Information and Software Technology 59 (2015) 109–135

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Synergy between Activity Theory and goal/scenario modeling for requirements elicitation, analysis, and evolution

Geri Georg [a,*], Gunter Mussbacher [b], Daniel Amyot [c], Dorina Petriu [d], Lucy Troup [e], Saul Lozano-Fuentes [f], Robert France [a]

[a] Computer Science Department, Colorado State University, Fort Collins, CO, USA
[b] Department of Electrical and Computer Engineering, McGill University, Montreal, Canada
[c] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
[d] Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
[e] Department of Psychology, Colorado State University, Fort Collins, CO, USA
[f] Department of Microbiology, Immunology and Pathology, Colorado State University, Fort Collins, CO, USA

ARTICLE INFO

Article history:
Received 17 December 2013
Received in revised form 11 October 2014
Accepted 6 November 2014
Available online 13 November 2014

Keywords:
Requirements engineering
Activity Theory
User Requirements Notation
Goal modeling
Scenario modeling

ABSTRACT

Context: It is challenging to develop comprehensive, consistent, analyzable requirements models for evolving requirements. This is particularly critical for certain highly interactive types of socio-technical systems that involve a wide range of stakeholders with disparate backgrounds; system success is often dependent on how well local social constraints are addressed in system design.
Objective: This paper describes feasibility research, combining a holistic social system perspective provided by Activity Theory (AT), a psychological paradigm, with existing system development methodologies and tools, specifically goal and scenario modeling.
Method: AT is used to understand the relationships between a system, its stakeholders, and the system's evolving context. The User Requirements Notation (URN) is used to produce rigorous, analyzable specifications combining goal and scenario models. First, an AT language was developed constraining the framework for automation, second consistency heuristics were developed for constructing and analyzing combined AT/URN models, third a combined AT/URN methodology was developed, and consequently applied to a proof-of-concept system.
Results: An AT language with limited tool support was developed, as was a combined AT/URN methodology. This methodology was applied to an evolving disease management system to demonstrate the feasibility of adapting AT for use in system development with existing methodologies and tools. Bi-directional transformations between the languages allow proposed changes in system design to be propagated to AT models for use in stakeholder discussions regarding system evolution.
Conclusions: The AT framework can be constrained for use in requirements elicitation and combined with URN tools to provide system designs that include social system perspectives. The developed AT/URN methodology can help engineers to track the impact on system design due to requirement changes triggered by changes in the system's social context. The methodology also allows engineers to assess the impact of proposed system design changes on the social elements of the system context.

1. Introduction

A prime goal of investment in requirements engineering (RE) is to produce a comprehensive, consistent set of system requirements covering various aspects of the system, e.g., operational environment constraints, general functionality requirements, and so-called non-functional requirements such as performance and security. Since systems and their requirements evolve over time, system designers also benefit from requirements models that can be easily evolved and that provide rationale for design decisions, where rationale includes the source of a requirement such that its continued relevance can be tested over time as the system evolves. The psychological framework of Activity Theory (AT) [14] may be used to help identify the societal constraints that need to be addressed

* Corresponding author.
E-mail addresses: georg@cs.colostate.edu (G. Georg), Gunter.Mussbacher@mcgill.ca (G. Mussbacher), damyot@eecs.uottawa.ca (D. Amyot), petriu@sce.carleton.ca (D. Petriu), Lucy.Troup@colostate.edu (L. Troup), slozano@colostate.edu (S. Lozano-Fuentes), france@cs.colostate.edu (R. France).

http://dx.doi.org/10.1016/j.infsof.2014.11.003
0950-5849/

by a system in order to help ensure its acceptance and success. AT systematically leads the designer through a process that can identify previously unknown stakeholders and their implicit social constraints as part of requirements elicitation. This is particularly critical for certain types of socio-technical systems that employ citizens to collect the data used by officials to, for instance, make policy. These systems can be highly interactive and often involve a wide range of stakeholders with very different backgrounds. Multiple studies indicate that the success of these types of systems often depends on how well local social constraints have been taken into account and addressed in the system design, as discussed below.

An example of a socio-technical system is one that Dr. Lozano-Fuentes has helped develop in collaboration with universities and public health officials in Mexico [13]. This system is designed to support collecting vector surveillance (in this case mosquito breeding site) data, which is used to provide evidence for health officials deciding vector control policy. Currently public health workers collect data regarding vector populations, but with the advent of mobile devices, the general public can also collect this data. In many parts of the world, vector surveillance is critical to the control of diseases such as Dengue, a mosquito-borne disease. Spiegel et al., van den Berg et al., and Vanlerberghe et al. report studies of different surveillance systems (both automated and manual) that have been deployed to help track mosquito vectors [46,49,50]. These authors report that the few successful programs addressed local community social issues, and that the local community drove and participated in data collection in these successful programs. In these socio-technical systems, the quantity of timely data, in addition to its accuracy, is critical to support vector control decisions. We also note that the general population from whom the data is gathered is affected by the system results in the form of the vector control decisions (e.g., when and where to spray for mosquitos).

Another example of a socio-technical system is that of home automation. Again, the general public interacts with the system: perhaps controlling all or portions of it, acting as sources of data for it through their actions, and being affected by it as the system makes changes in the home. Studies of home automation discuss some of the challenges in the adoption of these systems. For example, Brush et al. note the large diversity of target users, and cite barriers to adoption that include a tradeoff between convenience for users and security functions [8]. The study by Mennicken and Huang identified that it can be difficult for users to imagine and plan for the changes in their daily routines that come with home automation. Users sometimes chose to perform tasks without automation in response to these changes [34]. Takayama et al. identified social issues such as friction between household members regarding automation, anxiety about the system, and cognitive issues related to getting the system to work properly [47]. Sadri, in a survey of intelligent automation in several areas including home automation, cites concerns over security and social/ethical implications of these systems and the data they collect [44].

Our research is aimed toward the application of social sciences theories to the problem of determining the social requirements of systems such as the above, where data is collected directly or indirectly from the general public and is used to control some noticeable aspect of their environment. Our goal is to apply social sciences theories to identify these requirements in ways that allow them to be taken into account in system design, with the expectation that addressing them will help improve the successful deployment and use of these types of systems. We have identified several important requirements for such a social theory: (1) it must have prior successful use in the social sciences in identifying a system's social constraints and explaining how addressing them or not affected the system, (2) it should be used to at least some extent in the field of software development, (3) it must be adaptable for use in software development by persons who are not social scientists, (4) the adapted theory must produce repeatable results that are supported by automation as much as possible, (5) methodologies must be identified that allow the adapted theory results to be used in conjunction with existing system development methodologies and tools, and (6) the adapted theory results must identify social constraint requirements that might not have been found otherwise.

The second requirement may be relaxed to include social theory that has not previously been applied to the field of software development. However, if the social theory has been used before in such a context, it is more likely that it is usable by non-social scientists as there is at least some evidence to this effect. The sixth requirement is one of validation against other comprehensive system development methodologies, and has not yet been completed.

In this context, the main research question of this paper is: *can Activity Theory (a particular social science theory) be adapted, formalized, and combined with an existing system development methodology to support the repeatable discovery of non-obvious social requirements in socio-technical systems with local constraints*? The satisfaction of the above six requirements will result in a positive answer to this research question.

The contribution of this paper is to show that AT fits the first five requirements. We briefly discuss how AT meets the first two requirements, using examples from the social sciences and limited use of AT concepts in systems development, particularly in Human-Computing Interaction (HCI) development. We then focus on the technical aspects of the above requirements, specifically how we can constrain AT to be used by non-social scientists, how much we can automate its use to produce repeatable results, and how we can integrate it into an existing system design methodology and tools. To address this last issue, we have developed an RE methodology that leverages synergistic opportunities provided by the human activity formulations embodied in AT and the formal definitions of goal and scenario modeling of the User Requirements Notation (URN) [2,23]. The URN tool jUCMNav, developed at the University of Ottawa, is the tool we have used for the work described in this paper [24].

Our combined methodology can be used during initial system development and also, through tracing capabilities, it provides a way to monitor the effects of potential changes in the system. For example, changes might occur in the stakeholders' social constraints that can affect the system. These changes can provide a basis for planning system evolutions. Our method provides a mechanism to explore changes across basic objectives, stakeholders, and social constraints that occur over time, including their effect on the overall system and individual user needs. The combination of AT and URN allows the social constraints that are identified through AT to be traced back and forth to their respective places in goal models and in system design. This bidirectional tracing can be used to validate that a system design (or an evolved design) preserves social constraints that are still relevant for the system or to identify new social constraints that must be addressed in evolving designs.

We present a short introduction to AT and URN as well as the rationale for using them in Section 2. Section 3 describes how we rigorously defined an AT language for use in requirements engineering and how it is formally mapped to URN. Section 4 describes one possible methodology using AT and URN that we demonstrate in the paper. Section 5 describes the example system based on the Mexico Vector Surveillance system that we have used as a proof-of-concept for our approach and how both AT and URN are applied to it. Section 6 demonstrates how our approach can be used in the context of system evolution to create a possible extension to the example system. We discuss the results of our approach in the context of the example system, including the complementary nature of AT and URN, as well as our experiences, lessons learned, and threats to validity in Section 7. In Section 8, we contrast our

approach with related work, and in Section 9 we present our conclusions and outline planned future work.

2. Activity Theory and User Requirements Notation background

AT was initially proposed [29,52] to aid exploration of the complex social relationships inherent in any human activity. It was extended [14] and has also become a useful theoretical framework in the field of Human Computer Interaction (HCI) [19]. AT can be useful in situations where it is necessary to explore the diverse and complex social context of a system, particularly those with both human and computing components. It encourages discovery of all community and societal influences that may affect a system, not just those associated with a particular group of workers, and uses a structure and common language that can encourage discussion by diverse groups of stakeholders [18].

2.1. AT framework definitions

AT defines human activity as a system of several elements and their mediating relations. Fig. 1a shows the diagrammatic form of an activity system developed by Engeström [14]. We term this diagram an Activity System Diagram (ASD), and have used it as a basis for the AT language we developed (please see Section 3 for the language metamodel and details).

ASD elements:

- **Aim/Object:** Every human activity has an *aim*, or *object* (either physical or conceptual), which is why the activity is being pursued. There can be more than one aim associated with a single activity. We note that the use of the term *object* presents a potential source of confusion when it is used to refer to an AT item, especially by persons familiar with the software development domain, so we use the term *aim* for this concept. However, *object* is the term used in the AT literature and research dating from Vygotsky [52].
- **Outcome:** An *outcome* is produced as the result of executing the activity.
- **Subject:** The activity is performed by the *subject*. There can be multiple subjects engaged in a single activity.
- **Tools:** The subject performs the activity using a mediating *tool*. There can be multiple tools used to perform a single activity, and tools may be either physical or conceptual.
- **Community:** The *community* is anyone sharing the same aim. Members of a community can be individuals, stakeholder groups, role names, etc.
- **Rules:** Relations between the subject and community are mediated by *rules*. These rules may be implicit or explicit and they represent norms and conventions present in the activity.
- **Division of Labor:** Relations between the aim and community are mediated by the *division of labor* (DoL). The division of labor specifies how the task of achieving the aim is distributed across the community.

Mediation:

The general scheme of an ASD is two triangles, one placed inside the other. Structurally, the vertices of the outer triangle represent **mediat*ing*** elements (tool, rule, and DoL). The elements at the

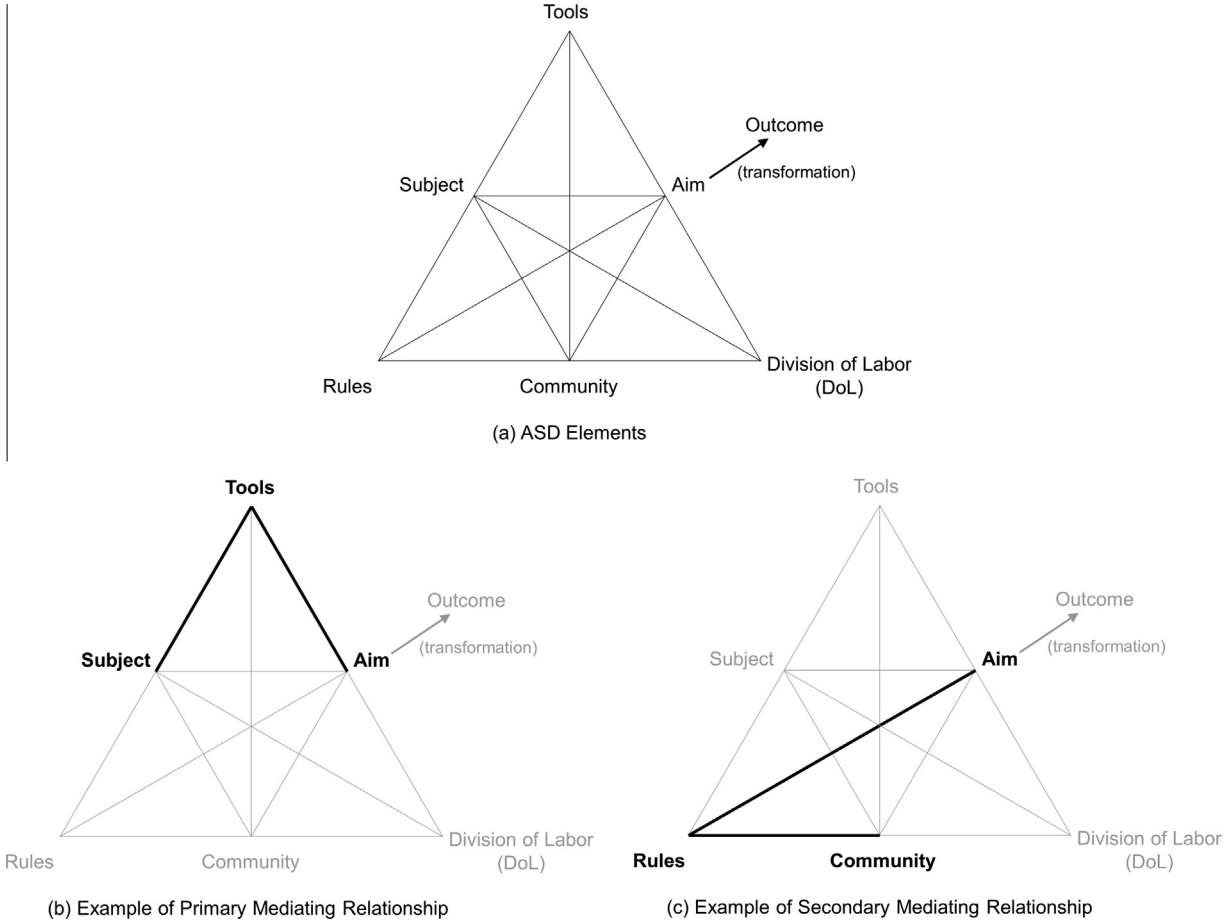

Fig. 1. Activity system diagram.

vertices of the inner triangle are the **mediated** elements (subject, aim, and community). Lines between any two mediated elements that pass through a mediating element describe what is called a *mediating relationship*. A mediating relationship causes some result, for example, the ability of the subject to achieve the aim of the activity by using a tool (see Fig. 1b), or the ability of the community to achieve the aim of the activity by dividing up the work according to the division of labor.

The primary mediating relationships are: a *tool* mediates the interaction between a *subject* and activity *aim* (that is, it helps the subject achieve the aim), *rules* mediate interactions between the *community* and *subject*, and the *division of labor* mediates interactions between the *community* and the *aim* of the activity. Besides the primary mediating relationships, the additional lines in the Engeström triangle indicate possible secondary mediating relationships. For example, a rule may mediate between the community and the activity aim (Fig. 1c), or between a subject and the activity aim. We have generally found that the primary mediating relationships are sufficient to describe an activity, and the examples in this paper only use these three types of mediating relationships.

Networking:

Beyond identifying ASD elements and mediating relationships, the AT framework acknowledges that human activity rarely takes place in isolation. It thus supports the concept of *activity system networks* where multiple ASDs are connected in the sense that an outcome of one ASD may be used as one of the other types of elements in one or more other ASDs. The most common use of this relationship is when the outcome of one activity is used as a tool in another. AT does not restrict networks to providing tools. For example, an outcome might be a rule applicable to another ASD or the training that would supply a subject for another activity.

Evolution:

AT also proposes that *contradictions* often exist at various levels within human activity, and it is these contradictions that lead to evolution of that activity. Evolution occurs to relieve contradictions within and between elements and previous versions of an activity. In particular, Engeström identified four types of contradictions: (1) within an AT element (e.g., between one rule and another in an activity), (2) between elements in an activity (e.g., between a DoL item and a rule of the same activity), (3) between multiple networked activities (e.g., between the outcome of one activity and another activity that uses that outcome as a tool), and (4) between multiple *evolutions* (e.g., between an ad-hoc activity and a more regulated one targeting the repeatability of that activity).

2.1.1. Social sciences ASD example

A more concrete example of using AT comes from the social sciences, where Hasu and Engeström use activity theory to analyze a technology transfer [20]. The particular case study is oriented around a complex piece of medical equipment that was developed by a research group and then transitioned into use in a hospital for research by staff doctors. The authors focused on the interaction dynamics between the developers of the technology and its users and employed AT to analyze these interactions.

The authors postulated contradictions based on the history of the technology in question, and then used observations to determine that these contradictions did lead to the actions reported. Contradictions involved the original activity, which used the medical technology and concepts in an academic, exploratory manner by a single group of researchers, and a second activity that used the results of this research (in the form of a machine) to perform additional research on patients in a clinical setting.

Across these two activities, contradictions existed in several places including the tools and rules of the activities. For the group who developed the machine, one key tool was the theory behind the machine, and a second was the system log that was used to trace, understand, and fix problems with the machine. For the hospital research doctors, the machine itself was the primary tool, with a telephone as another tool to ask questions of the developers when problems occurred. This is an example of a network relation between two activities: the outcome of the developers' activity was the machine that was used as a tool in the hospital activity.

A contradiction was that the rules for the developers were those associated with academic research and they did not include understanding the rules explicit and implicit in using the machine in a hospital setting. By contrast, the hospital staff was subject to procedures and rules associated with, for instance, patient safety, which affected how they had to use the machine. This in turn affected the DoL associated with using the machine: the actual machine had to be in a room separate from its controls, and the patient was located in that room. The doctor had to be in the control room, and a nurse had to be with the patient at all times. The doctor was unable to actually see the physical machine, patient, and nurse, and the nurse was unable to see any of the control settings on the machine.

Another contradiction lay in the hospital research doctor's activity aim (making a patient measurement) and the division of labor – the doctor was assumed to be responsible for the entire measurement, when in fact the nurse was needed to assist in performing it if certain problems occurred. The effect of this contradiction was that the aim of the activity could turn into troubleshooting the machine rather than making the patient measurement if these problems occurred. One of the situations investigated by the authors occurred when the doctor in the control room was alerted to this problem, but the nurse in the other room was the only person able to see the physical machine and hence what had caused the problem. Both participants therefore had access to only part of the necessary information, so neither was able to actively help the other in the troubleshooting activity. Ultimately, the measurement effort was terminated without successful completion.

The authors analyzed transcripts of actions and dialogs between the developers, the doctor and the nurse associated with this particular problem, and found evidence that the contradictions they identified could indeed lead to the observed actions of all the participants. Hasu and Engeström provide a networked ASD system in [20], which is partially extended, and shown below.

The two networked activities are shown in Fig. 2, with that of creating the machine on the left, and that of using the machine to take a patient measurement on the right. Hasu and Engeström explored the social interactions between the doctor and nurse during the aborted measurement session, and also between them and the academic research product developers who later visited for the purpose of updating some electronics in the machine. The authors point out collaboration issues within the group along with basic communication breakdowns that hampered addressing the measurement problem.

This example shows multiple types of contradictions as defined by Engeström. Contradictions across elements of a single ASD are found in the Hospital Activity between the Aim and the DoL, and between the DoL and the Rules. A single person operating the machine could not always make the measurement by themselves, so the Aim could not be achieved with the specified division of labor. Yet, even if the division of labor included the nurse, the nurse was not allowed to ever leave the patient and thus could not help the doctor. Contradictions between networked activities also exist, in that the Rules for the Machine Development Activity and those of the Hospital Activity contain contradictions regarding how measurements are taken (in an ad-hoc manner as part of the Machine Development Activity, and in a more regulated manner in the Hospital Activity). The shift in the Aim of the Hospital Activity from taking a patient measurement to troubleshooting the

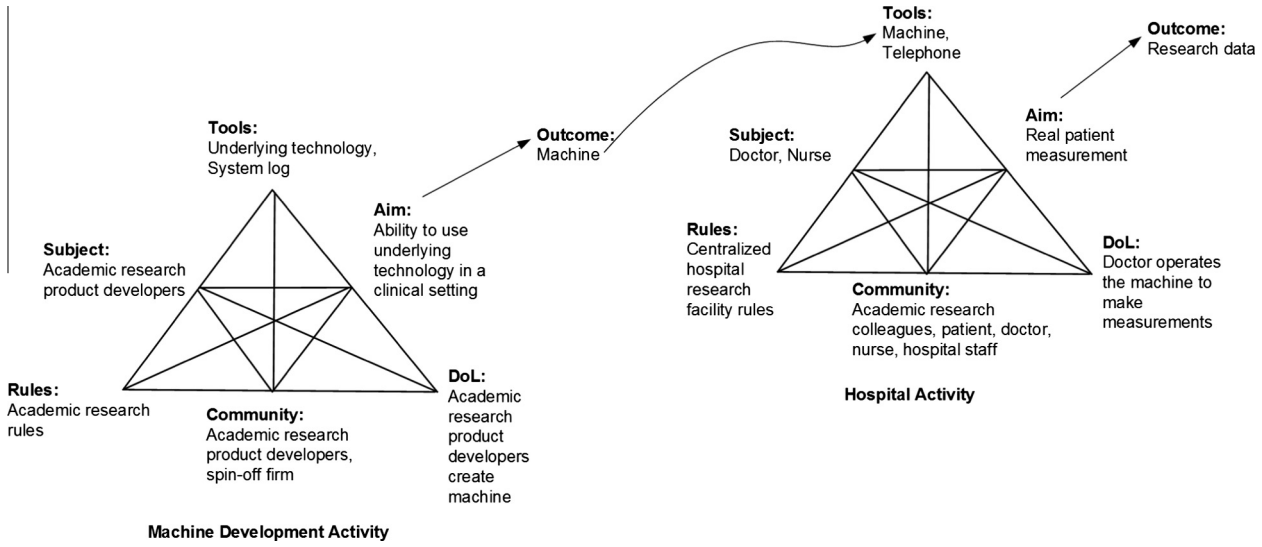

Fig. 2. ASD network described by Hasu and Engeström [20].

machine is an example of a conflict between evolutions (albeit short temporal evolutions) of that activity. The conflict arises primarily because the activity does not contain the Tools in terms of the System Log and the expertise needed to meet the evolved Aim of the activity.

It is currently difficult to analyze an ASD network systematically and even more so automatically because of a lack of formal representation. Furthermore, it is difficult for non-social scientists to come up with repeatable conclusions. The novel formalization presented in this paper is a prerequisite to any rigorous, repeatable analysis.

2.2. Rationale for choosing AT

The first two requirements for a social theory to use in our research are: (1) it must have had prior successful use in the social sciences in identifying a system's social constraints and explaining how addressing them or not affected the system, and (2) it should be used to some extent in the field of software development. The example in the previous section is just one of many where AT has been successfully used by social scientists to explore and explain the social interactions of human activities. It has been used in the fields of education and organizational learning, cooperative work, and technology transfers [6,15,35,39]. These and other works demonstrate that AT can be used in complex social situations to explore implicit as well as explicit relations and interactions among the participants of an activity. Research also describes AT applications in the field of human–computer interaction, and to some extent in software development [10,11,16,19,25,27,33,38]. The diversity of situations and participants described in this research indicates that AT is flexible enough to be used in diverse situations and that the concepts and terminology are conducive to shared understanding, as discussed by Halverson [18]. We therefore decided that AT does indeed meet our first two requirements of a social theory to be used in our research.

2.3. URN background

URN [2,23] is an ITU-T standard consisting of a goal-oriented language and a scenario-based language. The former is the Goal-oriented Requirement Language (GRL), used to model stakeholders and their intentions. The latter is the Use Case Maps (UCM) notation, which describes scenarios and high-level architectures and design.

GRL is a visual modeling notation for specifying intentions, business goals, and nonfunctional requirements (NFRs) of multiple stakeholders. It is also used to specify alternatives that have to be considered, decisions that have been made, and rationales for making decisions. A GRL goal model is a connected graph of *intentional elements* that optionally reside within *actors*. An *actor* (⬭) represents a stakeholder of a system, or the system itself. A goal model shows the NFRs and business goals of interest to the system and its stakeholders, as well as the alternatives for achieving these high-level elements. Actors are holders of intentions; they are the active entities in the system or its environment who want goals to be achieved, tasks to be performed, resources to be available, and softgoals to be satisfied. *Softgoals* (☁) differentiate themselves from *goals* (⬭) in that there is no clear, objective measure of satisfaction for a softgoal whereas a goal is quantifiable, often in a binary way. Softgoals are often more related to NFRs, whereas goals are more related to functional requirements. *Tasks* (⬭) represent solutions to (or *operationalizations* of) goals and softgoals. In order to be achieved or completed, softgoals, goals, and tasks may require *resources* (□) to be available.

A variety of links connect the elements in a goal model. AND, IOR, and XOR *decomposition links* (+—) allow an element to be decomposed into sub-elements. *Contribution links* (→) indicate desired impacts of one element on another element. A contribution link has a qualitative contribution type or a quantitative contribution. *Dependency links* (—●—) model relationships between actors.

GRL is based on *i** [53] and the NFR Framework [9], but is not as restrictive as *i**. Intentional elements and links can be more freely combined, the notion of agents is replaced with the more general notion of actors, i.e., stakeholders, and a task does not necessarily have to be an activity performed by an actor, but may also describe properties of a solution. GRL supports reasoning about goals and requirements, especially NFRs and quality attributes, since it shows the impact of often conflicting goals and various global alternative solutions proposed to achieve the goals. A GRL *strategy* describes a particular configuration of alternatives in the GRL model by assigning an initial qualitative or quantitative satisfaction value to a set of intentional elements in the model. GRL also takes into account that not all high-level goals and NFRs are equally important to a stakeholder. Therefore GRL supports the

definition of an *importance* attribute for intentional elements inside actors.

An *evaluation mechanism* propagates the initial satisfaction values to those of higher-level stakeholder goals and NFRs and eventually to satisfaction values at the actor level, taking the importance attributes into account. Strategies can therefore be compared with each other to help reach the most appropriate trade-offs among often conflicting goals of stakeholders. A good strategy offers rationale and documentation for decisions leading to requirements, thus providing better context for systems and software engineers, while avoiding unnecessary re-evaluations of worse alternative strategies. Currently, the URN standard does not prescribe a specific evaluation mechanism but provides non-normative examples of evaluation algorithms. Three evaluation algorithms applicable to GRL – a quantitative, a qualitative, and a hybrid approach – are described in more detail by Amyot et al. [1]. In general, evaluation results of a GRL strategy should not be considered out of context but always in relation to the results of other strategies.

The UCM visual scenario notation focuses on the causal flow of behavior. Behavior specification can be structured using components that abstract away message and data details. A *map* contains any number of paths and components. *Paths* express causal sequences and may contain several types of nodes. Paths start at *start points* (●) and end at *end points* (▮), which capture triggering and resulting conditions respectively. *Responsibilities* (×) describe required actions or steps to fulfill a scenario. *OR-forks* (possibly including guarding conditions) (⟍) and *OR-joins* (⟋) are used to show alternatives, while *AND-forks* (⊣) and *AND-joins* (⊢) depict concurrency. Loops can be modeled implicitly with OR-joins and OR-forks. The UCM notation does not impose any nesting constraints so joins and forks may be freely combined, and a fork does not need to be followed by a join. *Waiting places* (●) and *timers* (◷) denote locations on the path where the scenario stops until a condition is satisfied.

UCM models can be structured hierarchically with the help of *stubs* (◇) that contain sub-maps called *plug-in maps*. Plug-in maps are reusable units of behavior and structure. *Plug-in bindings* define the continuation of a path on a plug-in map by connecting in-paths and out-paths of a stub with start and end points of its plug-in maps, respectively. Plug-in bindings may also describe the relationship of components on the parent map with the ones on the plug-in map. A stub may be *static*, which means that it can have at most one plug-in map, whereas a *dynamic* stub may have many plug-in maps than can be selected according to a *selection policy*. *Components* (□) are used to specify the structural dimension of a system, enabling the specification of high-level architectures and design. Map elements that reside inside a component are said to be bound to it. Components may contain sub-components and have various types and characteristics. For example, a component of kind *actor* represents someone or something interacting with the system.

Given a specification of a *UCM scenario* including pre- and post-conditions, a *path traversal mechanism* can highlight the scenario path being simulated and verify whether the scenario unfolds as desired, essentially providing a regression testing environment for the UCM model.

The UCM notation shares many characteristics with UML activity diagrams but offers more flexibility in how sub-diagrams can be connected and sub-components can be represented. The UCM notation also integrates a simple data model, performance annotations, and a simple action language used for analysis. While activity diagrams have better support for data flow modeling, object flows, and a better integration with the rest of UML, the UCM notation is better integrated with goal-oriented models. GRL strategies and UCM scenarios are integrated with each other, allowing (a) a scenario to be highlighted that belongs to an active strategy and (b) a scenario to influence the satisfaction values of GRL model elements.

The URN standard also includes several language constructs that allow the definition of profiles for domain specific languages. For example, *metadata* are name/value pairs with which any URN model element may be tagged, *URN links* may connect any pair of URN model elements, and constraints in the Object Constraint Language (OCL) may be defined for any URN model element. URN links are used to establish traceability between elements in goal models to elements in scenario models.

2.4. Rationale for choosing URN

We are interested in using AT to identify social requirements, and therefore any system development methodologies we use with it need to include social aspects. Methods incorporating user intent are a natural fit since social constraints are often embedded in a user's intent. Those methodologies allowing intent to be treated as a first order concept are a therefore a good match with AT, and those providing analysis capabilities for intents are especially worth exploring. URN has these capabilities and existing tools that allowed us to achieve the fifth requirement of a social sciences theory as described in the introduction: (5) methodologies must be identified that allow the adapted theory results to be used in conjunction with existing system development methodologies and tools. (We describe two such methods in Sections 4 and 6 of this paper.) URN is also an international standard with available well-defined syntax and semantics. The language and associated tools actively evolve, with additional capabilities such as aspect-oriented concepts and evaluation [36] and the integration of feature-oriented modeling [30]. URN is also the first and currently only standard to address explicitly, in a graphical way and in one unified language, goals and scenarios, and the relationships between them. Furthermore, URN tool implementations exist that allow analysis and reasoning about models. In particular, the jUCMNav tool, developed at the University of Ottawa, enables goal model trade-off and impact analysis [24]. Finally, URN and jUCMNav provide traceability links between concepts and instances of goal modeling and behavioral design models. This traceability enables direct impact analysis of goal changes on a design and design changes on specific goals. Researchers have further demonstrated the ability of jUCMNav models to be refined and transformed into low-level design models that may be transformed into running systems using code generators [28,37]. Thus, integrating the results of an AT-driven methodology with the goal modeling and lower-level design capabilities of the jUCMNav toolset allow us to use the adapted theory results with existing tools to develop systems.

3. Constraining AT to produce repeatable results through automation and mapping it to URN

AT is a psychological framework, with concepts that are described using natural language. When it is applied to analyze a particular situation, the ASD elements are also described using natural language. Thus, there is ambiguity possible in both interpretation of the concepts and in their application to any particular situation. This makes the framework flexible in that it can be applied and interpreted in different ways. However, it makes it difficult to undertake automation since tools require precise definitions and interpretations to produce repeatable results. We have therefore constrained the theory in order to meet two of the technical requirements of a social theory to use in our research as described in the introduction: (3) it must be adaptable for use in

software development by persons who are not social scientists, (4) the adapted theory must produce repeatable results that are supported by automation as much as possible.

In order to address requirement (3) we have developed a domain-specific language (DSL) for AT that can be used as a basis for automated tools. We view this language as a step toward a grounded theory [45] adaptation of AT for software requirements engineering. This is only a step at this time because we only have one major situation investigated so far (this paper's case study), whereas many are required to properly ground the theory and its formal representation in reality. As we apply AT to other domains, we expect that our constraints may need to be modified to fit these different situations, leading to modified theory regarding the language definition. In order to meet requirement (4), we have added constraints in the language and created prototypical tools that automate analysis of resulting deterministic language element relationships. The language design goals were twofold: (1) to utilize all the concepts of AT in analysis, specifically all ASD elements, mediating relationships, network relationships and potential contradictions in order to support these different aspects of the AT framework and thus be able to explore their potentially varying ability to identify social constraints in a system, and (2) to constrain the language enough that automation can be used to realize repeatable results. The result of these goals is that there are relatively few constraints in the language, and most of them enable automated structural analysis of ASD network models to identify potentially missing elements, or elements that may not belong in the current model.

An example of one of the language constraints is the following. AT does not impose any constraints on Subject or Aim elements. Thus, it is possible to specify a subject who achieves the aim of the activity without a mediating tool. This situation may indicate that there is a tool missing from the ASD. We have therefore added a language constraint to identify such inconsistency through structural automated analysis. Such constraints can help non-social scientists to achieve repeatable results since, in their absence, it is only the experience of the social scientist analyst that can identify missing or extraneous elements in the model.

Our language is presented in the next section, followed by our use of trace-links to formally map relevant AT concepts to URN concepts, specifically those in the GRL language.

3.1. Metamodel specification of an AT language

We have defined a metamodel for AT [17] that includes Object Constraint Language (OCL) constraints [40,41]. A portion of this metamodel is shown in Fig. 3 (please see Appendix A for the complete language metamodel and OCL constraints).

Fig. 3 shows the AT concepts as specializations of an abstract class *Element*. A *Mediation* class is associated with three ASD elements, specifically two mediated elements and one mediating element, specified by the mediationElements multiplicity of 3, and the OCL constraints associated with the relation. The constraints related to mediation are: (1) the elements comprising a Mediation must be of specific different types, (2) every Subject/Aim Mediation must be mediated by a Tool, (3) a Mediation, its mediated elements, and mediating element must be part of the same ASD, (4) every mediating element must be part of at least one Mediation, and (5) an ASD must have at least one Mediation. Constraint (2) is used to enforce one of the precepts of AT, namely that a subject uses a tool to achieve an aim and thus there is at least one mediation relationship in an ASD (shown by the multiplicity of 1..* for the mediation role of the relation between an ASD and Mediation). These constraints are also used to highlight potential over-/under-specification of an ASD. For example, per (4), if a mediating element is not part of a mediation relationship, then either it does not belong in the specified ASD or else there is information still missing in the ASD specification.

The metamodel OCL constraints define well-formed ASDs and decrease ambiguity where needed to support automated analysis. Constraints are presented in natural language alongside related portions of the metamodel, outlined in thick borders, with arrows pointing to the location in the metamodel where they apply. For example, Fig. 3 shows that a well-formed ASD must have at least one each of the seven elements shown in Fig. 1a (e.g., rule, subject, tool, etc.). This is shown by the relation multiplicity of 7..* on the *elements* role and the natural language constraint pointing to this relation (*at least 1-each Tool*, *Rule*, ...).

An ASD network is created when the outcome of one activity becomes an element of another activity. This relation is constrained so that the element of the second ASD must be an element other than an aim or outcome of that second activity. This is another example of a constraint added to aid non-social scientists in using AT.

In addition to these explicit constraints, two relations are specified to help a practitioner identify whether an ASD is over-/under-specified. These are *dols2rules*, which makes sure that every Rule in an ASD is associated with some DoL item, and *whoDoesDoL*, which ensures that there is some Community (member) associated with each DoL. If a Rule exists that is not related to some DoL, the Rule probably does not belong in the respective ASD or else a DoL is missing. If there is a DoL that is not associated with some Community member, there is possibly a missing Community member. As in the other constraints and relations, the modeler must finally decide whether or not a real problem exists.

There are three classes shown that are grayed out: *EvolutionData*, *ContradictionData*, and *Motivation*. *Motivation* is an AT concept that can be used to specify information regarding the overall purpose of the activity modeled in an ASD. It is not an element that is integral to the mediation relationships in an activity, and thus we consider it an optional part of AT, and rarely include it in our work. *EvolutionData* and *ContradictionData* by contrast are placeholders for future concepts that are quite integral to AT, namely, how an activity evolves over time as a result of contradictions and tensions within it and across activity networks that include it. Expanding our language to include mechanisms to specify these concepts and reason about them is part of our future work. We introduce them here with the knowledge that they are currently incomplete.

In addition to specifying an AT language, we have developed mappings to transform the content of ASD network models into the system development languages of URN. AT concepts map to the GRL language specifically, with additional GRL elements being created based on the element relations in the ASD models. These mappings are explained below.

3.2. Transformations between ASDs and URN models

The key elements of GRL that relate to AT are Goals, Softgoals, Tasks, Actors, and Resources. Recall that Actors carry out actions (usually with the aid of Resources) to achieve Goals or Softgoals. Actions are most often specified as Tasks.

3.2.1. Informal mappings

Informally, we map GRL Actor elements to AT Subject and Community elements; GRL Task elements to AT DoL elements; GRL Goal and Softgoal elements to AT Aim, Rule, and Outcome elements; and GRL Resource elements to AT Tool elements. In addition, we note that GRL Goals derived from AT Aims contribute positively to GRL Goals derived from AT Outcomes. Also, if a Rule or Tool impacts several Community members, then the associated GRL elements very likely express a dependency between the GRL Actors derived from the Community members related to the Rule

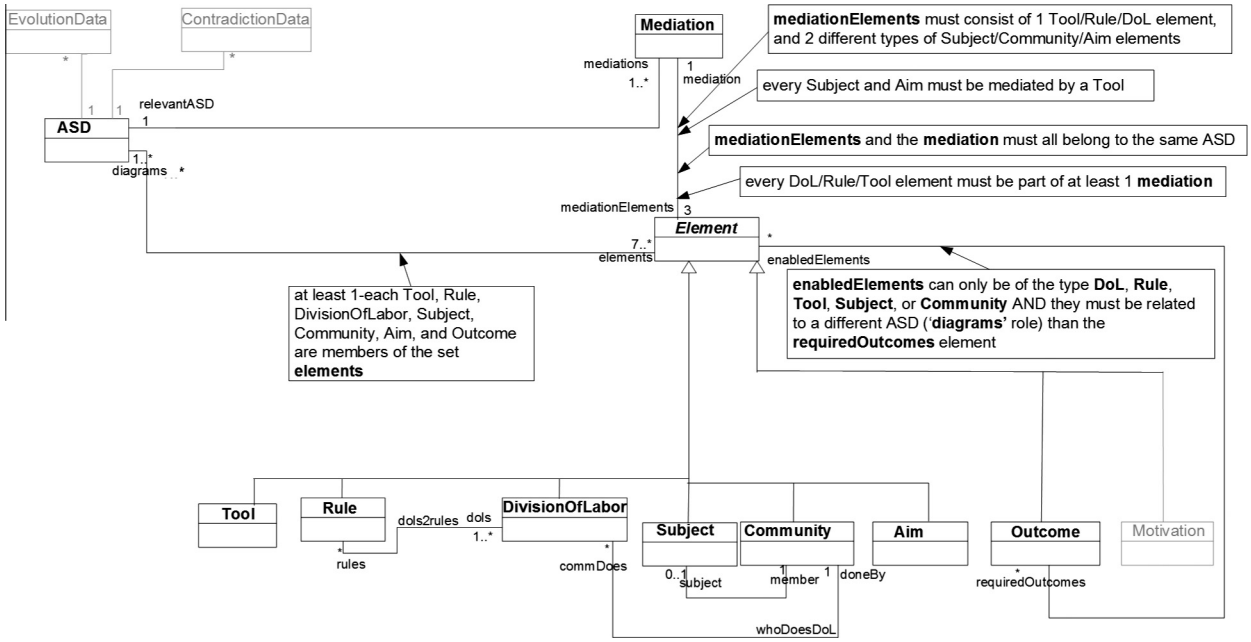

Fig. 3. Portion of AT metamodel with natural language constraints.

or Tool through mediation. To provide automated support for the transformation of AT models into URN models, we must formalize these mappings as described below.

3.2.2. Formal mappings

We have adopted the concept of *trace-links* [42] to specify the mappings and constraints between AT language elements and GRL elements. Trace-link semantics were proposed for modeling languages that differ significantly across a model-driven development process, and their specification includes (1) identifying the types of links that are needed between metamodels of two languages (shown in Fig. 4 for AT and GRL), and (2) constraints on the links that need to hold across models being mapped from one metamodel to the other (discussed in Appendix B). Such trace-links have the following advantages: they can be automatically generated by the model transformations taking place between the two related models (e.g., during model evolution); they do not affect at all the AT and GRL metamodels nor do they pollute any of the models with extra-information; and finally, they can be navigated in any direction during cross-model correctness analysis. This bi-directionality is particularly important since it allows ASD models to be created from system design and goal models, for use in, e.g., comparison with prior ASD models or stakeholder validation after system design evolution.

We have created trace-links for each of the informal mappings. Fig. 4 shows these trace-links with the relevant portions of the AT and GRL metamodels (shown as packages named *ATMetamodel* and *GRLMetamodel*, respectively), and the trace-link metamodel (also shown as a package and named *ATGRLMetamodel*).

Trace-links are specified between AT model elements (e.g., DoL) and their corresponding GRL model elements (e.g., IntentionalElement of type Task). The trace-links themselves are specified in the ATGRLMetamodel package. The main class in this package is *TraceModel*, which is a composition of *TraceLink*. *TraceLink* in turn is specialized. For example, a *CommunityActorTraceLink* instance is related to *Community* in the ATMetamodel package and *Actor* in the GRLMetamodel package. (We have used naming conventions for the trace-links to indicate the target types in the GRLMetamodel package since all targets except *Actor* are *IntentionalElement* of different types.)

According to the mappings informally discussed above, AT Community instances are trace-linked to GRL Actor instances, AT DoL instances to GRL Task instances, and AT Tool instances to GRL Resource instances. We constrain AT Outcome and Aim instances to be trace-linked to GRL Goal instances rather than Softgoal instances. This is because Goals have clear conditions for being met, and an activity must have clear conditions for being finished. This is the case even when an activity is on-going – there must be clear conditions for finishing an instance or iteration of the activity. AT Rules, however, may not be as clear and there may not be specific conditions that indicate they have been met. Thus, AT Rule instances may be trace-linked to either GRL Goal instances or GRL Softgoal instances (the modeler may provide direction in these cases). Note that there is no trace-link associated with the ATMetamodel package Subject class. This is because every Subject instance must be related to a Community instance per the *1* multiplicity constraint on the *member* role of the relation between Subject and Community (shown in Fig. 3). Since each AT Community instance is trace-linked to a GRL Actor instance, Subjects are thus represented in the GRL model.

Several correctness conditions can be formulated [26] based on the AT-GRL trace-links as detailed in Appendix B. This appendix also explains how additional GRL links can be established based on relations between respective trace-linked instances in an AT object model, as well as additional ASD relations.

Specifying an AT language and its transformation into a language that can be used in software development allows us to propose a methodology for incorporating the information in ASD network models into system designs using URN. A simple example is presented in the next section.

4. A requirements elicitation methodology using AT and URN

We demonstrate a methodology using AT and URN in this paper and explain it using an example and exploratory case study system. The methodology pertains to the elicitation and evolution phases of system development. In this section, Fig. 5 shows the activities executed during requirement elicitation, starting with AT models that are used to build URN models, and Fig. 9 in Section 6 shows the steps executed during model evolution, where

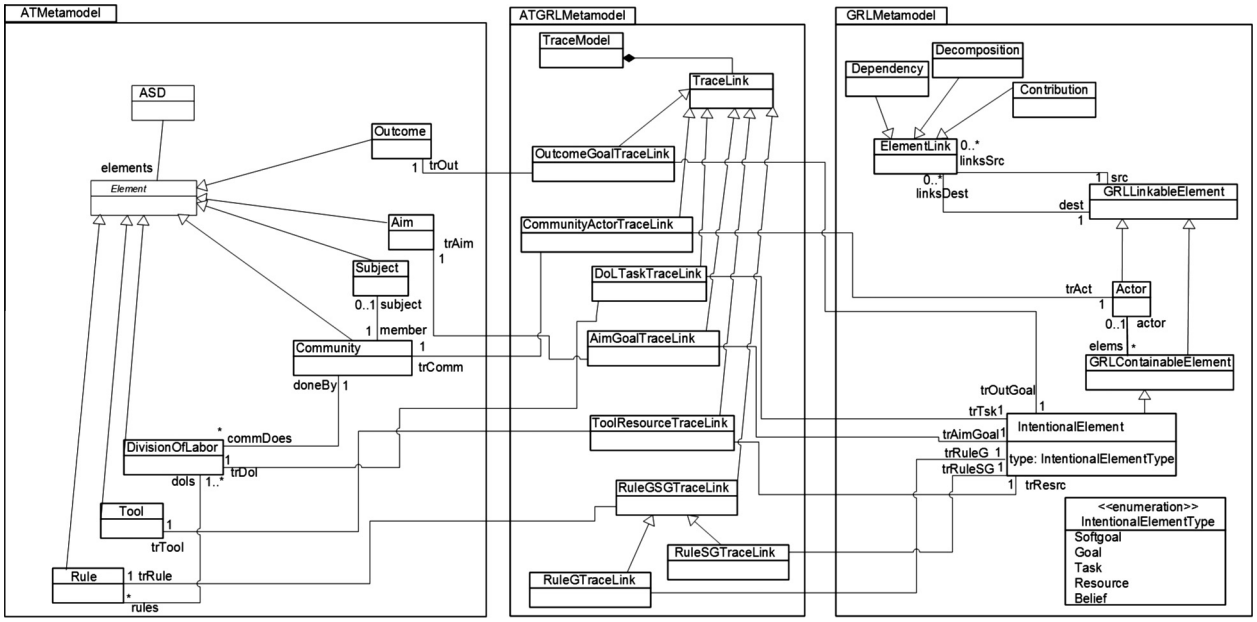

Fig. 4. Example trace-link specification from AT to GRL.

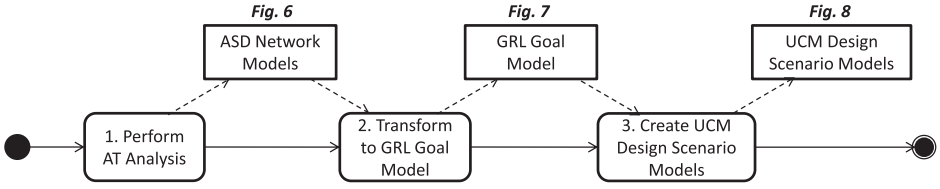

Fig. 5. AT/URN methodology, annotated with example artifact figures.

the analysis of the URN models guides the improvement of AT models. This methodology is not the only way AT and URN can be used in system development, however, it shows the potential complementary use of these languages.

The first part of our combined AT/URN methodology shown as a UML activity diagram in Fig. 5 consists of three activities (numbered rounded-rectangles) that produce and consume various models (represented by rectangles that are created or used according to the dotted line arrows). Control flow is indicated with solid arrows.

First, AT is used to create ASD network models of the example system (activity 1, the related artifact is shown in Fig. 6). This network is then transformed into a URN goal model using the trace-links mapping we have developed (activity 2, related Fig. 7). Once the goal model is created, additional analyses regarding positive and negative contributions to goals can be added. The next activity is to use the goal model to create high-level Use Case Map (UCM) design scenarios (activity 3, related Fig. 8). As system design advances, additional Tasks, relations, and perhaps Goals or Softgoals may be added to the goal model.

This methodology is demonstrated in Section 5 using a Vector Surveillance system.

5. Example mosquito vector monitoring system project

Examples in this paper are taken from the vector surveillance portion of our study system, a data capture system designed to help in the control of the Dengue virus and its associated diseases in Merida, Yucatán, Mexico. The Dengue virus is transmitted by a mosquito often found near human activity and is endemic to subtropical and tropical areas of the world. Since there is currently no Dengue vaccine with long-term efficacy, the most effective control method is to control the mosquito vector.

The study system resulted from a joint project between Colorado State University's (CSU) Microbiology, Immunology, and Pathology Department, one university (Universidad Autónoma de Yucatán), and the public health ministry (Servicios de Salud de Yucatán), both located in Merida, Mexico [22]. This project developed algorithms and a back-end database to support data entry and data presentation to decision makers in the public health ministry; it is referred to as the *Dengue management system* in this paper. A second, joint project that also included the Computer Science Department at CSU developed a cell phone application to be used in the field for mosquito surveillance data collection [13].

The most common practice for vector surveillance is to look for potential mosquito breeding sites and for the presence of larvae and pupae. Potential breeding sites often consist of uncovered containers of water in or near urban dwellings, either purposefully or accidentally collecting the relatively clean water preferred by the mosquito. The cell phone application allows Vector Control field agents from the public health ministry to enter data regarding the number and type of potential breeding sites into a cell phone from where it can be automatically uploaded to a central database. The application was developed and deployed for trials in Merida, and results were reported by the principal investigators [31]. We use this example system to demonstrate our AT-based approach primarily because the application is part of a complex socio-technical system with a diverse set of stakeholder groups that are collectively responsible for carrying out surveillance activities.

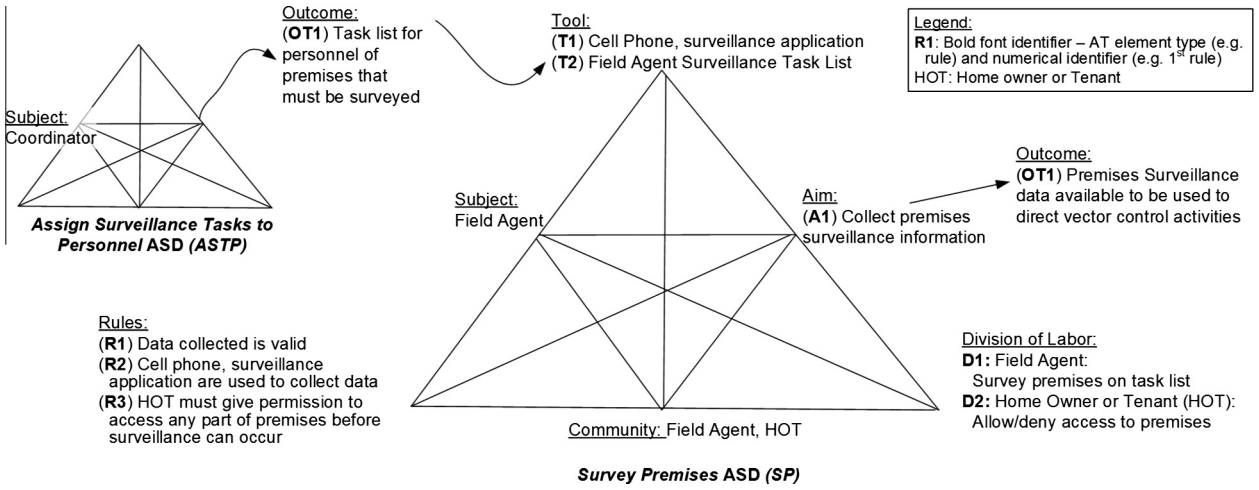

Fig. 6. Portion of premises surveillance activity network.

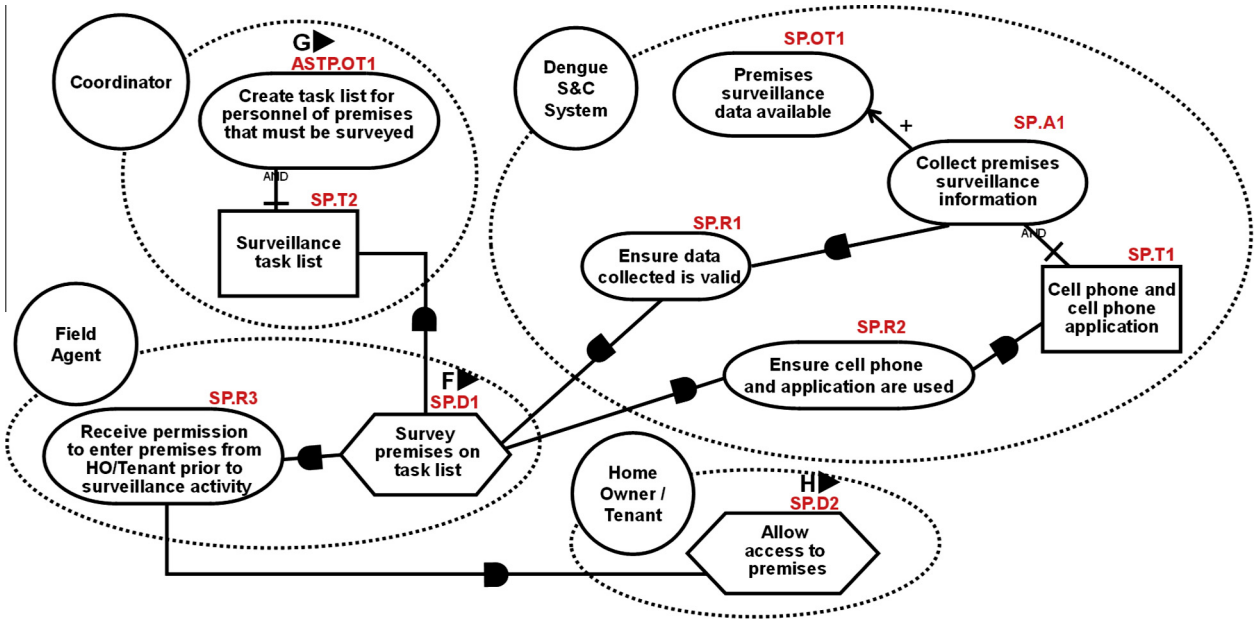

Fig. 7. URN goal model for premises surveillance.

There are many types of stakeholders who interact with the vector surveillance cell phone application. For example, field agents regularly perform surveillance activities in locations where confirmed disease cases have occurred. The persons living in the locations to be surveyed also interact with the system; they must grant permission for a field agent to enter the premises. Other members of the public health ministry must interact with the system. For example, coordinators must decide where field agents need to perform surveillance tasks, and policy makers must use the data gathered to determine if additional measures are required such as spreading larvacide or spraying for adult mosquitoes. Each type of stakeholder interacts with the system differently and has different goals and objectives. All stakeholders are also part of a larger community (e.g., the general public in the city of Merida) that affects and is affected by these interactions.

5.1. Modeling vector surveillance using AT

We used AT to describe vector surveillance in Merida as a set of ASD network models. This process corresponds to Activity 1 in Fig. 5. A portion of the resulting ASD network is shown in Fig. 6. The complete network consists of 4 different ASDs, each with a single aim and single outcome, and with a combined total of 7 community members (4 are subjects in the various ASDs), 9 tools, 15 DoL items, and 29 rules. Fig. 6 shows two ASDs in the network, represented by two ASD triangles. For simplicity, the ASD on the left (*Assign Surveillance Tasks to Personnel*, or *ASTP*) is only partially depicted. The diagram shows the name of the ASD, the Subject, and the Outcome. The main ASD of interest for our examples is *Survey Premises* (or *SP*), and all the ASD elements are shown for this diagram. *ASTP* is carried out by the Coordinator subject and has a single outcome, a list of premises that need to be surveyed by Field Agents. This list is used as a tool (T2) in the *SP* ASD by the Field Agent subject. The network relation between the two ASDs is shown by the arrow from the Outcome of *ASTP* to the Tool of *SP*. The *SP* Field Agent Subject also uses a cell phone and the surveillance application to achieve the Aim of the *SP* activity (*Collect premises surveillance information*). The activity is achieved through two DoL items – the Field Agent surveys the premises, and the Home Owner or Tenant (HOT) gives permission for the survey.

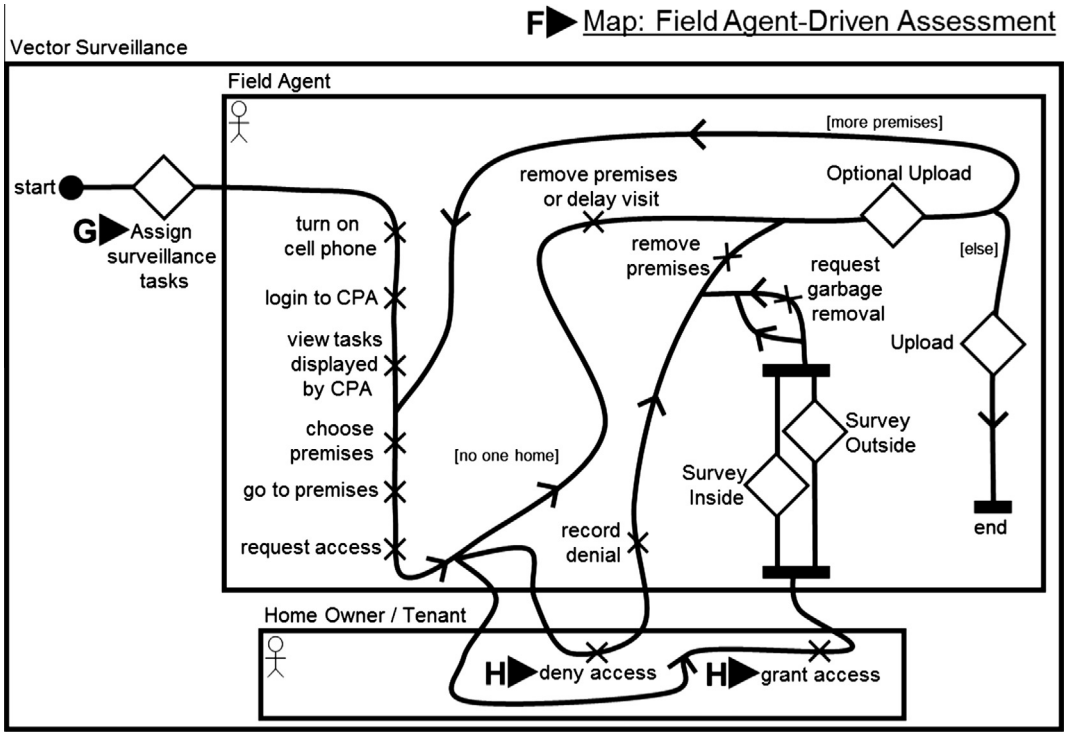

Fig. 8. URN scenario model for vector surveillance.

There are three Rules associated with the activity that are shown in Fig. 6: the data collected must be valid (R1), the cell phone and associated application must be used to collect the data (R2), and the HOT must give permission for the survey (R3).

Even though the ASD network shown in Fig. 6 focuses on the human activity of vector surveillance, it takes place in the larger context of the overall *Dengue management system* discussed at the beginning of this section. We have taken the position that many of the oversight and management functions related to the surveillance activity can be automated from the perspective of this overarching system. Therefore, while this overarching system is not directly visible in the surveillance ASD network, it does exist (most likely as a tool) in the complete hierarchy of activities that are part of dealing with Dengue in the city of Merida. We have added this overall system as a concept, specifically a URN stakeholder, in the goal and design models below. These models map Aims, Outcomes, and Rules that have impact or are of interest to the broader system to this stakeholder. The overall system is called the *Dengue S&C System* stakeholder in the discussion below.

5.2. Transform vector surveillance AT model to a URN goal model

We used the trace-links specified in Fig. 4 to transform the ASD network describing vector surveillance into a URN goal model. This corresponds to Activity 2 in Fig. 5. Items in the URN models below are annotated with the ASD from which they are derived and their ASD identifier. For example, SP.A1 refers to the only Aim of the SP ASD, *Collect premises surveillance information*. The resulting goal model is shown in Fig. 7.

The trace-links have been applied as follows.

Community-Actor mapping: *Actors* are created for each Community member in the *SP* and *ASTP* ASDs: *Coordinator*, *Field Agent*, and *HOT*.

DoL-Task mapping: IntentionalElements of type *Task* are created for each DoL, SP.D1 and SP.D2, and the *doneBy* role of the *whoDoesDoL* relation is used to create *actor*/*elems* relations (i.e., *containment* relations) for them. The SP.D1 Task is contained in the *Field Agent* Actor, and the SP.D2 Task is contained in the *HOT* Actor.

Aim and Outcome-Goal mapping: IntentionalElements of type *Goal* are created for SP.A1, SP.OT1, and ASTP.OT1. The *actor*/*elems* relations for these Goals require modeler direction for their creation. The ASTP.OT1 goal is contained in the *Coordinator* Actor, and the other two Goals are contained in the overall *Dengue S&C System* Actor.

Rule-Goal or Softgoal mapping: These mappings require modeler direction, as does their placement in an *actor*/*elems* relation. It is clear when SP.R2 and SP.R3 are met, so these can be mapped to IntentionalElements of type Goal. SP.R1 could be mapped either to a Goal or Softgoal. In this example we have chosen to map it to a Goal. The SP.R1 and SP.R2 Goals are contained in the *Dengue S&C System* Actor. The SP.R3 Goal is contained in the *Field Agent* Actor.

Tool-Resource mapping: IntentionalElements of type *Resource* are created for each Tool, SP.T1 and SP.T2, and modeler intervention is needed to specify their *actor*/*elems* relations. We have chosen to contain the Resource associated with SP.T1 in the *Dengue S&C System* Actor and the Resource associated with SP.T2 in the *Coordinator* Actor.

Additional dependency and decomposition relations in the goal model: We specify two decompositions from Goals to Resources, as well as additional dependencies between Tasks and Goals, and Resources and Goals. These additions generally need modeler direction to be completed.

In addition, several links have been added to the goal model presented in Fig. 7 according to the guidelines in Appendix B. The goal model shows several stakeholders as Actors (⬭, e.g., *Dengue S&C System* and *Coordinator*). Some stakeholders *depend* (—▶—) on each other. This dependency often takes place via Goals derived from the ASD Rules. For example, the *Dengue S&C System* depends on the Field Agent's Task (⬭), *Survey premises...* (SP.D1) to provide valid data (SP.R1) which results in its ability to achieve the *Collect premises...* Goal (⬭) (SP.A1). At the top of

the goal model in the *Dengue S&C system* Actor, the *Premises surveillance...* Goal (SP.OT1) corresponds to an Outcome in the *SP* ASD (i.e., *Premises Surveillance data...*, labeled OT1 in Fig. 6). This high-level Goal is further refined into a Goal corresponding to the Aim from the ASD (SP.A1) with the help of a *contribution* (→), i.e., the Goal trace-linked to the AT Aim contributes positively (+) to the Goal trace-linked to the AT Outcome.

Eventually, at the lower levels of the goal model, Tasks (e.g., SP.D1 and SP.D2) represent the DoL items from the *SP* ASD (e.g., the D1 and D2 items from Fig. 6, respectively). Tasks representing DoL items are contained in the Actor trace-linked to the AT Community member who is responsible for their completion. Thus, the SP.D1 Task is placed in the area of the goal model associated with the Field Agent Actor. Finally, Tools are represented by Resources (□, e.g., the *SP* ASD *Field Agent Surveillance Task List*, T2, is represented by the URN resource *Surveillance task list*, SP.T2), which is a decomposition (+—) of a Goal.

5.3. Transform URN goal model to UCM design scenarios

The goal model shown in Fig. 7 can be used to create UCM high-level system design scenarios for premises surveillance (Task SP.D1 in Fig. 7). This corresponds to Activity 3 in Fig. 5. The resulting system design scenario is shown in Fig. 8. URN links (▶) specify connections between elements of the goal model and use case maps or elements of use case maps. For example, the *allow access to premises* Task of the Home Owner/Tenant in the goal model is labeled H▶ in Fig. 7, and it relates to the *deny access* and *grant access* responsibilities (×) of the Home Owner/Tenant Component (□) in the UCM shown in Fig. 8. (Both of these responsibilities are labeled H▶ in Fig. 8.) The jUCMNav tool that we use to specify goal models and UCMs allows internal traceability between model elements such as these to be specified and maintained.

Fig. 8 shows the entire UCM inside a single UCM Component called *Vector Surveillance*. There are two participating Components, a *Field Agent* and a *Home Owner/Tenant* (HOT). Vector Surveillance begins with the assignment of surveillance tasks (G▶ related to the ASTP.OT1 Goal in the *Coordinator* Actor in Fig. 7). The Field Agent Component shows responsibilities for accessing the cell phone application to find surveillance tasks, choosing one, going to the premises, and requesting access. At this point the Home Owner/Tenant can either: deny access, grant access, or not be at home. In the latter case, the premises may be removed from the list or be left to visit at another time. If access is denied then this must be recorded and the premises removed from the list. If access is granted, then surveillance can be performed inside and outside the premises according to permission obtained. If there are enough existing or potential breeding sites found, the Field Agent may request that they be removed via a special garbage removal request. After the premises is surveyed, it is removed from the task list. Data can then be optionally uploaded from the cell phone to a centralized location, and the Field Agent can move on to other premises or stop the surveillance task. If the task is stopped, then data that has not yet been uploaded is uploaded from the cell phone.

The flow of surveillance is shown by the line from the *start* circle (●) to the *end* bar (▮). Directionality is shown using arrow heads along the line. Branches (—<) are often labeled with guards. For example, following the *request access* responsibility in the Field Agent component, there is one branch where the HOT denies access, another where the HOT grants access, and a third branch that is labeled with the guard *[no one home]*. The concepts of parallel *fork* and *join* are also present in UCMs and are shown as bars with flow lines coming out of them (—⊢, fork) or going into them (⊣—, join). For example, if the HOT grants access, there is a fork in the Field Agent component, where the Field Agent surveys the premises inside and outside. These two surveillance activities can be performed in any order, and then join prior to the next responsibility. Survey Inside and Survey Outside are not shown as simple responsibilities (×) but as hierarchical placeholders for more detailed actions that are defined in their own UCMs (◇).

The UCM shown in Fig. 8 implicitly uses functionality of the cell phone application (abbreviated CPA in the figure), even though the application component is not shown at this high level of abstraction.

6. Evolving the design

System designs are often modified as a result of changes in stakeholder requirements or changes in how and in what circumstances the system will be used. In these cases, AT models can be changed through stakeholder validation with respect to the new requirements. The activities shown in Fig. 5 can then be used to evolve the system design models. However, system designs are also modified as a result of the introduction of new technologies, and in these cases it can be difficult to decide if existing stakeholder requirements have been preserved. This problem can be compounded by trying to decide whether particular requirements are still relevant and need to be preserved in the evolving system. AT can also be used in this situation – by modeling design changes in UCM models, and propagating them back to Goal models and then to AT models, stakeholders can be engaged in validating additional or missing requirements in the evolved system.

Fig. 9 shows a method that consists of three main activities (numbered rounded rectangles) and two optional activities (with dotted lines borders) that produce and consume various models (represented by rectangles created or used according to the dotted line arrows). The optional activities are needed if the AT analysis discovers problems that cause changes to the ASD models. Control flow is indicated with solid arrows.

We demonstrate how the methodology shown in Fig. 9 can be used to employ AT during system evolution that is based on a change in system design.

6.1. Modifying the UCM design

An item of interest to the developing partners of the cell phone application is whether active neighborhood involvement in the surveillance process can increase the quantity of valid data that is obtained. This interest arose in part because of studies showing that community participation is critical to the success of vector surveillance and control programs that deal with diseases such as Dengue and Malaria [46,49,50]. The main issue with the Merida program is the need for a field agent to obtain permission from the HOT to enter the premises and perform a survey of potential mosquito breeding sites. Often the HOT is not at home when the field agent is surveying the area or denies permission to enter the premises [31].

A possible system evolution is to extend the system and cell phone application to allow individuals who are not field agents to survey their own premises. Note that the field agent-based approach (shown in Fig. 8) must still be supported by the system, and an entirely new scenario, as shown by the new Use Case Maps in Fig. 10, must also be supported. Note that while the only GRL Actors represented as UCM components in Fig. 8 are the Field Agent and HOT, the proposed change also requires *Coordinator* and a new *Entomologist* GRL Actors. Changing the UCM design scenarios corresponds to Activity 1 in Fig. 9.

The UCM scenario shown in Fig. 10 has three maps. (Links to the modified goal map shown in Fig. 11, e.g., O▶, are given in this UCM.) The left map (Fig. 10a) shows that a Coordinator must provide permission for a HOT survey. If this survey is the first for a

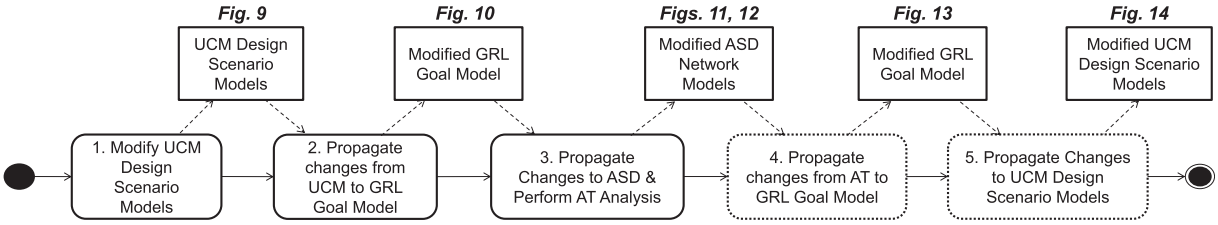

Fig. 9. Evolving a design and propagating changes to AT models, annotated with example artifact figures.

HOT, then the HOT needs to be trained to survey their premises (*Train HO/Tenant* map shown in Fig. 10c). Otherwise, the survey may go ahead (*Survey* map, shown in Fig. 10b). If, however, there are problems with the result of the survey, the HOT may have to go through another training session. Training must be scheduled by the Coordinator, and a Field Agent must be available to provide training. Training, however, is dependent on guides or other similar materials that the Field Agent can use to show the HOT how to perform surveillance and that can be left for use when conducting future surveys. An Entomologist must create these guides. We assume that a variation of the existing Field Agent vector surveillance cell phone application will be available to download and use in the self-survey situation.

During training (Fig. 10c), the field agent will provide guidance and check until satisfactory results are obtained, or if this is not possible, the HOT is blocked from performing self-surveys. Valid surveys include accuracy regarding the number of containers with water in them, the subset that has target larvae, and the subset that has target pupae. After surveillance, the HOT can upload results. We assume that the application will then be deactivated or removed until it is needed again, and the field agent will report that the HOT has been successfully trained.

The Coordinator must give permission (Fig. 10a) for subsequent HOT surveys. If no hold has been placed, the HOT survey can proceed according to Fig. 10b. The cell phone application must be obtained or reactivated before the HOT can perform the survey and upload results. Several checks can occur to better assure data validity. Examples are checking against previous data or data provided for neighboring premises. In the case of discrepancies, the Coordinator will be alerted and the data marked as suspect. The coordinator can then override the checking result and allow the survey results to be taken into account, require retraining, or simply deny the HOT further self-surveillance capabilities.

6.2. Propagating changes from UCM to the goal model

Changes from the evolved UCM design scenarios must be propagated to the goal model. This is accomplished through the use of tracing capabilities in the jUCMNav tool and designer actions to augment the goal model as needed to, e.g., add an Entomologist Actor. The resulting goal model is shown in Fig. 11. This corresponds to Activity 2 in Fig. 9. Items shown with colored backgrounds have been added to the goal model as a result of the changes made to the use case maps shown in Fig. 10. In the next section we discuss how new elements are added to an ASD Network Model using the trace-links as presented in Section 3.2. Much of this work is automated. The work needing modeler input is described in the next section. For the purposes of discussion we have annotated the new GRL elements in Fig. 11 with the ASD to which they need to be added and an ASD item identifier. For example, the new *Perform self-survey* Task in the Home Owner/Tenant Actor is annotated to show it should be part of the SP ASD as a new DoL item, D4.

6.3. Propagating changes to create an evolved ASD network model

Changes are propagated to the ASD Network Model by resolving inconsistencies in the trace-links between it and the modified goal model. This corresponds to Activity 3 in Fig. 9. Note that trace-link inconsistencies may arise from the addition of new GRL elements, deletion of GRL elements, or changes to GRL elements and/or the relations between them. In all cases the construction constraints

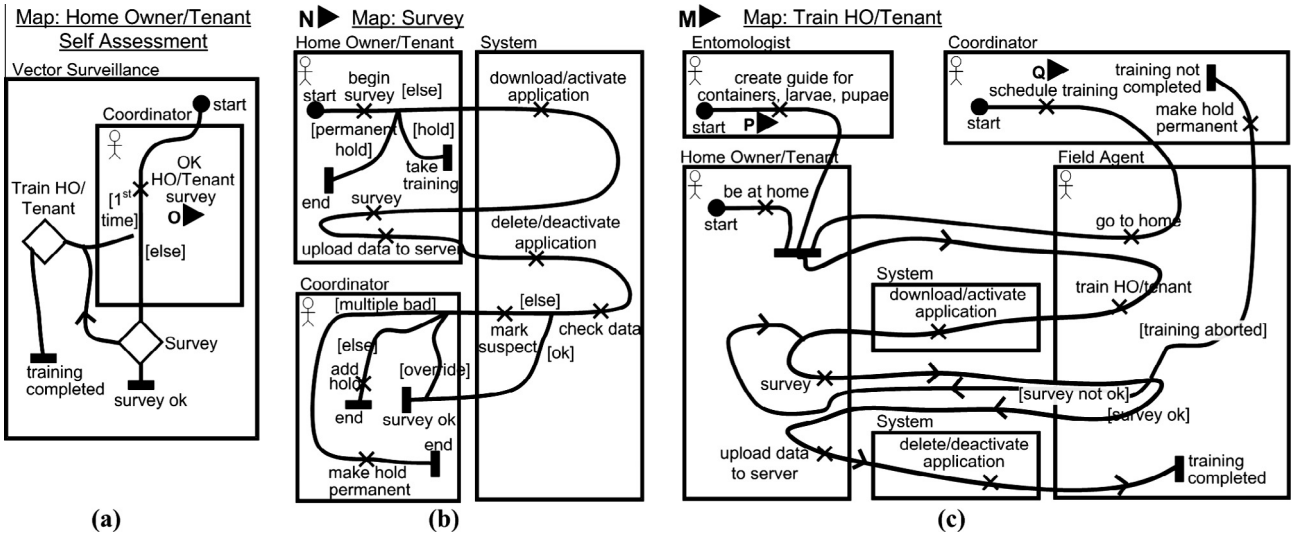

Fig. 10. UCM representing design change to allow home owner self-survey.

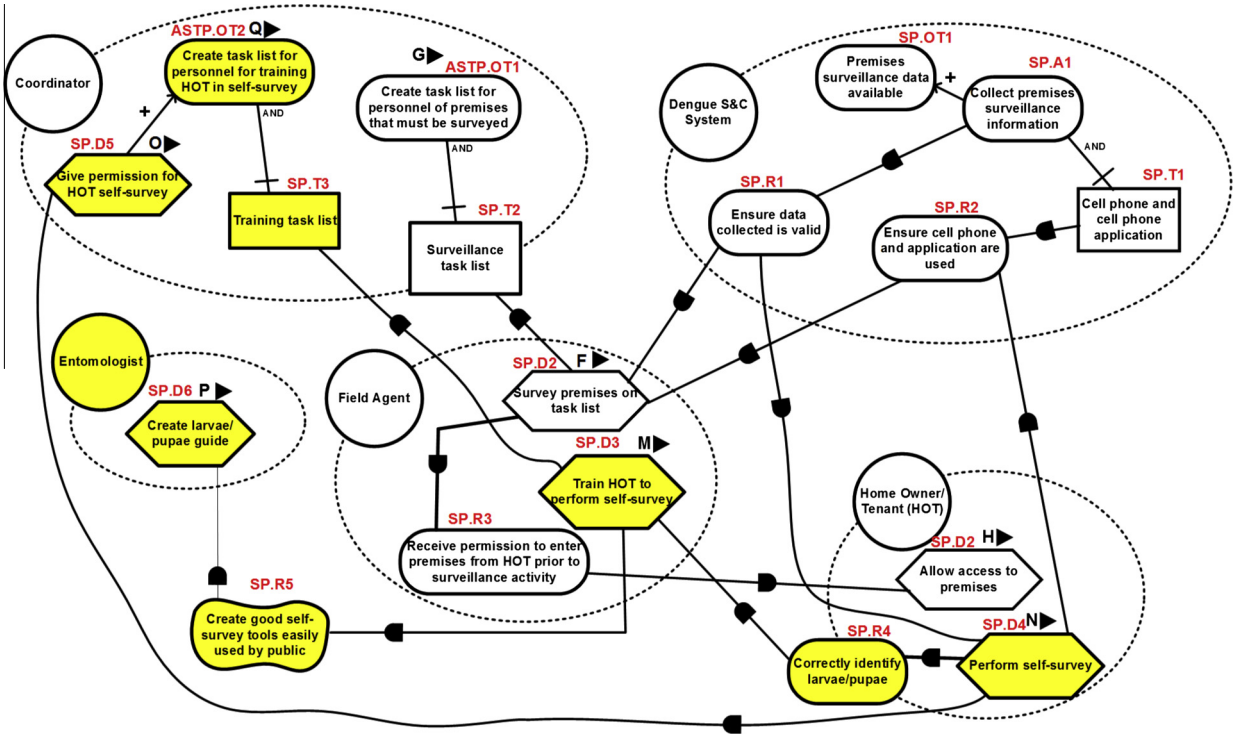

Fig. 11. Goal model relating to evolved system design that supports self-surveillance.

and additional relationship constraints in Section 3.2 and Appendix B are used to guide trace-link validation and direct changes.

If a new GRL element is added to the goal model, then a new trace-link of the appropriate type must be created. Resources, Softgoals, Tasks, and Actors are only associated with particular types of trace-links, but Goals can be associated with three different types of trace-links. We have developed heuristics to assist the modeler in making the correct association for these cases. An initial step is for the modeler to indicate the ASD to which new GRL elements should be associated. For example, in Fig. 11, the Goal *Create task list for personnel for training HOT in self-survey* is indicated via its annotation to be part of the *ASTP* ASD. In general, if the Goal has links to other GRL elements that are either trace-linked to AT elements in another ASD or that the modeler has indicated should be trace-linked to AT elements in a different ASD, then the Goal should be trace-linked to an ASD Outcome element. Otherwise the Goal most likely should be trace-linked to an ASD Rule element. Once the trace-links are created and associated with new AT elements, additional relations in the ASD network model can be created. For example, the annotations provided by the modeler in Fig. 11 allow creation of *eleInASD* relations between the associated new AT elements and the ASD to which they belong.

If the new elements are part of a new ASD, then the hierarchical and network relations of the new ASD must also be identified so that it is properly inserted into the overall system of ASDs. If a GRL element has been removed or changed, then the associated trace-links need to be validated. In the case of deletion, the associated AT element must be deleted, and any relations between it and other AT elements must be removed. In the case of change, there are three possibilities. First, if a GRL link has changed, then this information can be used to change ASD relations. Second, if the name of the GRL element has changed, there should be no impact on trace-links since the element itself still exists along with its previous relations. Finally, if an element has been changed from one IntentionalElement type to another, the trace-link needs to be replaced with the appropriate type, and the associated AT element needs to be replaced with the appropriate AT type. There may also be effects on existing relations in the ASD model. (These relation creation and changes can be performed automatically. Please see Appendix B for details on the conditions necessary for automation.)

The result of this propagation for the example system is shown in Fig. 12 as the evolved ASD network. For simplicity in the example, only the existing ASDs are modified. Items in italics indicate changes propagated from changes in the goal model.

6.3.1. Analyzing the evolved ASD network model

Analyzing the new ASD network model is also part of Activity 3 in Fig. 9. We have realized the AT metamodel as a model in the USE [48] tool. This allowed us to create ASD instances through the USE tool object creation mechanisms. These object models were then checked against the structural and OCL constraints of the language metamodel. We have used this implementation to perform structural and constraint analyses on the ASD network derived from the evolved design goal model (Fig. 11) that is shown in Fig. 12. This automated analysis identified inconsistences and missing items in the ASD network model.

One error was that D6 is not involved in any mediating relation, and a second error was that there is no *whoDoesDoL* relation between it and any Community member. The first error may be resolved by adding a mediating relation for D6 between it and the HOT Community member and the Aim; the justification being that the task of creating guides for the HOT to use during a self-survey mediates these two ASD elements. The second error is resolved by adding the Entomologist (who was added as an Actor to the goal model shown in Fig. 11) to the Community (shown in Fig. 13) and creating the *whoDoesDoL* relation between these elements.

Another error was that there is a Subject/Aim Mediation that is not mediated by some Tool. AT dictates that Subjects achieve activity Aims through mediating Tools, so this error indicates a missing Tool. This error resulted in the addition of a tool (T4 in Fig. 13) that mediates between the HOT subject and the aim. Finally, there was one item (SP.R5) in the goal model that most likely does not belong

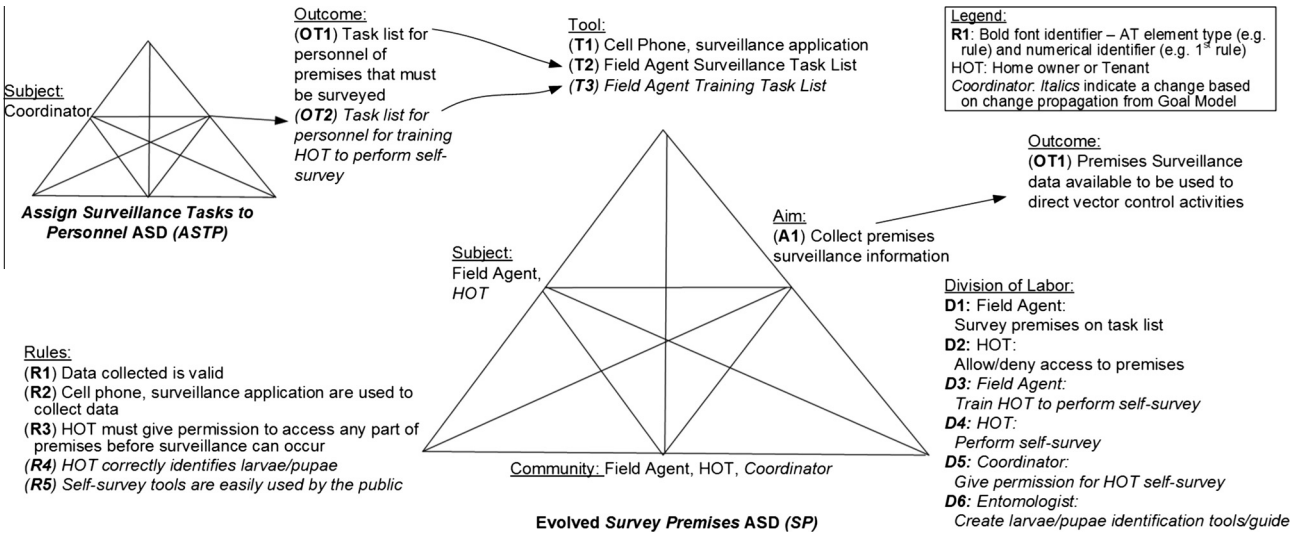

Fig. 12. Portion of evolved ASD network.

in the *SP* ASD. This problem was identified because there is no *dols2rules* relation between R5 and any DoL item. This Rule is also not involved in any mediating relation. As a result of this analysis, R5 was moved to a new ASD, called *Develop & Deploy Identification Tools* (*DDIT*). This change is shown in the ASD network in Fig. 13 as a strikethrough of R5 since the rule was deleted from the SP ASD, and in the goal model this change is propagated by changing the annotation on the trace-linked goal to *DDIT.R1*. The DDIT ASD is not discussed further in this paper. The missing items that were identified are shown in Fig. 13 as bolded.

In our roles as analysts, we studied the diagrams shown in Figs. 9–11, and it became apparent that there were potentially missing Rules regarding the HOT self-survey. Two examples are: (1) there is an order in the UCM shown in Fig. 10a, that permission must be obtained prior to performing and reporting the results of a self-survey, and (2) the HOT must use the Tools (i.e., T4 in Fig. 13) properly. It is significant that the USE analysis does not identify these two missing Rules (R6 and R7 in the *SP* ASD shown in Fig. 13). There are no general constraints that can be added to the metamodel that could find omissions such as these. Instead, they need to be identified through discussions with the (now augmented) stakeholder members of the community, or through study by the requirements engineer or analyst, and then be validated with the stakeholders. We have added these rules to the ASD shown in Fig. 13 to show how such changes will affect activities 4 and 5 of our methodology (Fig. 9).

6.4. Propagating ASD analysis results to the goal model

The ASD changes shown in Fig. 13 may be applied, via the trace-links, to the goal model of Fig. 11. The resulting goal model is shown in Fig. 14. This work occurs as part of Activity 4 in Fig. 9. Tasks, goals, and softgoals created as a result of the evolved UCM design are shown using a shaded background, as they are in Fig. 11. The additional Goals resulting from the addition of R6 and R7 in the SP ASD are shown with a patterned background and are annotated with SP.R6 and SP.R7 respectively. The additional Resource that results from T4 in the SP ASD is also annotated and shown with a patterned background. The Softgoal related to R5 in the SP ASD, which the USE tool analysis identified as probably belonging to some other ASD, has its annotation shown with a strikethrough. Finally, a new UCM link labeled T▶ is shown next to SP.R6 and relates to the change in the UCM design caused by this additional Goal.

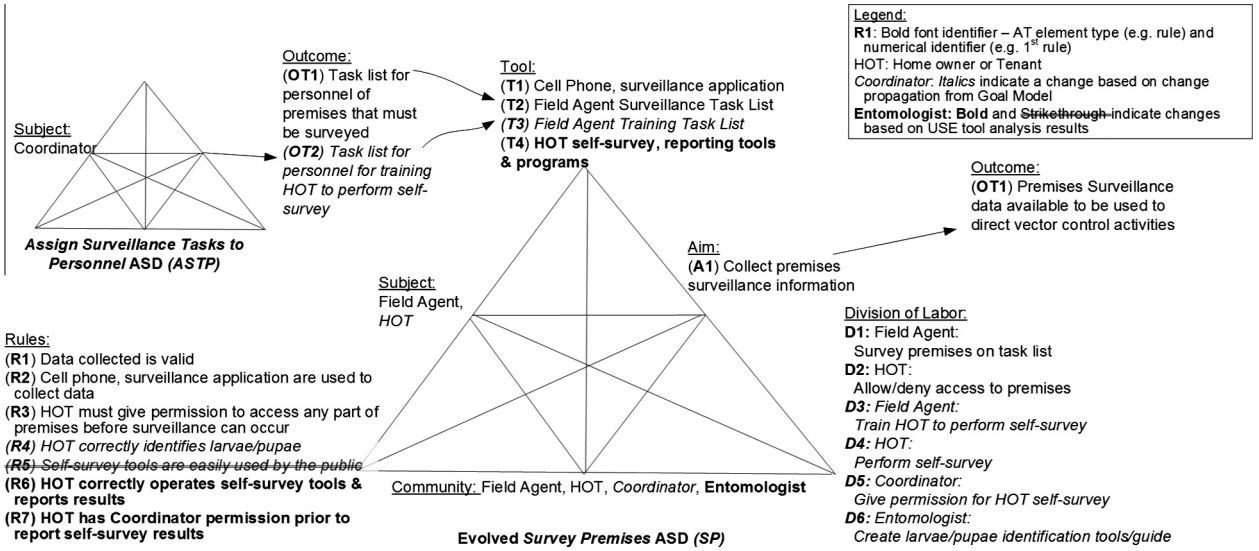

Fig. 13. Evolved ASD network modified with USE tool findings.

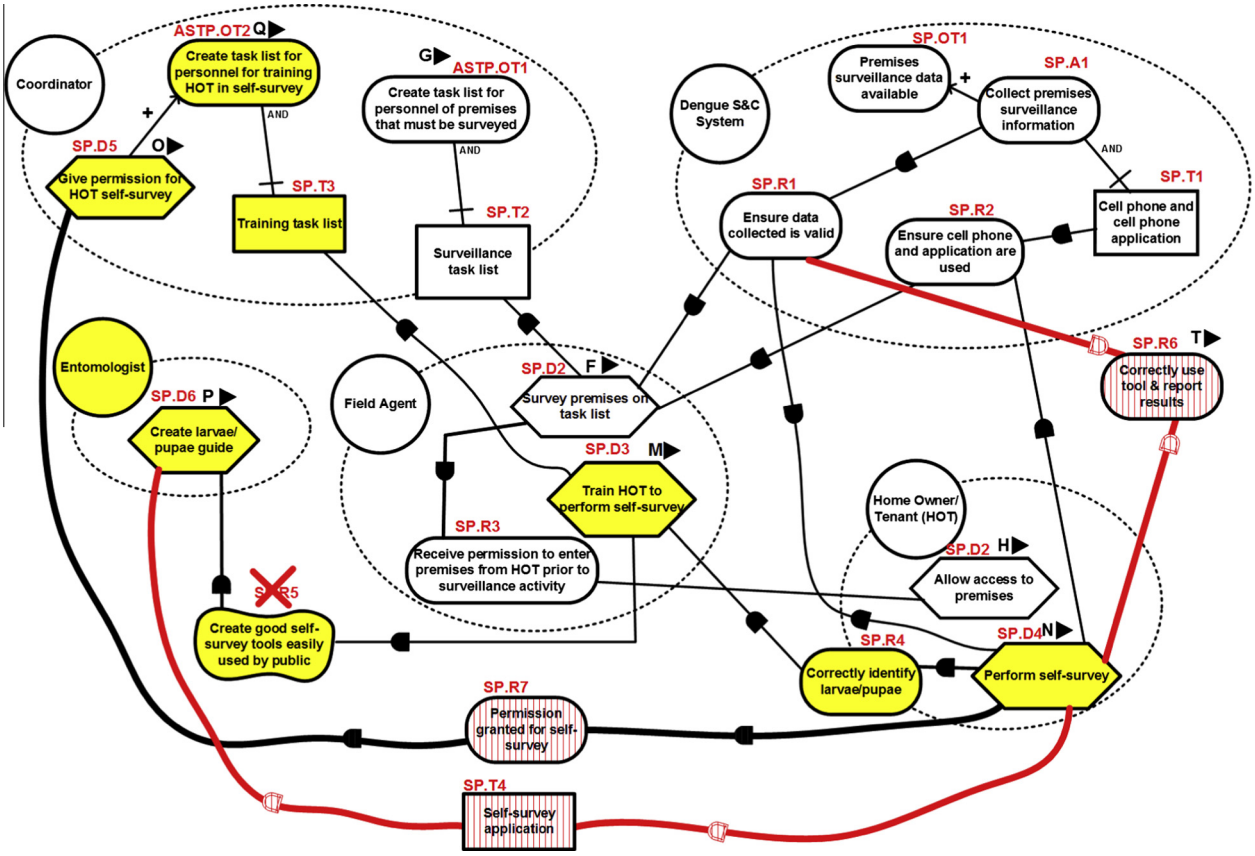

Fig. 14. Evolved goal model modified from changes to ASD network modified with USE tool findings.

6.5. Propagating changes to the UCM design scenario models

The changes caused by the modified goal model shown in Fig. 14 must be propagated to the evolved design scenarios (Fig. 10). This corresponds to Activity 5 in Fig. 9. First note that the Goal trace-linked to R7 is already represented in the UCM of Fig. 10b. Likewise the Resource trace-linked to T4 is already represented in the UCM of Fig. 10c. However, the Goal trace-linked to R6 needs to be adequately represented in the UCM design scenarios, particularly in the HOT training scenario, originally shown in Fig. 10c. The result is shown in the UCM of Fig. 15. These changes correspond to Activity 5 in Fig. 9. The changes to this map are shown with a shaded background.

The main difference in Fig. 15 is an additional responsibility in the System Component to check proper use of tools and proper reporting by the HOT. Once the Field Agent is satisfied that the HOT can perform surveillance properly, the HOT must initiate an initial report to the server, the System Component checks this, and if the tool and reporting has been performed properly, the data can be uploaded to the server and the training completed. If the HOT does not successfully use the tool and report data properly, this may be repeated until either the Field Agent aborts the training or the HOT is successful.

7. Discussion

This section relates the experience of the cell phone application development team and our own experience with using the integrated AT/URN approach, including threats to validity we have identified.

7.1. Merida vector surveillance cell phone application development – issues beyond fundamental functionality

The mosquito vector surveillance cell phone application was developed using requirements expressed as UML Use Cases. These use cases were created from the combined experience of domain experts, the system architect (Dr. Lozano-Fuentes), and designers, several of whom had experience with the vector surveillance activity in Merida, Mexico. There were multiple design reviews within the development team and reviews with other stakeholder groups including field agents from the Public Health Ministry. After the application was developed, field testing was conducted, which culminated in an experiment that contrasted the accuracy of data collected using the cell phone application and data collected using the previous paper-based reporting forms.

During testing and experiments, the researchers noted that less than 35% of the potential survey dwellings were actually visited either because no one was at home or permission to survey the premises was denied [31]. This relatively low response indicates that there are at least temporal and possibly societal issues that must be addressed before substantial data can be collected using either paper-based or automated systems. Because this is an urban setting, both the quality and the quantity of the data gathered are important when making vector control decisions. However, neither temporal nor societal constraints were present in the UML use cases.

It was during the experiments that field agents suggested the ability of a home owner/tenant to survey their own living areas might prove beneficial toward addressing both the temporal and societal issues regarding privacy voiced by some home owners/

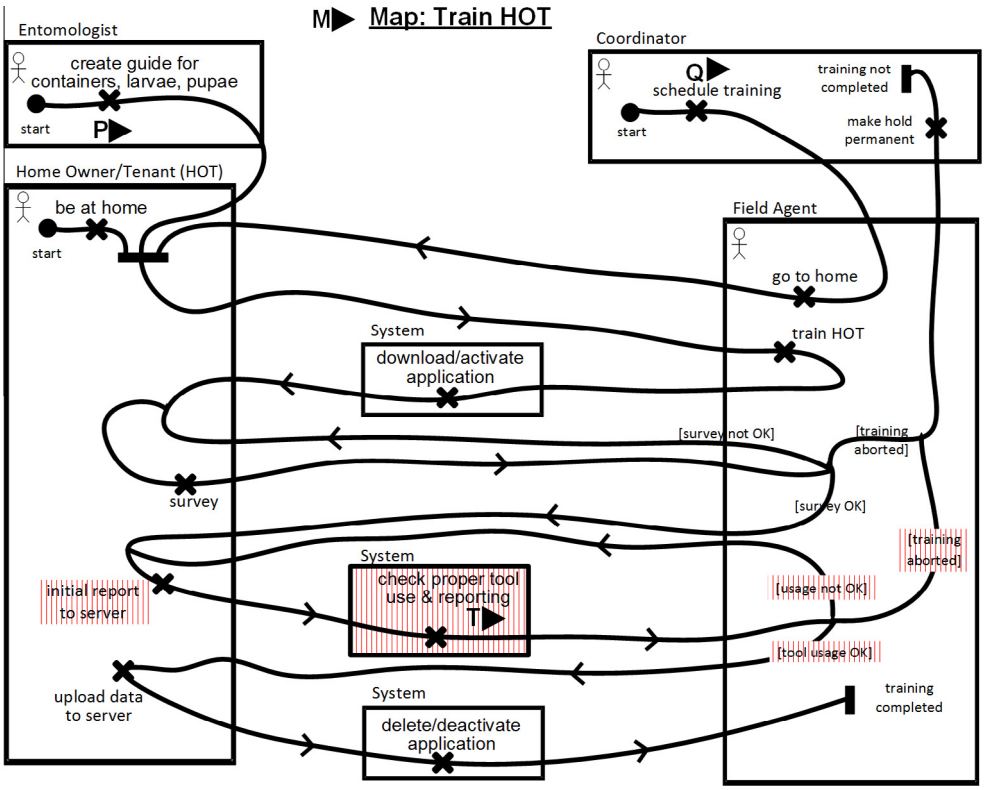

Fig. 15. Evolved train home owner/tenant UCM modified from goal model shown in Fig. 14.

tenants. This extension to the original cell phone application was the system evolution we decided to pursue with the AT/URN approach presented in this paper.

7.2. Opportunistic addition of the AT/URN methodology to the existing development project

The results from an opportunistic application of the constrained AT framework to the vector surveillance project mainly relate to our ability as mostly non-social scientists (except for Dr. Troup) to use it on a real development project. This is related to the third requirement of a social theory to be used in our research (as stated in the introduction), that it can be adapted for use in software development by persons who are not social scientists. Prior to completion of the cell phone project, we used UML Use Cases dealing with the premises surveillance activity to create initial ASD network models. An obvious but interesting issue is that many of the elements of an ASD are not present in a UML Use Case model. The cell phone project system architect (Dr. Lozano-Fuentes) helped develop the ASD models further and provided additional information based on working with the vector surveillance system in Merida. We were able to take the models thus created and put them into our USE tool prototype language, and then performed the automated structural analyses described in Section 6. Dr. Lozano-Fuentes subsequently used the modified ASD network models during some of the design and end-user reviews conducted as part of the cell phone project, with results described in the Lessons Learned, below.

7.3. Lessons learned

7.3.1. Identifying additional stakeholders

As we were developing the initial ASD network models from UML Use Cases, we experienced the ability of AT to aid in identifying additional stakeholders/members of the community firsthand. We were discussing the premises surveillance ASD from the point of view of the field agent, when the casual comment was made that when the number of potential breeding sites is above a certain threshold, the field agent requests a clean-up from the municipal garbage collectors. Thus, we discovered an additional Rule (what happens when the potential breeding site threshold number is reached), DoL (perform a clean-up), and community member (municipal garbage collection agency). In the case where AT is being used for the entire project, this new Community member would need to be interviewed regarding how they interact with the community at large and the field agents in setting up and performing a clean-up. We expect that new implicit and explicit social constraints would be discovered from each perspective.

7.3.2. Aiding stakeholder discussion

Another anecdotal contribution of AT occurred when Dr. Lozano-Fuentes used ASD network models during review discussions involving different stakeholder groups, including developers and field agents. He found that AT imposed a framework for discussion that was "easily" understood and used across these different stakeholder groups. He based this conclusion on the fact that in prior reviews conducted without the AT models, the stakeholders tended to focus discussion only on activities that required their direct involvement, while they considered other activities to be black boxes that were largely ignored. Dr. Lozano-Fuentes believes that the ASD network models encouraged different stakeholder groups to visualize additional activities in the system, beyond those for which they were personally responsible, and the models proved useful to create a template for open discussion between stakeholder groups. Stakeholders also expressed a higher degree of awareness and appreciation of activities undertaken by other stakeholders once the ASD network models had been used in reviews.

7.3.3. Identifying additional functionality and new stakeholders

An issue that was raised in a review after the vector surveillance cell phone application had been deployed in field testing for several months was that the participants identified a missing concept. A *premises* is currently the basic element for surveillance in the application and is assumed to be a single living unit. However, in the urban environment, a premises can actually be formed by several units (e.g., condos), and though owners/tenants can allow access to their own units, they are not responsible for or able to give permission to inspect common areas. The concept of premises must therefore be expanded to include areas inside a compound, such as patios and laundry areas, and the premises' superintendent is an important member of the community. Since the design of the application did not include the ability to identify more than the inside or outside of a premises at a particular address and unit number, the consequences of a change to add common areas that span such multiple localities are unknown. We believe that it is likely that if the AT framework had been available to obtain initial requirements for this system designers may have identified this broader definition of a premises prior to system design, specifically by identifying and involving the premises' superintendent as a stakeholder.

7.3.4. Identifying implicit rules

Another omission that was pointed out by field agents was lack of an explicit rule requiring the use of the cell phone application for surveillance tasks. Agents thought they could use the cell phone application or the previously used paper forms at their discretion. In this case, the rule was implicit to developers but not to the final system users. Adding this rule to an ASD network model makes this knowledge explicit and would have eliminated a point of confusion on the part of field agents.

7.4. Contradiction as the driver of evolution in the AT framework

AT includes as a basic tenet that tension caused by contradictions drives the evolution of an activity. For example, the rule that permission must be given before a field agent can perform surveillance is shown in Fig. 6. There is a contradiction between the rule and the aim of the activity, which is to collect surveillance information. If the home owner/tenant does not give permission or is not at home, then no surveillance data can be collected for the premises. The tension caused by this contradiction can lead to the evolution of the activity, for example the addition of the self-survey functionality that was proposed by field agents during field testing of the cell phone application (Section 6). We note that while no contradictions of any type are shown in an ASD, they are part of the AT language metamodel, and since they are related to ASDs they may be defined at any of the four levels described in the Evolution description of Section 2.1. Identifying, persisting, and visualizing such contradictions are some of our on-going areas of research.

7.5. Other benefits of integrating AT and URN

A contribution of URN to our integrated approach is the ability to specify design alternatives and compare them through trade-off analysis. As an example, consider the contradiction between requiring permission to survey and the need to collect data. This contradiction can be discovered from the ASDs, but there is no support to reason about alternatives that could be used to resolve this contradiction in the AT framework. However, advantages and disadvantages of possible alternatives may be described in URN goal models with further details of the alternatives shown in URN scenario models. These alternatives can be analyzed for their impact on stakeholders using existing URN analysis techniques [23]. Thus, a more informed decision regarding alternatives can be made with the URN model. The rationale for the decision, based on how well alternatives meet stakeholder goals, can be traced back to AT models and specific AT elements. This enables the entire rationale trace to be documented and used to test continued rationale validity as time passes.

URN can also be used to provide relative importance between rules that are included in an ASD. For example, if an analyst used the *SP* ASD to promote dialog among homeowners and field agents regarding the issue of permission to enter premises, there would be no way to relate the respective rules in their importance (e.g., "is collecting valid data" more important than "gaining permission"?). Yet this information needs to be captured because relative rule importance is required when tradeoff analysis is performed to determine how well design alternatives meet important goals. Using AT in conjunction with URN allows an analyst to use AT to elicit rules and transform these into URN goals and softgoals where the analyst can elicit and add the critical information of relative relevance required for further analysis.

URN can also show relationships, e.g., temporal or causal, among the GRL tasks created from DoL items. An ASD shows DoLs simply as an unstructured list. However, relationships do exist among these elements. For example, permission must be given by a homeowner/tenant *before* a field agent can survey a premises. When DoL items are transformed into URN tasks and then refined into scenario models, causal or ordering relationships are included. Similarly, there is an explicit activity order implied in any ASD network. In Fig. 6, a *Field Agent Surveillance Task List* is needed as a tool in order to complete the *SP* activity; therefore the *ASTP* activity must occur prior to the *SP* activity. However, activity order is not explicit, and if the activities in a network generate a cycle, then there is no way to identify an initial starting point from an ASD network. We do not transform an ASD directly into any URN element; however, their outcomes are transformed into goals. Thus, a networked relation can be described through the causal and ordering relations among the tasks that achieve these goals and the corresponding scenario models that refine the tasks in the GRL models.

Finally, the URN goal model makes explicit the dependencies among stakeholders that are only captured implicitly in the ASD model through rules and tools. The impact of a rule violation or absence of a tool can be analyzed systematically for each of the stakeholders using these goal models.

7.6. Additional AT/URN integrated approach research topics

Our experience using the integrated approach with the Vector Surveillance cell phone application has suggested several areas where additional research will be of benefit. The following three areas are of particular interest. The first area is automated support for contradiction analysis. Elements of an ASD are described using natural language, from the aim of an activity, to its rules, division of labor, etc. Automation, in terms of being able to more precisely specify these natural language elements and thus be able to reason about them, seems beyond the reach of our current tools. Thus, it is up to the modeler to determine where contractions may exist. We note, however, that once contradictions have been identified, URN goal model contribution links can be used to specify these contradictions and explore the effect alternative system design evolutions have in resolving them.

Second, we previously noted that implicit and explicit constraints are identified in ASD models. While not all of these are social, our hypothesis is that implicit Rules in particular tend to deal with social concerns. It is an on-going part of our research to develop heuristics that can partition constraints and identify those related to social concerns. In this case, natural language analysis seems to be an option worth pursuing. Such partitioning could

guide an analyst toward the social aspects of a system that may be riskier in the long term and more critical to its success than nonsocial aspects.

Finally, an interesting, remaining question from the example system is whether the inconsistencies and missing AT elements identified by the USE tool analysis would have been found solely through discussion with stakeholders using the evolved ASD network model (Fig. 12) as a discussion framing device or through an analyst studying this network model. While further experimentation is needed to determine the usefulness of such analyses, the fact that we have been able to rigorously specify the language and the relations among its elements does enable automated analyses that can help find issues in the models. Further, finding such inconsistencies prior to validation with stakeholders means that valuable stakeholder interaction time is not wasted on issues that could have been found automatically.

7.7. Threats to validity

In addition to the technical limitations discussed in earlier sections, there are several threats to the validity of the lessons learned and of the conclusions we have reached while attempting to answer our research question with the help of our case study. This section analyses validity threats based on the four aspects described by Runeson et al. [43] in their guidelines for case study research in software engineering.

7.7.1. Construct validity

This aspect of validity reflects to what extent the operational measures that are studied really represent what is investigated according to the research question [43]. One threat here is that we have a complex question operationalized with six requirements that are themselves rather complex. These requirements might collectively be too strict, leading to a sub-optimal solution or to no solution at all, or too relaxed, leading to a solution that may not work in some cases. Some alternative requirements were discussed and assessed in the introduction in order to mitigate this issue, but the risk of misalignment between the research question and selected requirements remains.

Another threat is concerned with the use of a case study (as opposed to a controlled experiment), whose topic and scope were imposed by an existing research project. Although we are benefiting from participation in a real-world project (hence avoiding some bias and providing a realistic motivation), given we did not have full control on the design of the case study, there is still a risk that it may not be entirely tailored for answering the question.

7.7.2. Internal validity

This aspect is concerned with causal relations between factors [43]. One threat is that we have not focused our attention on independent variables and on whether confounding factors exist that could affect the evidence used to support our assessment. Also, it is difficult to assess whether the feedback received from the participants resulted directly from our modeling and analysis effort. A better controlled experiment could provide some insight here and mitigate this issue.

7.7.3. External validity

This aspect of validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case study [43]. Indeed, one threat is the very specific scope of this unique case study, although it is representative of a larger collection of sociotechnical systems that involve local constraints and humans collecting data. This could be mitigated by having additional case studies from different contexts of interest to a larger audience. Additional case studies could also mitigate a related threat regarding the current formalization of AT, which may require enhancements to accommodate new situations (in a grounded theory way, meaning by discovering the necessary theory concepts by the analysis of grounded data).

In addition, we have selected particular languages for the social theory concerns (AT) and for system development (URN). Although each was justified at length in the paper, other related languages exist and it remains unclear which of the observations and lessons would still be applicable to these languages. In particular for URN, many other goal and scenario languages exist in the literature.

7.7.4. Reliability

This aspect is concerned with the extent to which the data and the analysis depend on the specific researchers involved [43]. The co-authors have complementary expertise in modeling, language design, software development, URN, psychology, public health, and vector surveillance. For much of this work, having these people enables triangulation at several levels, leading in turn to bias reduction. However, at the same time, some threats remain as various parts of this work were done by individuals. For example, the ease of understanding and of using the system (section 7.3.2) was assessed informally (through discussions and anecdotal evidence) and by only one of the co-authors (Lozano-Fuentes). Hence, this positive feedback may not transpose easily to other case studies executed by other people. Another obvious source of bias lies in the fact that the researchers are all co-authors of this paper.

8. Related work

Requirements engineering has long recognized the need to include system context (which may include social aspects) in the elicitation process. There are many proposed approaches that are related to our work ranging from stakeholder checklists to ethnographic and cognitive study techniques. Stakeholder checklists are aimed at helping analysts identify potential stakeholders for a system. Stakeholder identification using AT is more structured to the system being developed, in that a small set of stakeholders may be initially identified and through exploration of the holistic AT view of the activity the set may be expanded. While this method may seem less systematic than following a checklist, it takes advantage of the activity of interest, without effort on the part of the analyst to tailor or adapt a generic stakeholder checklist. Another benefit of the AT activity-directed approach is that time is not used to involve stakeholders who may not really be part of the activity community, but who were targeted through a checklist.

In comparison with the methods discussed below, AT is a broader approach that seeks to identify all community and societal influences on a particular human activity. It deals well with the often initial and potentially ongoing situation in a socio-technical system where there is no common goal among the widely divergent stakeholders participating in or affected by the activity. Another difference between the work presented in this paper and these approaches is that we have specified a language and transformations that allows us to take the information gathered through AT and insert it into existing system development methodologies and tools, specifically URN. The approaches briefly discussed below often focus on requirements gathering, and not on how the information they discover is specifically used in system design, development, and evolution. Some of the alternative approaches we have investigated include Distributed Cognition, social norming, Cognitive Work Analysis, Contextual Inquiry and Design, Complex Adaptive Systems Theory, and Emergent Knowledge Process design. Some of these approaches are more widely used than others, but they all focus on context elicitation from groups of stakeholders who ultimately share common goals.

Briefly, Distributed Cognition is an analytical framework that explores events driving information flow in a distributed problem solving system [18,21]. Social norming [12] considers social and organizational constraints as they introduce requirements conflicts across legal, organizational, and policy boundaries, especially in the domain of health information privacy. Cognitive Work Analysis [51] helps analysts explore the effect of unanticipated physical constraints on potential work tasks. The method distinguishes between desired work processes, and workarounds caused by the need to adapt to these physical constraints. Contextual Inquiry and Design [7] uses ethnographic methods to gather data from individual work participants who are focused on accomplishing a shared goal. Interactions and relations among people involved in the work, actual steps in the work, additional cultural norms in the work environment, and all pertinent physical information are captured in this methodology. Complex Adaptive Systems Theory [4] is a theory that can be used to explore behavior of small groups whose members interact and are interdependent. A key point of the theory is that while members are focused on a common purpose, this purpose evolves over time as the group interacts. Emergent Knowledge Process design [32] proposes a set of design principles that take different stakeholder user groups into account as part of the design process. Emergent Knowledge Process design recognizes that there are many unknowns regarding system usage, and hence system success depends to a large extent on how well access and application of emerging knowledge is supported in the system.

There is also a body of related work dealing with using AT in software engineering [5,10,27,33] and requirements engineering. Most often this work is in the context of categorizing requirements information into the elements of a single activity system diagram (e.g., tool, subject, DoL). AT has also been used to analyze human-computing interactions with the view of identifying obscured or ambiguous goals and designing interactions that allow these goals to be realized [25,54]. None of these examples explicitly constrains AT concepts for systematic, repeatable use as our research seeks to accomplish. We therefore have approached AT with the goal of formalizing it to the extent necessary to provide some automation in its use and a basis for automated transformations into URN so the resulting goals and intentions can be mechanically analyzed and incorporated into existing system development methodologies and tools.

Martins [33] developed a metamodel-based requirements elicitation methodology to guide the elicitation process and identify each of the elements in an activity system. Once all the elements are identified, an ASD is created. This approach can aid requirement completeness. Our approach differs in that we go beyond creating an ASD through the mapping to URN. We use the AT metamodel structure and OCL constraints to partially identify potential inconsistencies and incompleteness in the ASD. We can also identify similar issues as we map additional detail in the URN model using different abstraction levels provided by different ASD hierarchical levels.

Andreev et al. [3] build on the work by Martins and others to develop a requirements elicitation and specification framework for computer-interpreted guidelines for clinical practices in evidence-based patient management. Their framework assumes that all information related to user requirements exists in the form of what is called a semi-structured clinical practice guideline. This guideline is a flowchart or activity graph along with a narrative that explains individual steps. These are used to create a collaborative work diagram that includes aspects such as coordination and communication and means of work used across the team working from the clinical practice guideline. The actions of each team member are used to construct UML Use Cases, each of which represents a functional requirement of the computerized guideline, and each of which can be formulated as an Engeström triangle diagram. The authors state that nonfunctional requirements are identified through these diagrams. Our work differs in that it is targeted to be used in a more general-purpose domain, albeit one with socio-technical aspects. In addition, we use ASDs explicitly to identify the broader set of stakeholders and their implicit as well as explicit constraints, especially their social constraints. We make use of all of the ASD elements in our methodology, identifying them and using them in analysis, while Andreev et al. do not seem to use the Rules concept of AT. Similar to our combined methodology, they address networked activities, but unlike our use of AT, they do not include any hierarchical decomposition ability in their framework. Our use of AT is planned to make use of AT contradiction analysis, which they do not address. Finally, their framework does not consider design or trade-off analysis, which ours addresses through the combined use of AT and URN.

Neto et al. [38] describe how to integrate AT with the i^* methodology for RE. They use the concept of AT contradiction analysis to guide the i^* design process. Stakeholders and developers decide which tensions to use as the primary motivations for system development. The mapping to i^* does not directly use ASD tool, community, or rules elements, and the authors do not explore analysis after the i^* model is developed. Our method takes advantage of *all* information included in an ASD, and we use our mapping trace-links to propagate changes from goal models back to the ASD Network Models. In addition, the formal basis of URN allows us to add relations and relative importance to elements of the ASD (e.g., DoL and Rule items) and to analyze their effects on each other.

Fuentes-Fernández et al. [16] present a UML profile for AT that contains all the basic ASD elements. They develop patterns that are structural, requirements-level class diagrams and apply the profile and patterns (including conflict resolution patterns) in the design of multi-agent systems. Our approach varies considerably in that we begin with an ASD and use our domain-specific language and trace-links to transform it to a formal specification language where we can analyze relationships and properties. While we do not provide contradiction analysis per se, we are able to find possible inconsistencies and incompleteness of the original ASD networks through several means. First, by using hierarchical relations we are able to analyse the models at various levels of abstraction. Second, through mapping to URN, using its analysis capabilities to modify goal models, then propagating changes back to AT. Finally, the automated analysis provided by the USE tool helps us find structural and constraint issues in ASD network models. To our knowledge, outside of our work, there are no examples in the literature that use the entirety of AT to explore requirements completeness properties or to explore the relationships and consistencies of requirements across a network of activities or between multiple levels of refinements of an activity.

9. Conclusions

This paper proposes a combined approach for requirements engineering that takes advantage of the integration of two complementary and synergistic methods: Activity Theory and URN goal and scenario modeling. AT has been developed and successfully used in the field of psychology to analyze social aspects of human interaction and has become an important theoretical framework in HCI. AT provides a broad, conceptual framework for defining human activity as a system of multiple elements and their mediating relations. As explained in Section 4 and discussed in Section 7.3, AT helps identify the stakeholders in an activity, whether they are the subjects of the activity or members of the broader community, and makes explicit stakeholders' goals and actions, which are often implicit, obscure, or ambiguous.

In order to adapt the AT framework for use by non-social scientists and to help them achieve repeatable results through automation, we have made the AT framework precise by specifying an AT language metamodel and associated OCL constraints. We have implemented the resulting AT domain-specific language in the USE tool and used it to analyze ASD network models for consistency and structural constraints. We also mapped AT metamodel concepts to URN metamodel concepts with trace-links, further formalizing both the AT language and its relationship to URN. We use URN goal and scenario modeling to provide the formalisms necessary to analyze properties and perform trade-off analysis of alternative designs resulting from requirements identified with AT, thus allowing the AT results to be used with existing system development methodologies and tools.

We have demonstrated one possible methodology of an integrated AT/URN approach in the context of an example system that is part of a complex multi-user public health vector monitoring system. Our experience shows that the integration of AT and URN for requirements engineering is very promising. In general, AT provides a framework to explore social requirements and evolution while URN provides formality to analyze and exploit traceability. AT helps identify contradictions, resulting tensions, and possible evolution in human activity but an ASD does not capture contradictions explicitly. A URN goal model can make such contradictions explicit by first capturing rules as dependencies in the system and capturing any contradictions as contributions between the task causing the dependency and the rule. URN scenario models can be used to explore possible evolutions to examine whether tensions identified in the ASD network model can be resolved through system design evolution. The AT framework, along with a URN model, provides a systematic approach for a system design to be analyzed in terms of its ability to meet goals, and to alleviate contradictions that could impact system acceptance and use.

Our bidirectional trace-links allow automation, with limited modeler assistance in some situations, in moving between goal models and ASD network models. The jUCMNav tool similarly allows movement between goal and scenario models. Together, these capabilities provide analysts and designers with the ability to validate system-supported social constraints with stakeholders and to predict the impact of changes in social constraints on the system design. Particularly in complex socio-technical systems, the ability to identify social constraints and their relations to system design, and to be able to continually manage and reason about them, can increase the likelihood that both the initial system and its evolutions address the societal constraints critical to system acceptance and success, thereby helping preserve the resources invested in these systems.

Going back to the research question (“can Activity Theory be adapted, formalized, and combined with an existing system development methodology to support the repeatable discovery of non-obvious social requirements in socio-technical systems with local constraints?”), this paper provides evidence that supports five out of the six requirements defined in the introduction and used to refine this research question. AT is indeed a social science theory with prior successful use. It was adapted and formalized here for software development and then combined with URN to discover repeatedly (in a case study) non-obvious social constraint requirements, via a methodological approach supported by automated tools. However, the question remains only partially answered (albeit positively so far), with additional work required to satisfy the sixth requirement, which is concerned with whether these social constraint requirements might not have been found otherwise. Such work implies a more substantial comparison with related RE approaches claiming to detect social requirements in socio-technical systems. Lessons learned and threats to validity were discussed, leading to other improvement opportunities.

Our future work will include development of more elaborate tool support to integrate AT and URN tooling more tightly, going beyond the current implementation in USE. Important future work also involves validating the methodology by applying it to more case studies of complex multi-user interactive systems, thereby also providing opportunities to improve the completeness of our AT formalization. More elaborate tool support will also allow for independent case studies to be performed by other researchers, which in turn is a prerequisite for the mitigation of the reliability threats discussed in Section 7.7.4 and for improving external validity as discussed in Section 7.7.3. Such tool support is also needed to address the threats to construct and internal validity identified in Sections 7.7.1 and 7.7.2, respectively. Furthermore, we intend to provide automated support for contradiction analysis as described in AT. Thus, the contradictions between the Rule and the activity Aim identified in the example system in Section 7.1 could be identified systematically, and perhaps prior to system design, rather than having to wait for feedback from users during field testing. Another important area of future work is to provide tracing techniques that maintain relationships between tensions and their resulting system evolutions; to document design rationale and evaluate it for continued validity. Finally, we believe that tool support must be enhanced in order to test our sixth requirement of a social theory used in our research: discovering whether AT as we have adapted it for use in RE and system development can identify social constraints that are not identified using other RE methods. Automated support for experiments related to this requirement is critical so that complex systems can be explored.

Appendix A. OCL constraints on the activity theory language metamodel

The complete AT metamodel is shown in Fig. A-1. Constraints are presented in natural language alongside related portions of the metamodel, outlined in thick borders, with arrows pointing to the location in the metamodel where they apply. All of the natural language constraints shown in Fig. A-1 begin with the name of the relation, in italic font, to which they relate (e.g. *eleInASD: at least 1-each Tool*, *Rule*, ...).

Fig. A-1 shows the AT concepts as specializations of an abstract class *Element*. Two further abstract classes are defined as part of this specialization, *MediatedEle* and *MediatingEle*, along with the concrete *Outcome* class. The specialized abstract classes (*MediatedEle* and *MediatingEle*) are used to define the ternary relations associated with AT mediation. Specifically, *MediatedEle* specializations are the AT concepts that are mediated: Subject, Community, and Aim. *MediatingEle* specializations are the AT concepts that provide mediation: Tool, Rule, and DivisionOfLabor (DoL). A *Mediation* involves two mediated elements (specified by the multiplicity constraint *2* on the *scoEle* role of the *mediatedRel* relation) and one mediating element.

There are constraints related to mediation: (1) the elements comprising a Mediation must be of different types (*mediatedRel* relation constraint), (2) every Subject/Aim Mediation must be mediated by a Tool (*mediationRel* constraint), (3) a Mediation, its mediated elements, and any mediating elements must be part of the same ASD (multiplicity constraint *1* on *relevantASD* role of *medRelASD* relation and the *medRelASD* relation constraint shown in the figure), (4) every mediating element must be part of at least one Mediation (multiplicity constraint *1*..* on *mediates* role of *mediationRel* relation), and (5) an ASD must have at least one Mediation (multiplicity constraint *1*..* on *mediations* role of *medRelASD* relation). Constraints (2)–(5) are used to enforce one of the precepts of AT, namely that a subject uses a tool to achieve an aim and thus there is at least one mediation relationship in an ASD.

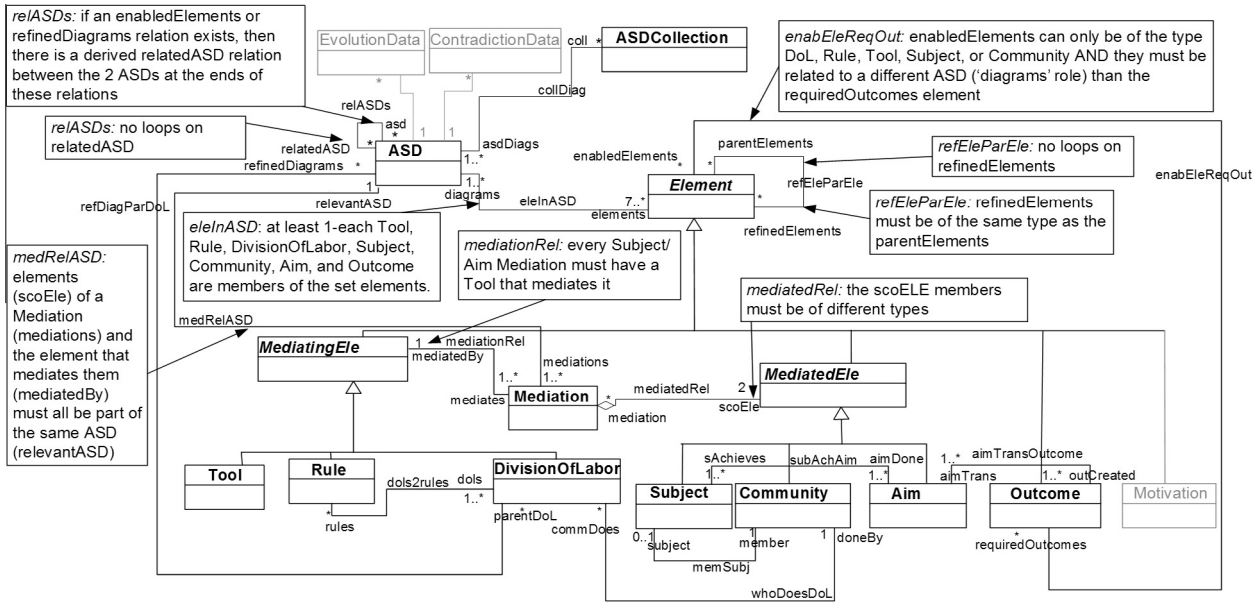

Fig. A-1. AT metamodel with natural language constraints.

These constraints are also used to highlight potential over-/under-specification of an ASD. For example, per (4), if a mediating element is not part of a mediation relationship, then either it does not belong in the specified ASD or else there is information still missing in the ASD specification. Clearly it is up to the modeler to decide if there is a problem or if the violation should be ignored.

Two refinement relations are defined in the AT metamodel. First, elements can be refined, with the constraints that the transitive closure of refinement cannot contain the original element (*refEleParEle* 'no loops' constraint), and refined elements are of the same specialized type as the original element (*refEleParEle* 'type' constraint). The other refinement relation is that a DoL item can be hierarchically decomposed into other ASDs (*refDiagParDoL* relation).

An ASD network (described in Section 2) is created when the outcome of one activity becomes an element of another activity (*enabEleReqOut* relation). This relation is constrained so that the element of the second ASD must be an element other than an aim or outcome of that second activity (*enabEleReqOut* constraint).

The metamodel in Fig. A-1 shows several relations among the AT framework concepts. For example, one relation specifies that a Subject is part of the community (*memSubj* relation), another that a Subject achieves an Aim (*subAchAim* relation), and another that an Aim is transformed into an Outcome (*aimTransOutcome* relation).

There are two grouping mechanisms defined in the metamodel. The first is the *relASDs* relation. This relation is derived: if two ASDs are related through hierarchical decomposition (*refDiagparDoL* relation) or they are networked (*enableEleReqOut* relation), then they have a *relASDs* relation. An additional constraint on this relation is that the transitive closure of the related ASDs cannot contain the initial ASD (*relASDs* 'no loops' constraint). The second grouping mechanism uses the *collDiag* relation to relate a set of ASDs into a collection (*ASDCollection*). The only constraint on an *ASDCollection* is that it must be related to at least one ASD (multiplicity constraint *1..** on *asdDiags* role of *collDiag* relation).

The natural language constraints on the AT metamodel shown in Fig. A-1 are given in the table below, along with their corresponding OCL predicates. Table A has three columns: the name of the relation to which the constraint applies along with the metamodel classes that are related, the natural language version of the constraint, and the OCL invariant.

Appendix B. Correctness conditions and additional links and relations

B.1. Correctness conditions

There are several correctness conditions related to the AT-GRL trace-links. These constraints are specified with the Epsilon Validation Language in a trace-link model [26]. For simplicity, we state the constraints in English: in all cases, exactly one trace-link must exist between one instance of a class in the ATMetamodel package and one instance of a class in the GRLMetamodel package. Table 1 (constraints i–vi) below shows the AT element, the trace-link type, and the GRL element when transforming an ASD into a goal model. For example, if a Community instance exists in an ASD, then the corresponding goal model must contain an Actor instance.

Table 2 (constraints vii–xi) shows the information when new GRL elements are added to a goal model, requiring new trace-links to new AT elements in the corresponding ASD. For example, if a new Softgoal instance is added to a goal model, then a corresponding Rule instance must exist in the ASD.

B.2. Additional GRL links and ASD relations

A modeler may choose to create additional GRL dependency links in a goal model between GRL elements. She may also add *actor/elems* relations between GRL Actors and other Goals, Softgoals, Tasks, and Resources that are trace-linked to ASD elements.

While a new GRL Resource element can be automatically related via trace-links to a new ASD Tool element, in the case where there are multiple ASD Subject elements, and/or multiple ASD Aim elements, the modeler must provide direction regarding which Subject and Aim elements should be included in a Mediation element for the Tool element (via the *mediationRel* relation in Fig. A-1). Additional mediation relations can also be created using direction from the modeler when new GRL model instances are trace-linked to ASD element instances.

We are also able to create GRL dependency and contribution links based on relations between respective trace-linked instances in an AT object model. The conditions for creating these GRL links are given in (a)–(d) below. In a similar manner, for new GRL elements that are not already trace-linked to elements in an ASD,

Table A
AT metamodel OCL constraints.

Relation, related classes	Constraint in natural language	Constraint in OCL
relASDs between ASD and ASD	*relASDs:* if an enabledElements or refinedDiagrams relation exists, then there is a derived relatedASD relation between the 2 ASDs at the ends of these relations	context ASD inv relatedIsRefinedEnabled: self.relatedASD → forAll (a:ASD | (self.elements → select (oclIsTypeOf(DivisionOfLabor)) → exists (e:Element | e.oclAsType (DivisionOfLabor).refinedDiagrams → includes (a))) or (self.elements → select (oclIsTypeOf(Outcome)) → exists (e:Element | e.oclAsType (Outcome).enabledElements.diagrams → includes (a)))))
relASDs between ASD and ASD	*relASDs:* no loops on relatedASD	context ASD inv noRelatedAsdLoops: self.relatedASD→ forAll(asd : ASD|self <> asd)
enabEleReqOut between Outcome and Element	*enabEleReqOut:* enabledElements can only be of the type DoL, Rule, Tool, Subject, or Community AND they must be related to a different ASD ('diagrams' role) than the requiredOutcomes element	context Outcome inv enabEleTypes: self.enabledElements→forAll(e : Element|e.oclIsTypeOf(DivisionOfLabor) or e.oclIsTypeOf(Rule) or e.oclIsTypeOf(Tool) or e.oclIsTypeOf(Subject) or e.oclIsTypeOf(Community)) and self.enabledElements→ forAll(e : Element|e.diagrams→intersection(self.diagrams)→isEmpty())
refEleParEle between Element and Element	*refEleParEle:* no loops on refinedElements	context Element inv noEleRefLoops: self.refinedElements→ forAll(e : Element|self <> e)
refEleParEle between Element and Element	*refEleParEle:* refinedElements must be of the same type as the parentElements	context Element inv refinedEleSameType: self.refinedElements→forAll(e : Element|self.oclIsTypeOf(Tool) implies e.oclIsTypeOf(Tool) and self.oclIsTypeOf(Rule) implies e.oclIsTypeOf(Rule) and self.oclIsTypeOf(DivisionOfLabor) implies e.oclIsTypeOf(DivisionOfLabor) and self.oclIsTypeOf(Subject) implies e.oclIsTypeOf(Subject) and self.oclIsTypeOf(Community) implies e.oclIsTypeOf(Community) and self.oclIsTypeOf(Aim) implies e.oclIsTypeOf(Aim) and self.oclIsTypeOf(Outcome) implies e.oclIsTypeOf(Outcome) and self.oclIsTypeOf(Motivation) implies e.oclIsTypeOf(Motivation))
eleInASD between ASD and Element	*eleInASD*: at least 1-each Tool, Rule, DoL, Subject, Community, Aim, and Outcome are members of the set elements.	context ASD inv asdHasEleTypes: self.elements→exists(e : Element|e.oclIsTypeOf(Tool)) and self.elements→exists(e : Element|e.oclIsTypeOf(Rule)) and self.elements→exists(e : Element|e.oclIsTypeOf(DivisionOfLabor)) and self.elements→exists(e : Element|e.oclIsTypeOf(Subject)) and self.elements→exists(e : Element|e.oclIsTypeOf(Community)) and self.elements→exists(e : Element|e.oclIsTypeOf(Aim)) and self.elements→exists(e : Element|e.oclIsTypeOf(Outcome))
mediationRel between MediatingEle and Mediation	*mediationRel:* every Subject/ Aim Mediation must have a Tool that mediates it	context Mediation inv STA: Mediation.allInstances() → forAll(mc:Mediation | mc.scoEle → forAll (e1,e2:MediatedEle | (e1<>e2 and e1.oclIsTypeOf(Subject) and e2.oclIsTypeOf(Aim)) implies (mc.mediatedBy→exists (t:MediatingEle | t.oclIsTypeOf(Tool)))))
medRelASD between ASD and Mediation	*medRelASD:* elements (scoEle) of a Mediation (mediations) and the elements that mediate them (mediatedBy) must all be part of the same ASD (relevantASD)	context ASD inv mediationInOneASD: ASD.allInstances() → forAll(a:ASD|a.mediations → forAll(mc:Mediation | mc.mediatedBy → forAll(e:MediatingEle | a.elements→includes(e))) and (mc.scoEle → forAll(f:MediatedEle | a.elements →includes (f)))))
mediatedRel between Mediation and MediatedEle	*mediatedRel:* the scoEle members must be of different types	context Mediation inv constituents: self.scoEle→forAll(e1, e2:MediatedEle | (e1 <> e2) implies ((e1.oclIsTypeOf(Subject) and not (e2.oclIsTypeOf(Subject))) or (e1.oclIsTypeOf(Community) and not (e2.oclIsTypeOf(Community))) or (e1.oclIsTypeOf(Aim) and not (e2.oclIsTypeOf(Aim)))))

Table 1
Trace-link constraints (i)–(vi).

Constraint	ATMetamodel package: Element class instance	ATGRLMetamodel package: TraceLink class instance	GRLMetamodel package: GRLLinkableElement class instance
(i)	Community	CommunityActorTraceLink	Actor
(ii)	DoL	DoLTaskTraceLink	Task
(iii)	Aim	AimGoalTraceLink	Goal
(iv)	Outcome	OutcomeGoalTraceLink	Goal
(v)	Rule	RuleGSGTraceLink; [a]one of either RuleGTraceLink or RuleSGTraceLink	[b]One of either Goal or Softgoal
(vi)	Tool	ToolResourceTraceLink	Resource

[a] Modeler assistance is generally needed to complete this decision.
[b] Depending on the decision made for the trace-link, the corresponding GRL element is created.

Table 2
Trace-link specifications (vii)–(xi).

Constraint	GRLMetamodel package: GRLLinkableElement class instance	ATGRLMetamodel package: TraceLink class instance	ATMetamodel package: Element class instance
(vii)	Goal	[b]One of either OutcomeGoalTraceLink, AimGoalTraceLink, or RuleGSGTraceLink	[a]One of either Outcome, Aim, or Rule
(viii)	Softgoal	RuleGSGTraceLink	Rule
(ix)	Actor	CommunityActorTraceLink	Community
(x)	Task	DoLTaskTraceLink	DoL
(xi)	Resource	ToolResourceTraceLink	Tool

[a] Modeler assistance is generally needed to complete this decision.
[b] Depending on the decision made for the AT element, the corresponding trace-link type is created.

we are able to create AT relations based on GRL links between respective trace-linked instances in a GRL goal model. The conditions for creating these AT relations are given in (e)–(i). The ASD in Fig. B-1 is used to demonstrate these additional link creations, which are shown in an excerpt of the goal model corresponding to it, as shown in Fig. B-2.

In this example, the *aim* of the activity is to add some (unspecified) new visualization features to an interactive software development environment (IDE). The *subject* who will perform this activity is an IDE Software Developer. The *outcome* of the activity is a new version of the IDE with the added visualizations. The *community* consists of personnel who collectively develop the IDE (member name prefixed with *IDE*) as well as the end users of this software (member name prefixed with *End User*), who are software developers themselves. Specifically, the community members are the IDE Requirements Engineer, IDE Software Developer, IDE Tester, IDE Integration Engineer, and End User Software Developer. The *division of labor* tasks of the activity are divided among the five members of the community as follows: (1) An IDE Requirements Engineer has the task of defining the requirements for the new visualization functionality. (2) The IDE Software Developer must design and code the new functionality in addition to creating developer tests. (3) An IDE Tester also develops tests and runs subsystem tests that include tests of the new functionality. (4) An IDE Integration Engineer is responsible for running system-wide regression tests. (5) The End User Software Developer member of the community must provide input on requirements and feedback on visualization as it is being developed (optimally) and as they use it in their daily jobs as developers. The *tools* used by the IDE Software Developer include detailed functionality requirements and company-wide guides regarding various aspects of software development. Finally, four example *rules* are shown that mediate behavior of the subject and community members: (1) The IDE Developer must attend design and project meetings. Behavior

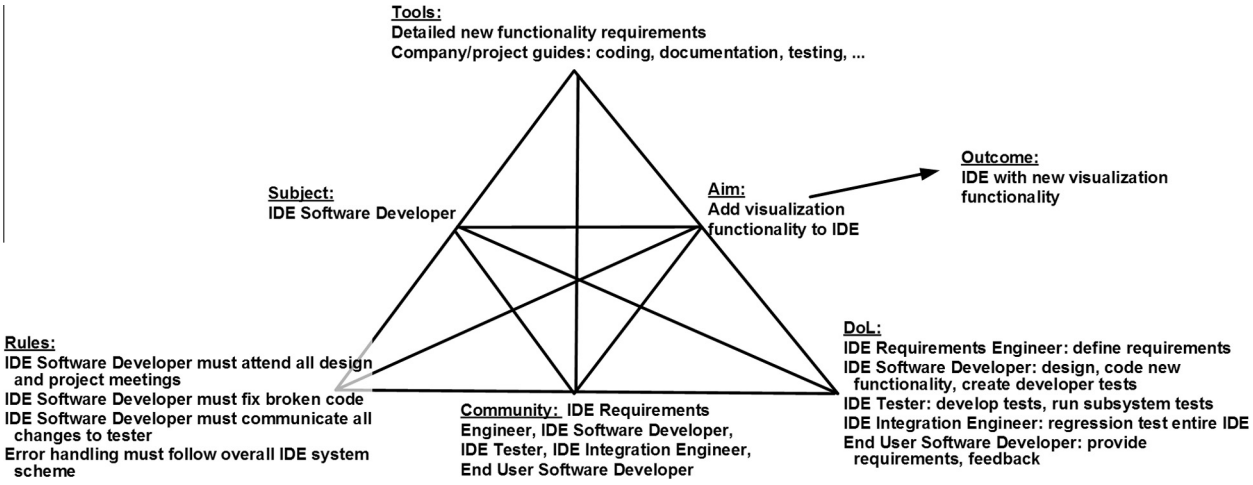

Fig. B-1. Example ASD.

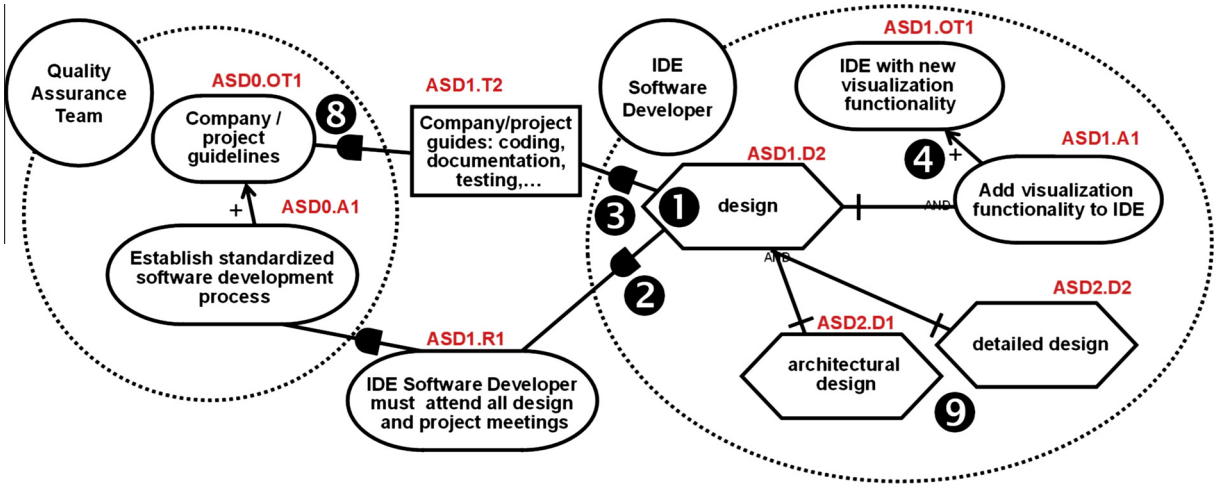

Fig. B-2. URN goal model for new visualization feature (excerpt).

between the IDE Software Developer and IDE Tester/IDE Integration Engineer is further mediated by the rules that (2) the IDE Software Developer must fix any broken code, and (3) the IDE Software Developer must keep the Tester apprised of all changes in the new functionality. Another example rule is that (4) the Developer must adhere to any system-wide protocols for error handling. The rules shown in the ASD example focus only on the IDE Developer for brevity, but do indeed also include rules for the IDE Requirements Engineer, IDE Tester, and IDE Integration Engineer. Other *mediating relationships*: As we noted earlier, rules may also mediate between the subject and object of the activity. In the example of Fig. B-1, the second rule that the IDE Software Developer must fix any broken code is an example of such a mediating rule. Another possible mediation is that of a tool between the subject and community. An example of this kind of mediation is that of the first tool, detailed functionality requirements, which is part of a mediating relation between the IDE Software Developer and the IDE Requirements Engineer. An obvious mediating relation between the subject and object by a DoL is that the IDE Requirements Engineer is responsible for defining the requirements that will be used by the IDE Software Developer.

The following are the additional link conditions that can be applied between ASD models and trace-linked goal models.

(a) In a single ASD, if there is a relation between an AT DoL instance and an AT Community instance, then the respective trace-linked GRL Task instance must be related to the respective GRL Actor instance via the *actor/elems* relation (i.e., the Task resides within the Actor). For example in Fig. B-1, the *IDE Software Developer* is responsible for the DoL item *design* and, hence, Fig. B-2 shows that the task *design* belongs to the actor *IDE System Developer* (see ❶).

(b) In a single ASD, if there is a relation between an AT Rule instance and an AT DoL instance, then there must be a *dependency* relation between the respective trace-linked GRL Goal or Softgoal instance and the respective GRL Actor instances related to the respective trace-linked GRL Task instance. Consider, for example, the first rule in Fig. B-1. Fig. B-2 shows that the task *design* depends on the goal *IDE Software Developer must attend all design and project meetings* (see ❷).

(c) In a single ASD, if there is a mediation relation between an AT Tool instance and an AT Mediation instance consisting of an AT Subject instance (related to an AT Community instance) and an AT Aim instance, then there must be a dependency relation between the respective trace-linked GRL Resource instance and the respective GRL Actor instance related to the AT Community instance. For example in Fig. B-1, the second tool *Company/project guides: coding, documentation, testing, ...* mediates between the subject *IDE Software Developer* and the aim of the ASD. Therefore, Fig. B-2 shows that the task *design* of the actor *IDE Software Developer* depends on resource *Company/project guides: coding, documentation, testing, ...* (see ❸).

(d) In a single ASD, there is a positive contribution link between respective GRL Goal instances that are trace-linked to AT Aim instances and respective GRL Goal instances that are trace-linked to AT Outcome instances. For example, consider the aim and outcome of the ASD in Fig. B-1. Fig. B-2 shows that the goal *Add visualization functionality to IDE* contributes positively to the goal *IDE with new visualization functionality* (see ❹).

(e) All new GRL IntentionalElement instances must be trace-linked to AT *Element* specialization instances that belong to a particular ASD instance (specified via *eleInASD* relations). If more than one ASD is represented in the goal model, then modeler intervention may be needed to accomplish this specification. Once such specifications have been identified, instances are related to the ASD instance using the *eleInASD* relation. For the example in Fig. B-2, this means that all goal model elements are traced to ASD model elements as indicated by the goal model elements annotated with ASD references.

(f) A GRL Task instance with a relation to a GRL Actor (via the *elems* role in Fig. 4) dictates a *whoDoesDoL* relation between the corresponding DoL and Community instances in the ASD. This is also indicated by ❶ in Fig. B-2 as this is the reverse relationship of the relationship described in (i) above.

(g) A GRL Goal or Softgoal instance with a dependency relation to a GRL Task dictates a *dols2rule* relation between the corresponding Rule and DoL instances in the ASD. This is also indicated by ❷ in Fig. B-2 as this is the reverse relationship of the one described in (ii) above.

(h) If the new AT element is an Outcome, and the GRL Goal has a decomposition or dependency link with another GRL element that is associated with an AT element Rule, Tool, or DoL in another ASD than the new Outcome, then a network relation (*enableEleReqOut*) should be added between these AT elements in the ASD network model. For example in Fig. B-2, this is the case for the resource *Company/project*

guides: coding, *documentation*, *testing*,... which depends on the outcome of another ASD (indicated by the goal annotation ASD0.OT1) involving the Quality Assurance Team and is used as a tool in the ASD in Fig. B-2 (see ❽).

(i) If a GRL Task is decomposed into other Tasks, or is dependent on Goals, Softgoals, or Tools that the modeler has indicated are part of a different ASD, then the DoL associated with the Task is hierarchically related (via the *refDiagParDoL* relation) to the other ASDs, and this relation can be created in the ASD Network Model. For example in Fig. B-2, this is the case for the sub-tasks of task *design* (see ❾). The modeler indication is shown by the task annotations ASD2.D1 and ASD2.D1. The DoL items related to these sub-tasks are described on yet another ASD that is linked to the DoL item of the ASD in Fig. B-1 via a hierarchical relationship.

References

[1] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, E. Yu, Evaluating goal models within the goal-oriented requirement language, Int. J. Intell. Syst. (IJIS) 25 (8) (2010) 841–877.

[2] D. Amyot, G. Mussbacher, User Requirements Notation: the first ten years, the next ten years, J. Softw. 6 (5) (2011) 747–768.

[3] P. Andreev, W. Michalowski, C.E. Kuziemsky, S. Hadjiyannakis, Application of activity theory to elicitation of user requirements for a computerized clinical practice guideline: the ActCPG conceptual framework, in: Proceedings of the 20th European Conference on Information Systems (ECIS 2012), 2012. <http://aisel.aisnet.org/ecis2012/205>.

[4] H. Arrow, J.E. McGrath, J.L. Berdahl, Small Groups as Complex Systems: Formation, Coordination, Development and Adaptation, Sage, Thousand Oaks, CA, 2000.

[5] P. Barthelmess, K.M. Anderson, A view of software development environments based on activity theory, Comput. Support. Coop. Work 11 (1–2) (2002) 13–37.

[6] O.K. Basharina, An activity theory perspective on student-reported contradictions in international telecollaboration, Lang. Learn. Technol. 22 (2) (2007) 82–103.

[7] H. Beyer, K. Holtzblatt, Contextual Design: Defining Customer-Centered Systems, Morgan Kaufmann, San Francisco, 1998.

[8] J.B. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, C. Dixon, Home Automation in the wild: challenges and opportunities, in: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI 2011), ACM, 2011, pp. 2115–2124.

[9] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, Dordrecht, USA, 2000.

[10] P. Collins, S. Shukla, D. Redmiles, Activity theory and system design: a view from the trenches, Comput. Support. Coop. Work 11 (1–2) (2002) 55–80.

[11] L. Constantine, Human activity modeling: toward a pragmatic integration of activity theory and usage-centered design, in: Ahmed Seffah, Jean Vanderdonckt, Michel C. Desmarais (Eds.), Sixth International Workshop "From Agent Theory to Agent Implementation", held at the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Human-Centered Software Engineering Software Engineering Models, Patterns and Architectures for HCI, 2008, pp. 27–51. http://dx.doi.org/10.1007/978-1-84800-907-3.

[12] J.B. Earp, A.I. Antón, O. Jarvinen, A social, technical and legal framework for privacy management and policies, in: Americas Conference on Information Systems (AMCIS), Paper 89, 2002.

[13] L. Eisen, J.M. Bieman, S. Ghosh, S. Lozano-Fuentes, Using Cell Phones for Entry to a Dengue Decision Support System, Research Project supported by Award Number R21AI080567 of the National Institute of Allergy and Infectious Diseases, 2009-2012. <http://www.cs.colostate.edu/ddss/> (acc. January 2013).

[14] Y. Engeström, Learning by Expanding, Orienta-Konsultit, Helsinki, 1987.

[15] K. Foot, Cultural-historical activity theory as practical theory: illuminating the development of a conflict monitoring network, Commun. Theory 11 (1) (2001) 56–83.

[16] R. Fuentes-Fernández, J.J. Gómez-Sanz, J. Pavón, Managing contradictions in multi-agent systems, IEICE Trans. Inf. Syst. E90-D (8) (2007) 1243–1250.

[17] G. Georg, R.B. France, An Activity Theory Language: USE Implementation, Colorado State University Computer Science Department Techn. Report, CS-13-101. <http://www.cs.colostate.edu/TechReports/Reports/2013/tr13-101.pdf> (acc. January 2013).

[18] C.A. Halverson, Activity theory and distributed cognition: or what does CSCW need to do with theories?, Comput Supported Cooper. Work (CSCW) 11 (1) (2002) 243–267.

[19] H. Hasan, Integrating IS and HCI using activity theory as a philosophical and theoretical basis, Aust. J. Inform. Syst. (AJIS) 6 (2) (1999) 44–55.

[20] M. Hasu, Y. Engeström, Measurement in action: an activity-theoretical perspective on producer-user interaction, Int. J. Human-Comput. Stud. 53 (1) (2000) 61–69.

[21] E. Hutchins, Cognition in the Wild, MIT Press, Cambridge, MA, 1995.

[22] Innovative Vector Control Consortium, Information Systems, Dengue Decision Support System website. <http://www.ivcc.com/projects/ddss.htm> (acc. January 2013).

[23] ITU-T, User Requirements Notation (URN) – Language definition, ITU-T Recommendation Z.151 (10/12), Geneva, Switzerland, 2012. <http://www.itu.int/rec/T-REC-Z.151/en> (acc. January 2013).

[24] jUCMNav Website, Version 6.0, University of Ottawa. <http://softwareengineering.ca/jucmnav> (acc. September 2014).

[25] V. Kaptelinin, B.A. Nardi, C. Macaulay, The activity checklist: a tool for representing the space of context, Interact. Mag. 6 (4) (1999) 27–39.

[26] D.S. Kolovos, R.F. Paige, F.A.C. Polack, On the Evolution of OCL for Capturing Structural Constraints in Modelling Languages, Rigorous Methods for Software Construction and Analysis, LNCS, vol. 5115, 2009, pp. 204–218.

[27] M. Korpela, H.A. Abimbola, K.C. Olufokunbi, Activity analysis as a method for information system development, Scand. J. Inf. Syst. 12 (2000) 191–210.

[28] S. Leblanc, G. Mussbacher, J. Kienzle, D. Amyot, Narrowing the gaps in concern-driven development, in: 2nd International Model-Driven Requirements Engineering Workshop (MoDRE 2012 at RE 2012), Chicago, Illinois, USA, 2012. http://dx.doi.org/10.1109/MoDRE.2012.6360085.

[29] A.N. Leont'ev, Activity, Consciousness, and Personality, Prentice-Hall, 1978.

[30] Y. Liu, Y. Su, X. Yin, G. Mussbacher, Combined goal and feature model reasoning with the user requirements notation and jUCMNav, in: 22nd International Requirements Engineering Conference (RE 2014), IEEE CS, Karlskrona, Sweden, 2014, pp. 27–36. http://dx.doi.org/10.1109/MoDRE.2014.6890823.

[31] S. Lozano-Fuentes, S. Ghosh, J.M. Bieman, D. Sadhu, L. Eisen, F. Wedyan, E. Hernandez-Garcia, J. Garcia-Rejon, D. Tep-Chel, Using cell phones for mosquito vector surveillance and control, in: 24th International Conference on Software Engineering & Knowledge Engineering (SEKE'12), Knowledge Systems Institute Graduate School, 2012, pp. 763–767.

[32] M.L. Markus, A. Majchrazak, L. Gasser, A design theory for systems that support emergent knowledge processes, MIS Quart. 26 (3) (2002) 179–212.

[33] L.E.G. Martins, Activity theory as a feasible model for requirements elicitation processes, Sci. Interdiscipl. Stud. Comput. Sci. 18 (1) (2007) 33–40.

[34] S. Mennicken, E.M. Huang, Hacking the natural habitat: an in-the-wild study of smart homes, their development, and the people who live in them, in: Proceedings Pervasive 2012, LNCS, vol. 7319, Springer, 2012, pp. 143–160.

[35] E. Murphy, M.A. Rodriguez-Manzanares, Using activity theory and its principle of contradictions to guide research in educational technology, Aust. J. Educat. Technol. 24 (4) (2008) 442–457.

[36] G. Mussbacher, J. Araújo, A. Moreira, D. Amyot, AoURN-based modeling and analysis of software product lines, Softw. Qual. J. 20 (3–4) (2012) 645–687.

[37] G. Mussbacher, J. Kienzle, D. Amyot, Transformation of aspect-oriented requirements specifications for reactive systems into aspect-oriented design specifications, in: 1st Model-Driven Requirements Engineering Workshop (MoDRE 2011 at RE 2011), IEEE CS, Trento, Italy, 2011, pp. 39-47. http://dx.doi.org/10.1109/MoDRE.2011.6045365.

[38] G.C. Neto, A.S. Gomes, J.B. Castro, Mapping activity theory diagrams into i* organizational models, J. Comput. Sci. Technol. (JCS&T) 5 (2) (2005) 57–63.

[39] I. Núñez, Activity Theory and the Utilisation of the Activity System According to the Mathematics Educational Community, Special Issue of Educate, December, 2009, pp. 7–20.

[40] Object Management Group: Object Constraint Language (OCL) 2.3.1, 2012. <http://www.omg.org/spec/OCL/2.3.1/> (acc January 2013).

[41] Object Management Group: Unified Modeling Language (UML) 2.4.1, 2011. <http://www.omg.org/spec/UML/2.4.1> (acc Jan. 2013).

[42] R.F. Paige, N. Drivalos, D.S. Kolovos, K.J. Fernandes, C. Power, G.K. Olsen, S. Zschaler, Rigorous identification and encoding of trace-links in model-driven engineering, Softw. Syst. Model. 10 (4) (2011) 469–487.

[43] P. Runeson, M. Höst, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, Wiley, 2012.

[44] F. Sadri, Ambient intelligence. A survey, ACM Comput. Surv. 43 (4) (2011) 36.

[45] H. Sharp, Y. Rogers, J. Preece, Interaction design: beyond human–computer interaction, second ed., Wiley, 2007.

[46] J. Spiegel, S. Bennett, L. Hattersley, M.H. Hayden, P. Kittayapong, S. Nalim, D.N.C. Wang, E. Zielinski-Gutiérrez, D. Gubler, Barriers and bridges to prevention and control of dengue: the need for a social-ecological approach, EcoHealth 2 (2005) 273–290.

[47] L. Takayama, C. Pantofaru, D. Robson, B. Soto, M. Barry, Making technology homey: finding sources of satisfaction and meaning in home automation, in: Proceedings 14th International Conference on Ubiquitous Computing (UbiComp 2012), ACM, 2012, pp. 511–520.

[48] USE (The UML-based Specification Environment) Website, University of Bremen. <http://sourceforge.net/apps/mediawiki/useocl/index.php?title=Main_Page> (acc. June 2014).

[49] H. van den Berg, R. Velayudhan, A. Ebol, B.H.G. Catbagan, R. Turingan, M. Tuso, J. Hii, Operational efficiency and sustainability of vector control of malaria and dengue: descriptive case studies from the Philippines, Malaria J. 11 (2012) 269.

[50] V. Vanlerberghe, M.E. Toledo, M. Rodríguez, D. Gomez, A. Baly, J.R. Benitez, P. Van der Stuyft, Community involvement in dengue vector control: cluster randomised trial, Br. Med. J. 338 (2009) b1959.

[51] K.J. Vicente, Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work, Lawrence Erlbaum Associates, Mahwah, N.J., 1999.
[52] L.S. Vygotsky, Mind in society: The Development of Higher Psychological Processes, Harvard University Press, Cambridge, MA, 1978 (written 1931).
[53] E. Yu, Modeling Strategic Relationships for Process Reengineering, Ph.D. Thesis, Department of Computer Science, University of Toronto, Canada, 1995.
[54] J.P. Zappen, T.M Harrison, Intention and motive in information-system design: toward a theory and method for assessing users' needs, Digital Cities 3: Information Technologies for Social Capital, LNCS, vol. 3081, Springer, 2005, pp. 354–368.