i.e. social practitioners, engineers and customers. The aforementioned works in [4, 5] address communication between engineers and practitioners specifying collaborative activities, but are still very much biased towards Social Sciences and their interests. There have also been attempts to bring requirement specifications closer to customers. Works in this line using goals [9, 10] and agent concepts [11, 12] have been relatively successful. However, these works still do not include a proper knowledge of the human factor as occurs with other approaches emerging from Software Engineering. Besides, a common issue in all the cases is that even if these groups of actors can work together in some procedures of the elicitation, it does not imply that they share the same perspective on their objectives and the meaning of the information.

In this line of research that looks for reducing the communicative gaps between experts, our work takes as its basis the Activity Theory to address these problems. The Activity Theory (AT) [13] is a socio-psychological paradigm focused on human interactions in social settings. It has already been applied and adapted to software development including RE [3, 14, 15]. Thus, it offers a suitable starting point for a RE framework for the human context. The main limitation of previous approaches to development with AT is that they rely on social practitioners or their practices to elicit the information required by engineers. Hence, they also present the previous problems regarding communication between different groups, and the need for the translation of textual requirements to software models.

The Activity Theory for Requirements Elicitation (ATRE) framework of this research is a resource to elicit requirements regarding the human context of a system. It integrates a set of analytical techniques and knowledge from AT. Its main components are a repository of *social properties* that formalizes the AT expertise in the study of human contexts, and an iterative RE process that uses it. Both the development of the repository and its application collaboratively involve different groups of experts: social practitioners, requirements engineers, domain experts and customers.

The *social properties* represent relevant information about the human context of a software system. Their development begins with social practitioners proposing some knowledge they consider should be elicited. Requirements engineers evaluate if this knowledge can be relevant for development according to research in RE and their own experience. If practitioners and engineers agree, they work together with domain experts to give a description of the social properties according to a structure with several perspectives: a *social* perspective oriented to practitioners; a *requirements* one for engineers; and several *domain* perspectives for domain experts and customers. Each perspective has textual and design descriptions. The design description uses a language called UML-AT [16] based on AT [13] and formally specified with the Unified Modelling Language (UML) [7, 8]. Mappings of concepts link the different perspectives, so users know which elements of a perspective correspond to the elements in another. This multi-perspective description facilitates the different actors understanding the information and sharing it with other experts.

The framework manages these social properties through a knowledge repository. This repository classifies and organizes the social properties according to relationships in the information they gather. For instance, a property that elicits information about a certain type of organization is linked with others that consider norms and workflows related with that organization.

The process describes the main activities to locate relevant properties for a context, to use them to create new requirements, and to navigate from the already considered properties to related ones. To create the new requirements, properties act as templates with variables that are grounded with project information. This elicitation is a process jointly performed by engineers and customers, although social practitioners and domain experts collaborate to enhance the properties according to new experiences.

The rest of the paper discusses the points in this introduction with further details. It is organized as follows. Section 2 makes a brief introduction to the underlying AT paradigm of the framework. The next sections discuss the components of the framework: Sect. 3 introduces the language UML-AT, which is the basis for the description of the social properties; Sect. 4 presents the structure for properties and the knowledge repository; and Sect. 5 describes the process to use the properties. The case study in Sect. 6 deals with the conflicts that the development of a new system for an enterprise causes with the status quo of its marketing department and how to address them. Section 7 compares ATRE with related work in RE for the human context of systems. Finally, Sect. 8 discusses some conclusions about the overall approach and the envisioned future work.

2 Activity Theory

The Activity Theory (AT) [13] is a paradigm for the analysis of human groups focused on their contextualized acts. People are embedded in a physical and socio-cultural context and their behaviour cannot be understood outside of it. At the same time, they actively interact with that environment and change it. These interactive acts with the environment are called *activities* and their contexts *activity systems*.

The *activity* [17] is a transformation process driven by people's needs. The process transforms *objects* into the *outcomes* that satisfy those needs. The active component that carries out the activity is the *subject*. Any other element used in the transformation is a *tool*. Tools shape the interaction of the subject with the environment. The *communities* [17] are the sets of subjects related directly or indirectly to the same objects. The relationships between the subject and the community are mediated by *rules* (e.g. laws, social conventions or norms) and the relationships between the community and the object by the *division of labour* [18]. In the transformation process, the division of labour establishes the role of the actors in the community, the power that they possess, or the tasks for which they are responsible. All the elements of these types related to a given activity constitute its activity system. From an AT point of view, objects, tools and outcomes can be both material and abstract elements. So the paradigm considers physical processes but also mental ones. Given these concepts, AT describes social systems as networks of activity systems interconnected by shared elements. Subjects simultaneously execute activities in these networks following their own rationality.

AT considers the hierarchical decomposition of activities. *Activities* pursue high-level *objectives* that meet people' needs. These activities are executed through sequences of *actions*, which try to achieve low-level *goals* that contribute to or are part of higher-level goals. Finally, *operations* that depend on the specific state of the environment, implement the actions.

The AT analyzes the evolution dynamics of social systems, that is, how they change over time and why. Activity systems are not static: elements appear in the environment and leave it, and existing elements modify their features. These changes cause that the activity systems previously suitable to satisfy some needs stop being it. These inadequacies produce tensions both inter and intra activity systems called *contradictions* [18]. The systems' contradictions trigger and drive their evolution, as the systems evolve looking for adaptation to the emerging situations.

Recurrent structures of neighbourhoods of activity systems, such as contradictions or organizations, play a central role in AT analysis [18, 19]. These social patterns facilitate researchers and practitioners establishing correspondences with other studies, and considering the analysis of these studies for the current situations. An example of these patterns is the introduction in [18] of the "zone of proximal development" to characterize expansive learning transitions and their mechanisms. Different researchers have later applied this pattern (or "category" in Engeström's work) to the study of contexts where similar processes appear. Nevertheless, the application of these patterns occurs in social analysis processes, whose methods do not correspond to the standard practices in RE. Applying such social processes requires that social researchers or practitioners participate in RE, which causes the problems already mentioned in Sect. 1.

3 UML-AT: a language for the human context

Originating from Social Sciences, AT has not produced a formalism to express its information. AT studies use natural language whose understanding relies on subtle interpretations on specific concepts pertaining to experts. Given that the main context of use of the ATRE results is development, the traditional descriptions of AT with natural language do not seem fully adequate for a software engineering framework. Engineers and developers usually feel more comfortable with design models, which have a clear meaning for them, and try to reduce the use of natural language to complementary explanations. For this reason, we have formalized the conceptual framework of the AT in a design language.

UML-AT [16] is a UML profile [7, 8] that defines stereotypes for the concepts and relationships of AT, and uses the Object Constraint Language (OCL) [20] to specify constraints such as the stereotypes that a relation can link. The full specification of UML-AT is available at http://grasia.fdi.ucm.es/atkoss/uml-at. Figure 1 shows an example of specification with UML-AT. It corresponds to the core elements of AT (see Sect. 2). Every concept has its stereotype of class in UML-AT; relationships are depicted as associations among those concepts that they can link. When an element of an activity system has a certain stereotype, it indicates the role that the element plays in that activity system according to AT. For instance, the elements *tool* and *activity* play the roles *Tool* and *Activity*, respectively, in the activity system of Fig. 1, and a relationship *use* links them.

Besides the elements in Fig. 1, UML-AT incorporates other concepts that do not belong to the AT core but frequently appear in AT studies. These are concepts such as artefact, and relationships such as compose, contribution and change of role. An artefact is a generic abstraction that allows the representation of any element in an activity system. Artefacts are used to represent elements that can

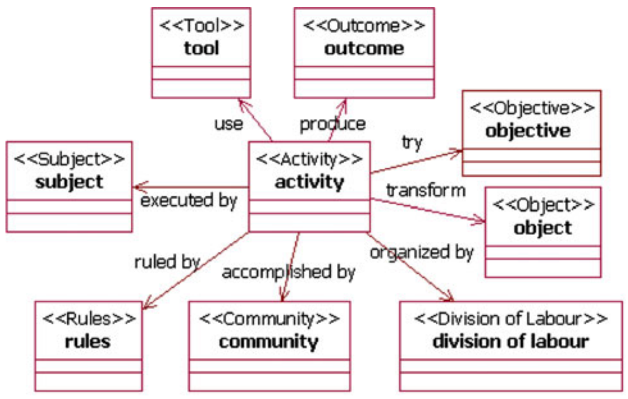


Fig. 1 Activity system with UML-AT

potentially play different roles in activity systems. For instance, an activity that produces a report can consider a document contributing to it either as its object, if its information directly appears in the outcome, or as a tool, if its information is further processed for the outcome. The correct role for that document depends on the specific instance of the activity system, so its stereotype is *artefact*, although it can be later on refined to a more specific role. A *compose* relationship indicates which components constitute an element. A *contribution* relationship shows how *artefacts* influence each other. These contribution relationships can be used to represent, for instance, the satisfaction of objectives, the construction of objects, or the damage of tools. According to their effects, these relationships can be qualified with different values such as *contribute positively*, *contribute negatively* or *impede*. Special mention should be given to the relation *change of role* that is used to highlight the change of stereotype of an element. For instance, the outcome of an activity system can become the object, tool, community, or even the activity of other activity systems. This is one of the most relevant features of the AT analysis, as it allows requirements to be studied from different perspectives.

Additional features of UML-AT are related with its application in software development. For instance, UML-AT includes adornments for relationships that can specify cardinality or optionality. UML-AT includes inheritance relationships. These relationships can be established between any pair of concepts to express that the subconcept can participate in all the activity systems where the superconcept participates replacing it. However, the subconcept can also participate in additional activity systems, and therefore its behaviour be altered with new constraints or extensions. Inheritance allows the incremental definition of social properties based on hierarchies of concepts. For instance, some social properties can describe the general concept of hierarchical organization with specializations for firms, crews or gangs.

The choice of UML-AT as the primary language for the information in ATRE achieves two objectives. First, it supports the different actors in the understanding of the requirements information. UML-AT is integrated with mainstream UML, which is widely known and used in Software Engineering. Although engineers are not experts in AT concepts, they can grasp part of their intuitive meaning from the vocabulary and relationships in UML. The language is also suitable for social practitioners, since it represents activity systems, being the names of their stereotypes and relationships taken from AT. Finally, domain experts and customers determine the vocabulary of the specific models that describe their settings. Second, the definition of UML-AT as a profile of UML allows taking advantage of the available support for UML in the development of specific tools for ATRE. This has been experienced with the Activity Theory Assistant (ATA) [16] tool, which is in the line of existing tools for model-driven engineering (MDE) [21] and development based on domain specific languages (DSL) [22]. The ATA is a graphical editor and transformation tool for UML-AT specifications. It supports a bidirectional transformation process based on declarative mappings between UML-AT and other modelling languages. This transformation process is semi-automated, with user's assistance required only to solve ambiguities in the translation. This process means that ATRE can work with UML-AT and the rest of the development process with the modelling languages of choice, so ATRE can be integrated in existing software practices.

4 Social properties

A *social property* is a piece of information extracted from Social Sciences and applicable to RE. It represents information to elicit or check about the human context according to the expertise of social practitioners. This use requires a suitable representation aimed at three goals:

- To facilitate communication between social practitioners, domain experts, customers and requirements engineers.
- To help establish mutual relationships between information and properties. For instance, a property representing a conflict points out to properties that can contribute to fixing it.
- To support the development stages after elicitation. As stated before, one of the main problems in RE is the migration of requirements to the semi-formal or formal representations used in the remaining development.

To satisfy these requirements, ATRE uses the structure in Fig. 2 for the description of social properties. Examples of its use implemented with a template appear in Figs. 3 and 4, where some parts are summarized for the sake of brevity. These properties will be used later in the case study.

The social properties are written as open questions that the RE process completes for specific projects. Their representation uses multiple perspectives, each of them targeted to the language and needs of one of the groups participating in the RE process. For instance, a *social property* can ask to describe the conflicts between the goals of a firm and those of its workers, and provide customized representations of this information with UML-AT as conflictive requirements for engineers, a social contradiction for social practitioners, and a motivational problem in the organization for managers. The structure includes three different basic *Perspectives*: for social practitioners (i.e. *Social*), domain experts and customers (i.e. *Domain*) and engineers (i.e. *Requirements*). Social descriptions are mainly used for the specification of properties. They explain the motivation of the property according to Social Sciences: what useful information can be gathered about the human context and why it may be relevant. Domain descriptions describe this information for the customers, talking about the organization, laws, or workflows relevant

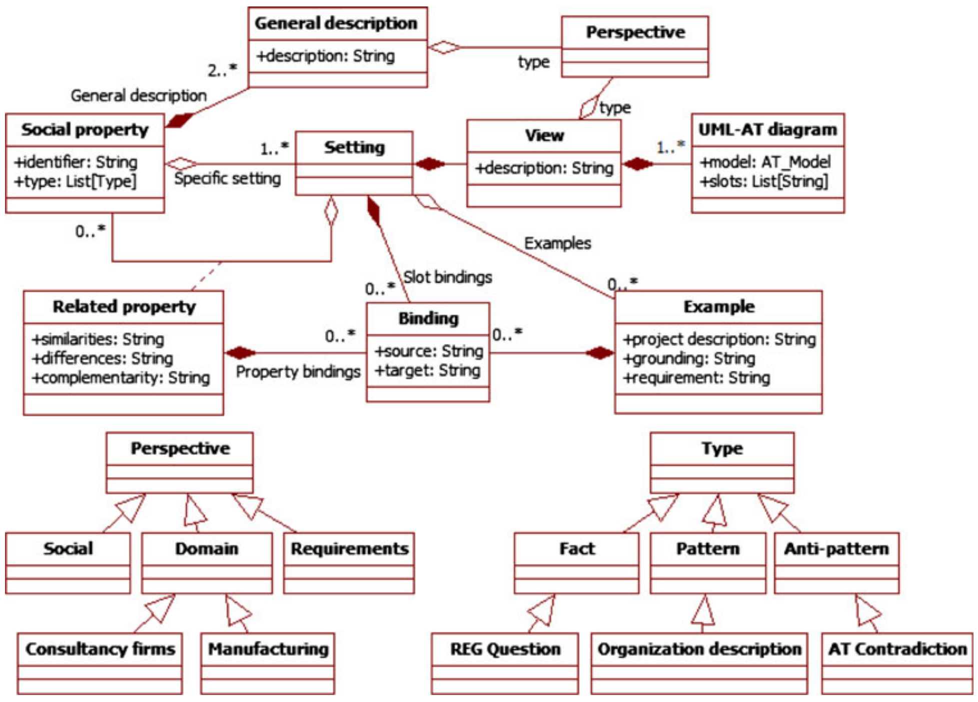


Fig. 2 Structure for social properties in ATRE

Fig. 3 Example of social property for a REG question

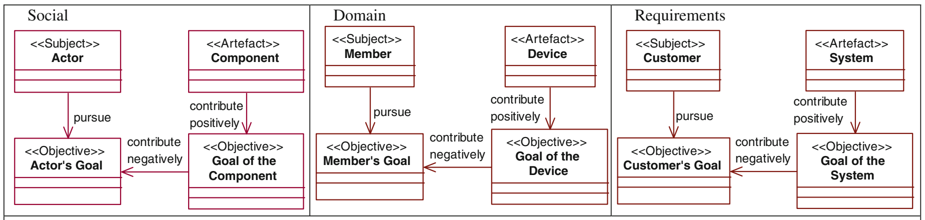
Property 1.13.6 - Type Fact

General description

Social = Does the satisfaction of the objectives of the component interfere with the satisfaction of other actor's objectives?
Domain = If the device carries out the proposed tasks, will the interests of some people in the organization be harmed?
Requirements = Will the new functionality of the system cause some negative effect over the goals of actors in the environment?

Specific setting

Social = The *Component* is intended to satisfy the *Goal of the Component*. This *objective* is contradictory with the *Actor's Goal*, which represents a need of the *subject Actor*.
Domain = The new *Device* tries to satisfy a *goal* detected in your organization *Goal of the Device*. However, it is possible that this *goal* can negatively affect some members or groups in your organization. That is to say that those groups have their own objectives, like *Member's Goal*, that cannot be satisfied (or can be with difficulty) at the same time as the *Goal of the Device*.
Requirements = The new *System* pursues the *Goal of the System*. The context of the system includes the *Customer*, which pursues, among others, the *Customer's Goal*. The satisfaction of the *Goal of the System* will imply the presence of evidences/services that reduce the possibilities of satisfaction of the *Customer's Goal*.



Slot bindings: Actor = Member = Customer; Actor's Goal = Member's Goal = Customer's Goal; Component = Device = System; Goal of the Component = Goal of the Device = Goal of the System

Examples: See the case study in section 6.

Related properties: See properties about conflicts like 4.1.5. Property bindings: [...]

in their scenario. Note that there is not an unique domain perspective, as in fact, these descriptions have to be customized for specific domains of application, such as *Consultancy firms* and *Manufacturing* in Fig. 2. Requirements descriptions tell engineers of the potential impact of each requirement in different aspects of their systems.

The structure has a unique *identifier* for every property and an indication of its *type* (more on this field later in this section). Its *General description* explains the information about the requirements that the property tries to elicit, why it should be gathered, and how it can affect other requirements.

The *Specific settings* are the key elements for gathering the information of the social properties and interpreting the requirements with a social perspective. A *Setting* considers a specific context for the elicitation of the property. It is composed of several *Views*, *Related properties* and *Examples* of use.

The *Views* provide explanations of the setting corresponding to the different perspectives. They are specified with *UML-AT diagrams* and textual *descriptions*. The UML-AT diagrams have slots for the name and type of their elements, and the value of their properties. These slots can contain fixed elements, which are inside double quotes, or variables. For instance, in the social view of the setting in Fig. 3, *Actor* and *Actor*'*s Goal* are variable slots to replace with project specific information; on the contrary, the activity "*Give Command to Subordinate*" in Fig. 8 (see the case study) indicates a non-editable slot. As these slots are shared between the different UML-AT perspectives (i.e. social, domain and requirements), they allow the exchange of information. The *Slot bindings* specify the correspondences between the slots in the different views. In general, a *Binding* indicates a mapping between a *source* and *target* slots, so they always take the same value, or an assignment of a value to a slot.

When the elicitation process applies a property, the variables in the views of one of its settings are substituted with specific information. The instantiated UML-AT diagrams of the setting represent new information for requirements in a design language. It must be noted that the process of fulfilling a setting is semi-automated. Given the specifications, the ATA tool [16] performs a pattern matching algorithm between the UML-AT representation of settings and the requirement specifications, attempting to assign already available values to the slot variables. Users can also manually edit the slots to introduce new concepts or relations.

The *Related properties* are other social properties that consider information applicable in contexts similar to the one of the setting. For instance, the setting in Fig. 4 considers a conflict due to the introduction of a new activity in an existing system; the related property 4.1.6 describes a potential modification of the activity system to solve the conflict according to AT. Besides the explanation of the relationships, there are bindings indicating possible correspondences between the related settings of the different properties. Having properties related by context, customers and engineers can depart from the current information, and find new properties to complete it or to discover additional aspects that have not yet been considered.

The *Examples* in a setting are instantiations of the requirement extracted from previous projects. They provide several elements that help to understand how the

Fig. 4 Example of social property for a contradiction of the ATCON

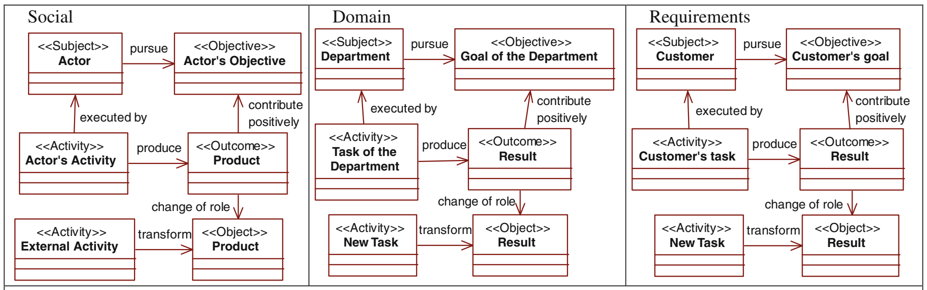
Property 4.1.5 – Type Anti-pattern

General description

Social = Need state contradiction. The outcome of the activity has changed. The subject is unable to determine if it is going to satisfy its objective.
Domain = If a new task in your organization affects a existing result or also works on it, is it possible that this result does not longer satisfy all the goals? Maybe some people are not confident about the new results or they prefer the previous situation.
Requirements = Has the new functionality got a negative impact on existing functionality, objectives or requirements?

Specific setting

Social = The *Actor* pursues the *Actor's Objective*. This *objective* is satisfied through the *Product* of the *Actor's Activity*. The introduction of an *External Activity* changes the *Product* in a non-determined way that may render it unable to satisfy the objective.
Domain = The *New Task* works on the *Result* that is currently aimed at satisfying a goal of your organization. This *Result* is the responsibility of the *Department* through the *Task of the Department* now. The *Department* makes this task to achieve its *Goal of the Department*. Do you think that as envisioned, the new task will affect the goals of the organization and the *Department*?
Requirements = The *New Task* affects the generation of the *Result*. As this *Result* is related with the *Customer's goal*, the impact of the modification in other requirements must be considered. If the changed *Result* does not fulfil all the previous requirements for the *Customer's goal*, further modifications must be introduced.



Slot bindings: Actor = Department = Customer; Actor's Objective = Goal of the Department = Customer's goal; Actor's Activity = Task of the Department = Customer's task; Product = Result; External Activity = New Task

Examples: See the case study in Section 6.

Related properties: See property 4.1.6 for a potential solution (Note: in the case study in section 6). Property bindings:[...]

setting can be used. A *project description* explains the project from which the example was extracted, providing the context to understand the problem and its human setting. The instantiation of the setting is explained with text in the *grounding* and a set of bindings between the slot variables in the views of the setting and the values used in the project. The *requirement* describes the information resulting from the application of the setting, and the meaning and impact of the obtained requirement in that context. The information in the case study of Sect. 6 could be used to provide examples for the properties in Figs. 3 and 4. The case study of a new enterprise application describes the context of the RE in the project, the reasons for selecting those properties, the answers that the requirements team provides for them and why they do it.

This format with several *settings* copes with multiple interpretations of the original textual descriptions of properties in AT studies. The *social property* has a setting for every interpretation of it. Moreover, this allows adapting the content of properties to further research and experience.

Another important aspect in the ATRE process is how to find suitable properties to analyze and discover new information. Engineers can look for properties pertaining to the current state of the requirement specifications that lead them to related information, but they can also need to explore new concerns about the system and its context. ATRE addresses this problem adopting the organization for the information of the human context proposed in the Activity Checklist [15]. ATRE organizes its knowledge repository into areas, aspects, and properties. *Areas* are insight spaces adjacent to the main concepts of AT. Each *area* includes different *aspects* that represent the information considered in that *area*. *Aspects* have related properties to elicit knowledge about them. Table 1 shows an excerpt of this structure with these *areas*:

- *Means and ends*. The relationships between users' goals and the system.
- *Environment*. The integration of the technology with requirements, resources and norms of the social and physical environment.
- *Learning, cognition and articulation*. The way in which the target technology provides support to organize the *activity* and the acquisition of knowledge about it.
- *Development*. How the use of the technology has changed the previous form of the *activity* and the foreseen evolution of the current form.

The short name of properties illustrated in Figs. 3, 4, and Table 1 corresponds to this organization of the repository. An identifier of the form *Property i.j.k.* states that the property is the *k*th of the *j*th *aspect* in the *i*th *area*.

Table 1 Partial View Of The Knowledge Repository Of Social Properties

Areas	Aspects	Social properties
1. Means/ends	1.13. Potential conflicts between target objectives	*1.13.6. "Does the satisfaction of the objectives of the component interfere with the satisfaction of others actor's objectives?"*
2. Environment	2.4. Access to tools and materials necessary to perform target actions	2.4.3. "What are the evidences that indicate that the element is not available?"
	2.8. Rules, norms, and procedures regulating social interactions and coordination related to target actions	*2.8.4. "The organization has a hierarchical structure"*
3. Learning/cognition/articulation	3.1. Monitoring and reflection through externalization	3.1.2. "How should the distributed knowledge from the component be accessible? What information should be needed to represent it?"
	3.7. Coordination of individual and group activities through externalization	3.7.5. "Must the component provide ways of giving its control to different actors?"
4. Development	4.1 Anticipated changes of target actions after the new component is implemented	4.1.4. "What activities do you think that the use of the component will make obsolete and tend to disappear from the work practices?"
		4.1.5. Need State contradiction
		4.1.6. Balanced Need State

Examples of *areas*, *aspects*, and *properties* from the ATRE repository. Properties in italics are used in the case study

Though a property can be located under different aspects in which it is relevant, this identifier is unique and corresponds to its considered main aspect.

The *type* of the properties, which was previously mentioned, complements the classification of the properties in the repository. The type is a multi-valued attribute conceived to give an indication of the kind of potential applications of the property. Its three basic values are: *fact* for properties whose purpose is gathering new descriptive information about the system; *pattern* for properties that represent configurations to keep according to the available information; *anti-pattern* for negative configurations that should be avoided or are forbidden in the system or its environment according to the available information. For instance, a property that allows specifying the type of organization of a firm is a *fact*. If that firm has a hierarchical organization, there are established power relationships between superiors and subordinates that have to be kept, and these are patterns. This type of organization frequently faces the problem of lack of motivation in the subordinated staff, and the property that describes the signs of this negative situation that should be avoided is an anti-pattern. Thus, the type gives additional guidance to choose a property in the elicitation process, depending on if the requirements team is looking for additional information or trying to guarantee that there are not problems in the already available.

Besides defining the structure for social properties and the repository, the ATRE framework needs to populate the repository with specific properties. The definition of social properties is probably the most demanding activity in ATRE. It implies discussion between the different experts to decide which information is relevant for requirements and how to describe it as a social property. A simple guideline with general activities for this development has been provided in [23]. Nevertheless, it must be pointed out that this is one of the main open issues in ATRE, since the process highly depends on the knowledge and skills of the involved experts.

ATRE currently considers two main analytical techniques derived from AT as the source for its properties. The first is the Requirements Elicitation Guide (REG) [24], which assists in discovering new information and whose theoretical basis is the previously mentioned Activity Checklist [15]; the second is the verification of potential conflicts or missed features through the identification and analysis of contradiction patterns (ATCON) [25]. Both of them use repositories of knowledge extracted from the AT and adapted to the development of multi-agent systems. They can be regarded as a restricted form of the ATRE framework for several reasons. REG and ATCON are tied to a specific software paradigm (i.e. the agent one) and more biased to analysis and design. Besides, their repositories are conceived as sets of unconnected properties. The

ATRE research has generalized these properties to its definition of social properties for RE and complemented them with new ones extracted from the original AT sources. Moreover, according to the information these properties elicit, ATRE has established the pertaining links in their related properties section.

The complete knowledge repository of ATRE social properties can be found at http://grasia.fdi.ucm.es/atkoss. It addresses several generic issues about the human context of systems. This repository comprehends the previous four areas with a total of 38 aspects. These aspects include 185 properties described with different levels of detail, of which 23 are patterns (mainly describing organization types), 11 are anti-patterns (mainly ATCON contradiction patterns), and 151 are facts (mainly REG questions). Thus, it supports RE with a very detailed account on how to inquire about a given issue of the system and its context.

5 The elicitation process

The ATRE process for requirements elicitation performs cycles where customers and engineers choose a requirements issue for analysis and add new information for it using social properties. In the line of current trends [26], ATRE conceives requirements elicitation as a stage interleaved with the rest of the development. It works over the current specifications of the requirements, part of which can emerge from the already available system architecture. Figure 5 summarizes this process.

An execution of the elicitation stage begins when customers or engineers realize that the current requirements do (or simply may) not capture all the relevant information. This recognition appears as choice 1.

In activity 2, customers and engineers discuss about the potential issues to consider for the new or refined requirements. The ATRE repository helps them with its hierarchical structure. According to the target information, customers and engineers navigate the repository from the more generic information to the most specific, that is, from areas to aspects and from these to properties. If there is not a clear idea about what can be missed in the requirements, the information in the repository may suggest introspection issues. The result of this activity is the choice of a social property to elicit.

Then, there is a discussion in activity 3 about the property in the current stage of the specifications. The general descriptions show the meaning of the properties, and the settings and their examples assist to understand the kind of information they gather. The specifications contextualize the overall discussion. For instance, customers can know what groups, restrictions or devices are already in the specifications and use them to fulfil some of the slots

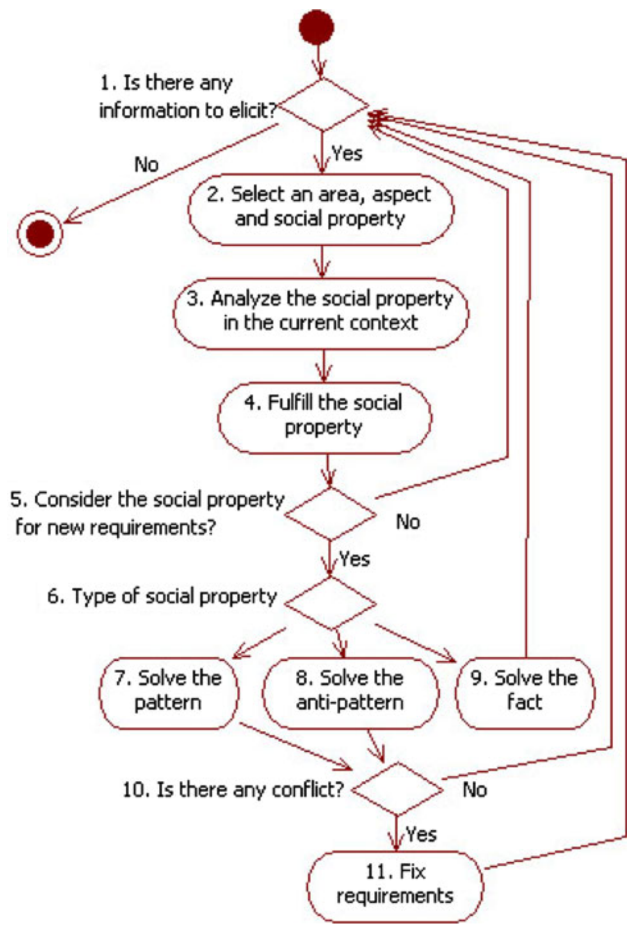

1. Is there any information to elicit?
No
Yes
2. Select an area, aspect and social property
3. Analyze the social property in the current context
4. Fulfill the social property
5. Consider the social property for new requirements?
No
Yes
6. Type of social property
7. Solve the pattern
8. Solve the anti-pattern
9. Solve the fact
10. Is there any conflict?
No
Yes
11. Fix requirements

Fig. 5 The ATRE elicitation process

in a setting of the current property in activity 4. If customers and engineers fulfil all the slots, they get an interpretation of the existing requirements that can be useful to improve their understanding of the system. For instance, they can discover underlying organizations or implicit workflow dependences. Note also that the same property can match against different parts of the specifications. If customers and engineers agree that there is a new answer for the property, they fill up in activity 4 some of the variables of one of the settings with new values. This action adds new concepts or relationships to the existing specifications. If they do not want to add new information or they discard the property, they can consider new properties from choice 5.

All the previous activities are common to the different types of social properties in the repository. Nevertheless, the steps after them differ. Activities 7 and 8, respectively, correspond to properties to keep (i.e. patterns) and to avoid (i.e. anti-patterns) in the requirements. When engineers and customers look for them in the requirements, some problem can arise, because either the specifications violate a pattern or an anti-pattern appears in them. In this case, the team can ignore the problem, maybe because it is accepted as inevitably or they will tackle with it later; they can also fix it in activity 11 using the related properties of the settings of patterns and anti-patterns. Some of these related properties point out to other properties that represent reconfigurations of the models to remove conflicts. For instance, a department that loses part of its responsibilities because automation can regard the new system as a menace and being reluctant to use it. This is a typical anti-pattern situation. A related property can propose assigning the processing of some of the new outputs of the system to that department, to compensate it and to reduce its sense of losing. Note that thanks to the property bindings (see Fig. 2), the team gets an accurate idea of the instantiation of the related properties from the current instantiation of the property under study. Of course, the customers and engineers can fix the problems in another way using their own particular knowledge of the environment. The fact properties in the repository are intended to gather additional knowledge, and not to check configurations of the requirements. In case that some problem arises with them in activity 9, customers and engineers solve it answering additional properties from activity 2.

The ATA tool [16] supports the work in the previous process. It allows customers and engineers to navigate the repository (in activities 2 and 11), examine social properties (in activities 3 and 11), perform the matchings to interpret available requirements according to AT (in activities 3 and 4), add requirements, (in activities 4, 7, 8, 9 and 11), and translate them from/into other modelling languages for the rest of the development process (not included in Fig. 5).

6 A case study: Marketing Department

The case study that illustrates the use of the ATRE framework and the ATA support tool focuses on the development of a new enterprise system for a consultancy firm. This system will provide functionality that is expected to enable new services and to change the way in which some existing tasks are carried out. As part of the elicitation, customers and requirements engineers study how functional requirements of the new system, which have already been identified, are going to affect the organization, and therefore other requirements of the system. Figure 6 offers a high-level view of part of the current requirements with UML-AT, which concerns the organization by describing its members, activities, outcomes and tools.

The firm is organized into three main departments: *Projects*, *Accounting* and *Marketing*. The "Projects Department" carries out the customers' projects. When it finishes one of them, it generates a "Final report on project". These reports have several objectives: they provide information about costs for accounting; the quality assessment program of the firm uses them to evaluate the execution of projects; they provide information on valuable consultants. Regarding the first objective, the "Accounting Department" uses the reports to generate the "Account balance". The "Marketing Department" compiles, integrates and analyzes these balances along with information obtained from external sources, and delivers it as a "Market report". The "Direction" studies the outputs of the three departments to make the decisions that govern the firm. Knowing this, all the departments consider that the more useful their information is, the more relevance they have. This relevance affects their budgets, staff, and decision power. Thus, all the departments pursue the goal "Provide valuable services" to the "Direction".

As part of the functionalities of the new system, requirements state that it will include a data-mining component that will perform an important part of the work to generate the "Market report". Engineers want to know more about the impact of this requirement. For this purpose, they use the ATRE framework with the ATA. They have already taken decision 1 in Fig. 5, as they consider that their requirements are not complete.

When a new elicitation stage begins, requirements engineers and customers navigate in activity 2 through the repository looking for properties relevant to the problem. The ATA supports them presenting the different areas, aspects and properties, and their related information. As they want to gather new information, their focus is on fact properties. Particularly, the area of "Means and ends" contains an aspect about "Potential conflicts between target objectives" with the property 1.13.6 "Does the satisfaction of the objectives of the component interfere with the satisfaction of others actor's objectives?" (see Fig. 3). This property highlights that the component aimed at satisfying a given need may also have negative side effects on the interests of actors in the environment. These effects are not just tangible products or information; they may also be on, for instance, social power, decreased motivation of participants, perceived value of assets, or constraining rules.

In activity 3, customers and engineers study the property in the current context. The ATA presents the UML-AT form of the views of the settings of the property. It also proposes concepts already available in the specifications that can replace the variables in the UML-AT diagrams. If a user selects a concept, the ATA displays its relationships with other concepts in the requirements. The user can edit the setting diagrams selecting proposed concepts. Figure 3 includes a setting for property 1.13.6. The ATA can discover potential subjects, artefacts and objectives extracted from Fig. 6 for them. Figure 7 shows Fig. 3 fulfilled. The relation "pursue" connects subjects or communities with

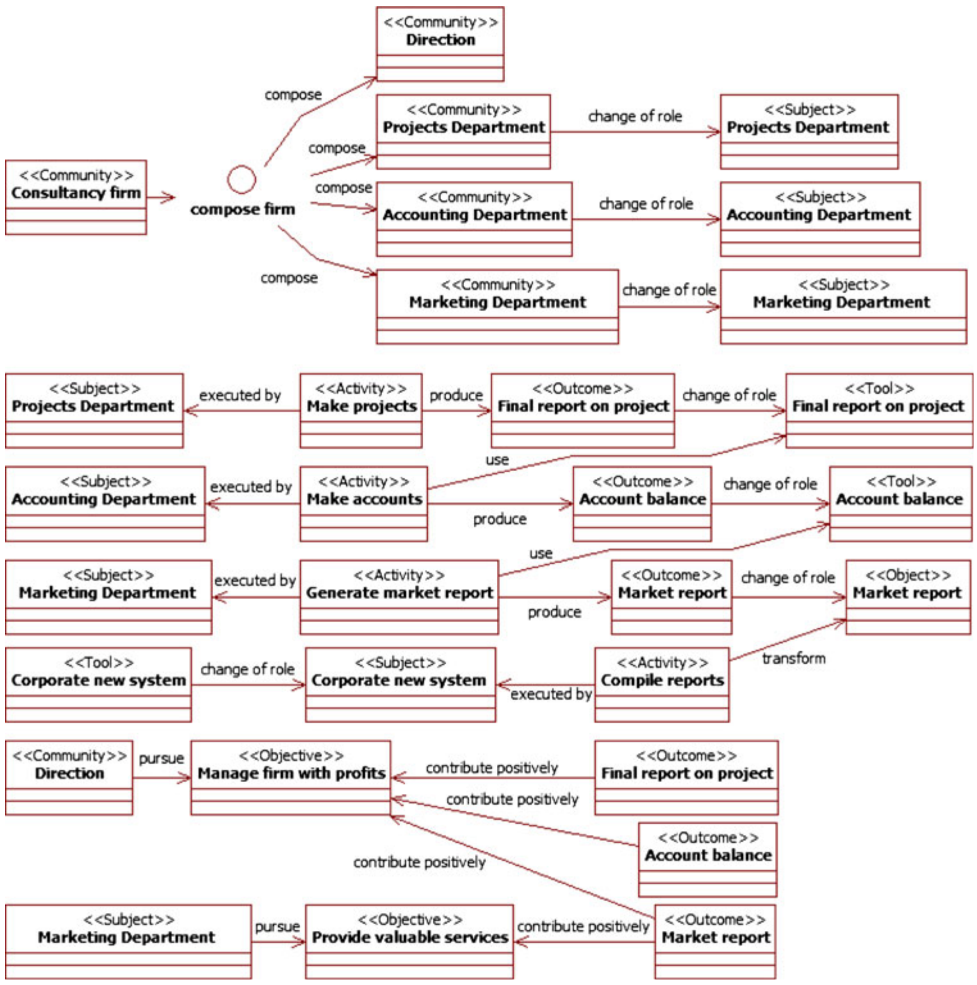

<<Community>> Direction
<<Community>> Projects Department
<<Subject>> Projects Department
<<Community>> Consultancy firm
compose firm
compose
change of role
<<Community>> Accounting Department
<<Subject>> Accounting Department
<<Community>> Marketing Department
<<Subject>> Marketing Department
executed by
produce
use
transform
<<Activity>> Make projects
<<Outcome>> Final report on project
<<Tool>> Final report on project
<<Activity>> Make accounts
<<Outcome>> Account balance
<<Tool>> Account balance
<<Activity>> Generate market report
<<Outcome>> Market report
<<Object>> Market report
<<Tool>> Corporate new system
<<Subject>> Corporate new system
<<Activity>> Compile reports
pursue
<<Objective>> Manage firm with profits
contribute positively
<<Objective>> Provide valuable services

Fig. 6 Organization of the consultancy firm and services of the new enterprise system

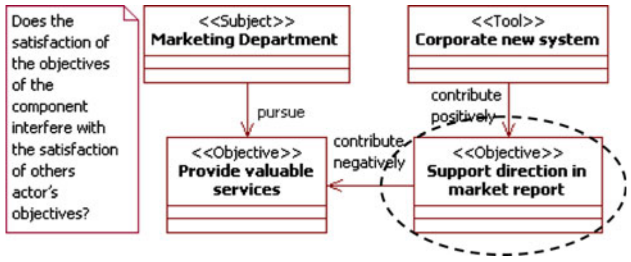

Does the satisfaction of the objectives of the component interfere with the satisfaction of others actor's objectives?
<<Subject>> Marketing Department
<<Tool>> Corporate new system
pursue
contribute positively
contribute negatively
<<Objective>> Provide valuable services
<<Objective>> Support direction in market report

Fig. 7 Fact about potential conflicts in the development of the system highlighting a new entity in the requirements

their objectives. The ATA proposes here the subject "Marketing Department" that is already linked to the objective "Provide valuable services". Note that the original upper right entity in Fig. 3 has the stereotype artefact. It represents a generic entity that can be substituted by any other modelling entity (see Sect. 3). The ATA proposes for this artefact the tool "Corporate new system". This implies a refinement of the artefact role to a tool role. Remember that this change of stereotype is a valid modification of the analysis perspective according to AT (see Sect. 2).

In this case, after examining the property, customers and engineers add new information with activity 4. They include a new objective "Support direction in market report". As they accept it as a new requirement in choice 5, the ATA records this answer in the requirements as part of activity 9.

The customers add the new objective because they consider it one of the goals of the "Corporate new system". In fact, Fig. 6 shows that the system will have the activity "Compile reports" to help in the generation of the "Market report". However, this new objective reduces the relevance of the "Marketing Department" characterized by the objective "Provide valuable services". Therefore, the answer in Fig. 7 reveals a conflict in the current situation. The functionality required for the system is seen as a menace to the current status quo by the Marketing department. In order to gain further insight in the conflict and to find a possible solution, the requirements engineers and customers consider the ATRE again, but this time focusing on the *related properties* of property 1.13.6 as part of activity 2.

The ATA presents the repository again for navigation from property 1.13.6. The area "Environment" contains the aspect "Rules, norms, and procedures regulating social interactions and coordination related to target actions" in which the property 2.8.4 poses a question about the organization of the community, which is in this case the consultancy firm. The property states that if the group is organized as a hierarchy, the subordinates must accomplish the orders of the superiors.

According to the information in the specifications, the ATA can provide in activity 3 a partial instantiation of this fact property. Figure 8 shows a fulfilled view of the setting for it where all the entities but two are suggestions of the ATA from the specifications. The objective of the Marketing department appears as the result of the new outcome that represents a command of the Direction. The activity "Give Command to Subordinate" introduced by the property generates this outcome. This property means that the "Marketing Department" must fulfil its objectives because they come from the "Direction", and the firm is a hierarchical organization.

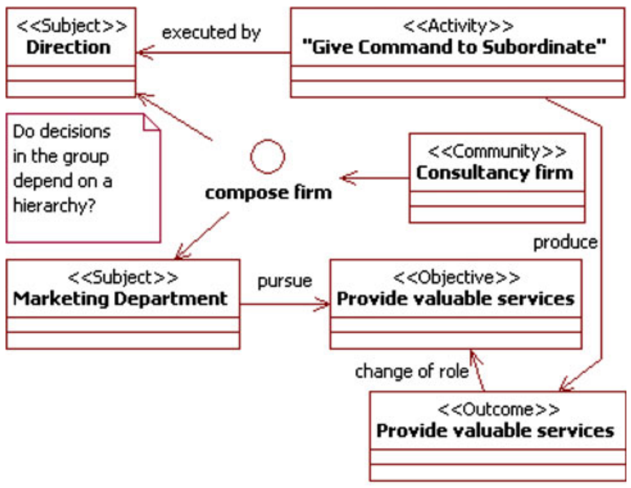
A point which should be highlighted in Fig. 8 is the use of the relationship *change of role* to point out changes in the perspective of the analysis. This relationship emphasizes how an artefact is considered in a given activity system. Here, it states that objectives are outcomes produced by the activity. Note that the community "Consultancy firm" is decomposed in Fig. 6 in four communities, two of which appear in Fig. 8 as subjects using another change of role. Besides the use of this relationship, the diagram also contains an example of constant values represented inside double quotes.

Engineers and customers discard this setting in choice 5 as not suitable for the problem. The social information in the property warns that if subjects' motivation is a concern, this pattern should only be applied in communities where subjects perceive group success as the most relevant issue. For instance, it works for a military unit in combat or a sports team competing in a key match, but not for a department or a joint venture.

Looking for a solution that is better suited to this case, customers examine another related property of property 1.13.6 in activity 2. The area "Development" contains the aspect "Anticipated changes of target actions after the new component is implemented" with the property 4.1.5. It is the AT contradiction anti-pattern known as "Need state" (see Fig. 4). The ATA once again looks for a match with the current requirements in activity 3 and finds the complete one that appears in Fig. 9. It does not show new information, but a rearrangement of the existing one that may correspond to a conflict according to AT. Its interpretation following the explanations in Fig. 4 about the "Need state" is that the Marketing department considers that its old activities may not be enough to satisfy its objectives of status in the firm when the new system works.

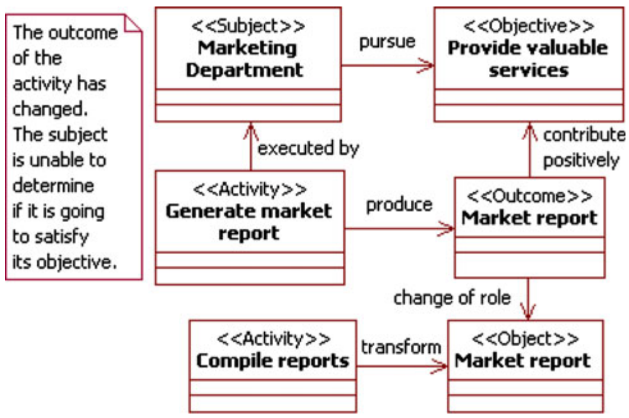
Note that when the ATA tool finds one of the contradiction anti-patterns, it does not necessarily imply a problem in the requirements. Customers and engineers must evaluate the situation in activity 8, decide whether it constitutes a problem in their context in choice 10, and if so, consider how to address it in activity 11.

Activity 11 focuses on the solution of these problems. The "Need state" property has a related property called "Balanced need state", which corresponds to a potential solution to the conflict according to AT. Its instantiation for this case using the *property bindings* proposes adding additional activities accomplished by the "Marketing Department" that are also intended to satisfy its objective. Following this line, the proposal of requirements engineers and customers is splitting the document "Market Report"



<<Subject>> Direction
executed by
<<Activity>> "Give Command to Subordinate"
Do decisions in the group depend on a hierarchy?
compose firm
<<Community>> Consultancy firm
produce
<<Subject>> Marketing Department
pursue
<<Objective>> Provide valuable services
change of role
<<Outcome>> Provide valuable services

Fig. 8 Fact about the hierarchical structure of the firm



The outcome of the activity has changed. The subject is unable to determine if it is going to satisfy its objective.
<<Subject>> Marketing Department
pursue
<<Objective>> Provide valuable services
executed by
contribute positively
<<Activity>> Generate market report
produce
<<Outcome>> Market report
change of role
<<Activity>> Compile reports
transform
<<Object>> Market report

Fig. 9 Anti-pattern about potential conflicts in the development of the system

in an automated digest generated by the system and a report on recommendations produced by the Marketing department. On doing so, the Direction can get the full set of data from the system when needed. At the same time, the Marketing department can focus on the analysis of the data, which is the task that truly requires human knowledge, and it still provides a complete and valuable report that preserves its status quo. In order to increase the acceptance of the system, it should provide support for this new task of the Marketing department, which implies eliciting new requirements for this support.

An important consideration regarding social properties is that the settings in the structure are hints for modelling the core information of a requirement, but developers may need to introduce more diagrams and elements. An example of this in the solution to this conflict would be models describing the workflows with the new pair of reports.

7 Related work

The ATRE framework is related with several lines of research in RE. It includes a conceptual and modelling framework based on AT and therefore, it is linked with research in modelling languages for the human context and the system domain. It also works on the process to elicit this knowledge, the communication between different groups of experts, and conflict management. The rest of the section discusses these issues with more details.

7.1 Modelling languages

Modelling languages introduce a bias in what is modelled or at least in its focus [27]. Thus, it is important choosing a language with a suitable support for the target information, the human context in our case. Natural language [26, 27] is probably the most popular representation for RE. It is usually the choice of users and even engineers use it largely in their documents. Moreover, it offers the highest conceptual richness and flexibility. However, there are important limitations arising from its use: misunderstandings between stakeholders, lack of precision, verbosity and difficulties for automated processing are only some of them. To reduce the negative impact of natural language specifications, RE has considered different kinds of semi-formal and formal languages that can be grouped according to their conceptual focus [1, 2, 6].

Object-oriented languages [26] are the most widely used for analysis and design of systems, and therefore they are well-known by engineers and developers. These languages offer two key advantages for RE: engineers do not need to learn a specific language for the elicitation; and using the same modelling language through all the development facilitates the integration of the elicitation with the other stages. The paradigmatic example of these languages is UML [7, 8]. It supports RE mainly with use cases [28]. Use cases include as standard elements actors, use cases, and different types of relationships between actors, actors and use cases, uses cases, and use cases with their specifications in terms of other UML diagrams. These relationships have limited semantics: generalization, participation, inclusion and specification. Other types of information that can be relevant for the human context must be specified as templates with natural language, which are not part of the standard definition of use case diagrams [28]. This limitation is shared by the other languages in this group. They do not offer specific modelling support for the human context, but they rely on natural language to describe it, which causes the already mentioned problems.

In order to overcome some of these limitations, there is work in defining modelling languages that explicitly consider some aspects of the human context. Languages for elicitation focused on goals and agents belong to this line.

Representing and reasoning about goals have a high relevance when specifying requirements [9]. Among other issues, it allows studying the motives to provide certain functionality, evaluating its affordable cost or finding potential conflicts between customers. KAOS [10] and i* [29] are relevant examples of these goal-oriented languages. KAOS [10] has defined a conceptual meta-model for goals that includes different elements related with their satisfaction, such as agents as the active elements that pursue them, actions as the way to achieve them, or constraints. Besides the graphical modelling language, KAOS has a typed temporal first-order logic to include additional information and verify properties. i* [29] focuses on the qualitative reasoning about goals. It considers *soft goals* as those whose satisfaction cannot be precisely stated in terms of other artefacts. Examples of these soft goals are the customer satisfaction or the usability of an interface. In order to reason about these soft goals, i* defines contribution relationships (i.e. positive/negative enough/partial AND/OR) between goals and of these with other elements such as agents, tasks and resources. Although the concepts considered in these goal-oriented approaches are relevant when modelling the human context, they are not the only aspects to consider. For instance, these approaches commonly disregard the organization of the society, its norms, and how these affect its individuals. KAOS only offers hierarchical groups of agents and i* just includes the refinement of actors as new actors. ATRE considers these issues specializing the community concept. New communities are defined with a set of related social properties that establishes their features. A partial example of this corresponds to Fig. 8, which introduces a norm of hierarchical organizations. Nevertheless, ATRE does not currently consider additional logical languages for the fine-grained specification of the human context, although it can use OCL [20] to specify some constraints. Another issue to consider about these languages is the definition of the goal concept itself and the role that its different types play in RE [30]. Depending on the selected approach, goals are used to describe the current and envisioned systems, the organizational context or the development process. The understanding and characterization of these types would enable in ATRE a more precise definition of the meaning of some social properties. This is especially relevant for the properties related with the area 4 *Development* (see Table 1) about change management. At the same time, this kind of research could benefit of the work in AT about the differentiation between objectives, goals and context (see Sect. 2) as means to evaluate the relevance of different goals and support conflict management.

Agent-oriented approaches have also played a relevant role in RE. They facilitate the description of responsibilities and interactions among components to achieve the functionalities related with the system. At the same time, since agents are intentional and social abstractions, they can be used for an integrated study of the system and its environment [2]. Some illustrative approaches in this line are Albert [11] and Tropos [12]. Albert [11] is a RE framework that includes graphical modelling languages, a temporal modal logic and some guidelines for the elicitation. Tropos [12] is a complete agent-oriented methodology for development, whose conceptual framework is an extension of i* [29]. Its additions include temporal modal logics to reason about the specifications, and concepts such as capabilities, events or plans to characterize the agents. These agent-oriented approaches enrich goal-oriented ones with the detailed specification of the involved agents. Their purpose is describing how agents reason and interact to satisfy their goals according to their knowledge about the world and themselves. This information allows describing relevant parts of the human context. Nevertheless, these approaches also fail to consider the society as a first class citizen in models and are not flexible to represent new social constructions. If they consider groups of agents, they do it as pre-defined types of organizations whose instantiations have linked specific elements (e.g. goals, resources or capabilities). The norms of these societies are usually represented using some kind of modal logic. As stated before, ATRE adopts a definition of communities and norms based on general social properties. Besides the flexibility, our approach offers the advantage of being understandable by non-engineers. The specific perspectives for the different groups of stakeholders make understandable the represented aspects of the human context without requiring expertise in agent abstractions or logics. As a drawback of ATRE, these social properties are not suitable for the formal or semi-formal specification of fine-grained issues of the social systems without further extensions.

Complementary to previous approaches are scenarios [31]. Scenarios represent paths of behaviour in the system and therefore are closely related with its dynamics. The current vocabulary of UML-AT is focused on activities and it can be regarded as a modelling language for these paths. For instance, UML-AT and scenarios in [31] have an important set of similar concepts. This could allow the incorporation of part of the available support for checking scenarios to ATRE, and at the same time, the use of social properties and the kind of process proposed in ATRE to validate scenarios from a social point of view.

Some works [3, 14, 15, 32–34] have adopted the AT concepts to work directly in RE. The issue here is that they also adopt the practices of Social Sciences and work mainly in natural language. When using some kind of semi-formal language, they adopt the Engeström's traditional depiction for activity systems [18]. It represents activity systems as triangles with elements in their vertexes and the middle point of its sides. Although this representation is common for social researchers and practitioners, it has several problems to represent networks of activity systems. It makes difficult to distinguish the different artefacts that participate in each of the roles in an activity system, it lacks of a standard representation for shared artefacts, and it does not make explicit the needs satisfied by the system. The depiction of activity systems in UML-AT with entities and relationships overcomes these limitations. Besides, UML-AT makes explicit additional AT concepts that with the traditional depiction can only be represented in natural language, like contribution relations or community components. The Brahms framework [14] is an exception in this trend. It has developed a programming and simulation language with the main concepts of AT that also covers the specification of some constraints. Its main drawback is that using a programming language for RE makes difficult the involvement of non-engineer stakeholders. Anyway, stakeholders find difficult the study of complex textual specifications if there are not facilities to perform a partitioned analysis of them (e.g. views, aspects or workflows).

Independently of the conceptual focus, there is a trend towards using domain-specific languages to cover specific issues of RE [6]. These languages provide native facilities for modelling closer to those common in the customer's domain. Examples of these languages are some standards for telecommunications [35] or the work about railway control in [36]. ATRE can be seen as framed in this trend, since it adopts the AT framework from Social Sciences as its basis. However, these languages are usually developed from scratch [37]. On the contrary, ATRE regards trends in

MDE [21] and defines UML-AT as a UML profile. In this way, ATRE benefits from the available support for UML in the development of its tools [16].

7.2 Processes

A suitable modelling language is not enough for a proper elicitation of knowledge about the human context. There is also need for a process that guides such acquisition. For this purpose, RE has proposed a variety of elicitation, modelling and analysis techniques integrated in methodologies and processes [2, 6]. They present some limitations to work with the human context.

Methodologies and processes are usually specified from the point of view of engineers. Examples of these are specific processes for elicitation like [38, 39], but also the more general elicitation activities included in the Unified Software Development Process [40]. All of them describe processes where engineers take the initiative and customers are collaborators with the passive role of information providers. These approaches preserve the mutual misunderstandings between the stakeholders common in RE, given that the processes themselves introduce the bias of the engineers who drive them [41].

Some alternatives that embed in the elicitation process activities oriented to give a higher relevance to stakeholders different than engineers have been proposed, mainly in ethnographic informed elicitation. The assumptions underlying ethnography are that a true understanding of a group activity can only be gained from within that group, as its social organization depends on tacit knowledge and implicit practices that are not usually obvious to the casual observer. Participatory design [3, 42], the Coherence method [43] or the Goguen and Linde' work [4] appear in this line. The main limitation of these approaches is that they lack of well-defined processes. Although intended for RE, most of them [3, 4, 42] only offer highly abstract processes described with natural language. Implementing one of such processes requires a background in Social Sciences that common engineers and customers do not have. The closer work in this line to ours is the Coherence method [43]. Its goal is that engineers perform a social-aware analysis guided by ethnographic expert knowledge. This expert knowledge takes the form of questions in natural language grouped by concerns. Engineers record the answers to the questions with standard templates. These templates use natural language complemented with UML diagrams [7, 8]. The ATRE social properties also gather the expert knowledge of Social Sciences but they differ from the Coherence questions in several ways. ATRE links social properties according to the relationships in the information they gather. Thus, it provides support for a process driven by the information needs emerging from the own requirements. This guide is also a key differentiating factor from the other previous approaches in this line. Moreover, each social property considers potential representations with UML-AT. Modelling requirements is a conflictive stage that implies interpreting information and expressing it in semi-formal or formal languages, making explicit the details required for the rest of the development stages [6]. Since social properties include initial models for their requirements from different perspectives, they help to reduce the potential misunderstandings and the gap with other development models in these activities.

Another important issue to consider in the elicitation process is how to manage communication between the stakeholders. Modelling languages do not solve per se this problem, since the stakeholders can have different and even conflictive perspectives [2]. The management of these perspectives is commonly done through explicit representations of the different viewpoints [1, 44]. This is the approach adopted by social properties with the social, domain and requirements perspectives. When compared with viewpoints in [44], ATRE does not include particular mechanisms to check consistency among perspectives, like the use of logics. It relies on the detection of social properties in the requirements for this purpose. Anti-pattern social properties represent conflicts with a social basis and therefore they do not only correspond to inconsistencies in the requirements but to true conflicts between stakeholders. As a limitation, this mechanism is not well suited to detect modelling inconsistencies such as the lack of refinements for some diagrams or unfulfilled constraints expressed with constraint languages.

Closely related with communication is the issue of conflict solution. Conflicts appear in requirements at several levels that must be considered. First, there can be inconsistencies in the requirements caused by mistakes in the elicitation. Approaches to this problem usually consider a partial formalization of requirements that enables their automated checking [44, 45]. Even with consistent requirements, there is a second level of conflict when the same or different stakeholders have conflicting goals or perceptions of the problem. In this case, available approaches [46, 47] mainly focus on the proper identification of the context of the conflict (e.g. precedence of involved objectives, affected stakeholders or type of the information). Then, they propose a standard process to address conflict solution through negotiation between stakeholders. ATRE can complement these approaches with the use of its social properties. AT provides its expertise in social conflicts and their solutions through its contradictions. ATRE codifies this information as social properties, where conflicts and their solutions are linked as related properties.

8 Conclusions

The ATRE framework is an analytical tool for requirements elicitation about the human context of systems. It is built upon a well-established theory from Social Sciences, the AT, and standard practices of Software Engineering.

ATRE abstracts and formalizes the concept of *social property*. A social property presents knowledge from Social Sciences that can be relevant in gaining new insights into the human context of a system. This knowledge is offered through three types of perspectives in each social property: one targeted at social practitioners who have knowledge on social issues; another for requirements engineers who describe the requirements for future development; and particular perspectives for domain experts and customers who know the target domains. These perspectives are interconnected through mappings of their elements as means to improve collaboration and share information.

A knowledge repository stores these properties, which are organized in areas that are related to dimensions of concern in AT and aspects that refine them. This structure guides users to properties related to their current information interests. The knowledge repository now includes 4 areas, which contain 38 aspects with 185 properties, but can be extended with practice.

ATRE proposes an elicitation process to use this repository. It indicates how customers and engineers must look for social properties according to the information they are interested in eliciting. The use of properties allows them adding new information, and also checking that requirements satisfy certain social patterns and do not present undesirable anti-patterns.

The main concern in the application of the ATRE framework is the difficulty in elaborating the properties, as they require the collaboration of heterogeneous teams on wide topics. Although there is a process to develop social properties, it only defines general activities and strongly depends on the knowledge and skills of the participants to determine the relevant information and specify it in the proper format. Nevertheless, this is an effort made once for a given property. The property can then be extended for different domains just adding another specific domain perspective. Each version for a domain can be reused in multiple projects. Another point is that users commonly bias requirements to the properties in the repository, forgetting that it cannot be complete. Users have to be encouraged to consider potential requirements others than these properties, which in principle are mainly considering human and social issues. Finally, even with the structure of the repository, it may be difficult to find the properties relevant at a given setting. We are currently studying the use of case-based reasoning to suggest the potential properties to consider and assess how they can improve the requirements.

In conclusion, the result of using the ATRE framework are requirements specifications that are more complete regarding the human context and its influence in the design and behaviour of the system to be. Even apparently simple properties reveal requirements that may be neglected or difficult to discover in real specifications, which would raise problems in the final product. An additional advantage is that the ATRE process is supported by tools, with the consequent effort saving in requirements elicitation when compared with other techniques.

References

1. van Lamsweerde A (2000) Requirements Engineering in the Year 00: a research perspective. In: Proceedings 22nd international conference on software engineering (ICSE-2000). ACM Press, pp 5–19
2. Nuseibeh BA, Easterbrook SM (2000) Requirements engineering: a roadmap. In: Proceedings 22nd international conference on software engineering (ICSE-2000). ACM Press, pp 35–46
3. Bødker S, Grønbæk K (1996) Users and designers in mutual activity: an analysis of cooperative activities in system design. In: Engeström Y, Middleton D (eds) Cognition and communication at work. Cambridge University Press, pp 130–158
4. Goguen JA, Linde C (1993) Techniques for requirements elicitation. In: Proceedings of IEEE international symposium on requirements engineering (RE 1993), pp 152–164
5. Hughes J, King V, Rodden T, Andersen H (1994) Moving out from the control room: ethnography in system design. In: Proceedings of ACM 1994 conference on computer supported cooperative work (CSCW 1994). ACM Press, pp 429–439
6. Cheng BHC, Atlee JM (2007) Research directions in requirements engineering. In: Proceedings future of software engineering (FOSE'07), IEEE Computer Society, pp 285–303
7. OMG (2007) OMG Unified Modeling Language (OMG UML), Infrastructure, V2.2. 4 Feb 2009, http://www.omg.org
8. OMG (2007) OMG Unified Modeling Language (OMG UML), Superstructure, V2.2. 4 Feb 2009, http://www.omg.org
9. van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. In: Proceedings 5th IEEE international symposium on requirements engineering (RE 2001), pp 249–262
10. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20:3–50
11. Dubois E, Du Bois P, Dubru F, Petit M (1994) Agent-oriented requirements engineering: a case study using the ALBERT language. In: Proceedings 4th international working conference on dynamic modelling and information systems (DYNMOD'94), pp 205–238
12. Castro J, Kolp M, Mylopoulos J (2002) Towards requirements-driven information systems engineering: the Tropos project. Inf Syst 27(6):365–389
13. Vygotsky LS (1978) Mind and society. Harvard University Press, Cambridge, MA
14. Clancey WJ, Sachs P, Sierhuis M, van Hoof R (1998) Brahms: simulating practice for work systems design. Int J Hum Comput Stud 49(6):831–865
15. Kaptelinin V, Nardi BA, Macaulay C (1999) The Activity Checklist: A tool for representing the 'space' of context. Interactions 6(4):27–39

16. Fuentes-Fernández R, Gómez-Sanz JJ, Pavón J (2007) Model integration in agent-oriented development. Int J Agent Oriented Softw Eng 1(1):2–17
17. Leontiev N (1978) Activity, consciousness, and personality. Prentice-Hall, Englewood Cliffs, NJ
18. Engeström Y (1987) Learning by expanding. Orienta-Konsultit
19. Kuutti K, Arvonen T (1992) Identifying potential CSCW applications by means of activity theory concepts: a case example. In: Proceedings ACM 1992 conference on computer-supported cooperative work (CSCW 1992), pp 233–240
20. OMG (2006) Object Constraint Language, Version 2.0. 1 May 2006, http://www.omg.org
21. Schmidt DC (2006) Model-driven engineering. IEEE Comput 39(2):25–31
22. Sierra JL, Fernández-Valmayor A, Fernández-Manjón B (2008) From documents to applications using markup languages. IEEE Softw 25(2):68–76
23. Fuentes R, Gómez-Sanz JJ, Pavón J (2005) Requirements elicitation for agent-based applications. In: Proceedings 6th international workshop on agent-oriented software engineering (AOSE-2005), Lecture Notes in Computer Science, vol 3950. pp 40–53
24. Fuentes-Fernández R, Gómez-Sanz JJ, Pavón J (2009) Requirements elicitation and analysis of multiagent systems using activity theory. IEEE Trans Syst Man Cybern A Syst Hum 39(2):282–298
25. Fuentes-Fernández R, Gómez-Sanz JJ, Pavón J (2007) Managing contradictions in multi-agent systems. IEICE Trans Inf Syst E90-D(8):1243–1250
26. Sommerville I (2005) Integrated requirements engineering: a tutorial. IEEE Softw 22(1):16–23
27. Al-Rawas A, Easterbrook S (1996) Communication problems in requirements engineering: a field study. In: Proceedings of the conference on professional awareness in software engineering (PASE'96), pp 47–60
28. Bittner K, Spence I (2002) Use case modeling. Addison-Wesley Professional
29. Mylopoulos J, Chung L, Yu E (1999) From object-oriented to goal-oriented requirements analysis. Commun ACM 42(1):31–37
30. Kavakli E (2002) Goal-oriented requirements engineering: a unifying framework. Requirements Eng 6(4):237–251
31. Sutcliffe AG, Maiden NAM, Minocha S, Manuel D (1998) Supporting scenario-based requirements engineering. IEEE Trans Softw Eng 24(12):1072–1088
32. Martins LEG, Daltrini BM (1999) An approach to software requirements elicitation using precepts from activity theory. In: Proceedings 14th IEEE international conference on automated software engineering (ASE'99), pp 15–23
33. Collins P, Shukla S, Redmiles D (2002) Activity theory and system design: a view from the trenches. Comput Supported Coop Work 11(1):55–80
34. Zurita G, Nussbaum M (2007) A conceptual framework based on Activity Theory for mobile CSCL. Br J Educ Technol 38(2):211–235
35. International Telecommunication Union (1999) Recommendation Z.100 (11/99). CCITT Specification and Description Language
36. Haxthausen AE, Peleska J (2002) A domain specific language for railway control systems. In: Proceedings 6th biennial world conference on integrated design and process technology (IDPT 2002), p 7
37. van Deursen A, Visser J (2000) Domain-specific languages: an annotated bibliography. ACM Sigplan Notice 35(6):26–36
38. Parnas DL, Weiss DM (1987) Active design reviews: principles and practices. J Syst Softw 7(4):259–265
39. Potts C, Takahashi K, Antón AI (1994) Inquiry-based requirements analysis. IEEE Softw 11(2):21–32
40. Jacobson I (2000) The road to the unified software development process. Cambridge University Press, London
41. Goguen J, Jirotka M (1994) Requirements engineering: social and technical issues. Academic Press, San Diego, CA
42. Muller MJ (2003) Participatory design: the third space in HCI. In: Jacko JA, Sears A (eds) The human-computer interaction handbook: fundamentals, evolving technologies, and emerging applications. Lawrence Erlbaum Associates, Mahwah, pp. 1061-1082
43. Viller S, Sommerville I (1999) Social analysis in the requirements engineering process: from ethnography to method. Integrating ethnography into the requirements engineering process. In: Proceedings 4th IEEE international symposium on requirements engineering (RE 1999), pp 6–13
44. Nuseibeh B, Finkelstein ACW (1992) ViewPoints: a vehicle for method and tool integration. In: Proceedings of the IEEE international workshop on computer-aided software engineering (CASE-92), pp 50–60
45. Hausmann JH, Heckel R, Taentzer G (2002) Detection of conflicting functional requirements in a use case-driven approach: a static analysis technique based on graph transformation. In: Proceedings 24th international conference on software engineering (ICSE 2002), pp 105–115
46. Easterbrook S (1994) Resolving requirements conflicts with computer-supported negotiation. In: Jirotka M, Goguen J. (eds) Requirements engineering: social and technical issues. Academic Press, San Diego, CA pp 41–65
47. Boehm B, Bose P, Horowitz E, Lee MJ (1995) Requirements negotiation and renegotiation aids: a theory-w based spiral approach. In: Proceedings 17th international conference on software engineering (ICSE-17), pp 243–254