Requirements Model Generation to Support Requirements Elicitation: The Secure Tropos Experience

Nadzeya Kiyavitskaya (nadzeya@disi.unitn.it)
Dipartimento di Ingegneria e Scienza dell'Informazione - Università di Trento

Nicola Zannone (zannone@cs.toronto.edu)
Department of Computer Science - University of Toronto

Abstract. In the last years several efforts have been devoted by researchers in the Requirements Engineering community to the development of methodologies for supporting designers during requirements elicitation, modeling, and analysis. However, these methodologies often lack tool support to facilitate their application in practice and encourage companies to adopt them.

In this paper, we present our experience in the application of methods for the transformation of requirements specifications expressed in natural language into semi-structured specifications. More specifically, we apply a lightweight method for extracting requirements from system descriptions in natural language to support the Secure Tropos methodology during requirements elicitation phase. Our proposal is based on Cerno, a semantic annotation environment, which uses high-speed context-free robust parsing combined with simple word search. To evaluate our proposal, we discuss its application to the requirements elicitation process followed in the course of a European project on four industrial case studies.

Keywords: Lightweight text analysis, Model generation

1. Introduction

Requirements elicitation has received particular attention in the last years (Bresciani et al., 2004; Cordes and Carver, 1992; Mich and Garigliano, 2002; van Lamsweerde et al., 1991) since it has been recognized as an essential part of the software development process. Requirements elicitation is the process of discovering system requirements through consultation with stakeholders, from system documents, domain knowledge, or any other means of information (Ratchev et al., 2003). Yet, most work in this field has addressed the design of systematic methodologies, but such methodologies are unlikely coupled with tools supporting the entire elicitation process. For example, many existing off-the-shelf technologies, such as IBM Rational RequisitePro (IBM, 2007), assist requirements engineers in authoring and organizing requirements specifications, but do not offer facilities for the analysis of the documentation provided by stakeholders and the extraction of requirements from it. To promote the use of systematic requirements elicitation methods already proved effective in real-life applications, we need to devise tools that support the entire elicitation process.

We had evidence of these issues in the course of the SERENITY project,[1] where the SI* modeling language (Massacci et al., 2007) and the Secure Tropos methodology (Giorgini et al., 2005) have been adopted for the development of Security and Dependability (S&D) patterns at organizational level. As in many other methodologies, the requirements elicitation process in Secure Tropos consists of an interactive knowl-

[1] EU-IST-IP 6th Framework Programme – SERENITY 27587 – http://www.serenity-project.org

edge exchange between company employees, requirements engineers, and security experts. In particular, four milestones for requirements elicitation were identified during the application of Secure Tropos in different European and national projects. (1) First, a preliminary informal description of application domains is provided by the industrial partners, who are domain experts. (2) This informal description is then revised and refined by focusing on a number of scenarios that address issues of the application domain specific to the particular scope of the project. Industrial partners were requested to describe those scenarios in terms of the concepts supported by the modeling language, but still in natural language. Our previous experience, e.g., (Massacci et al., 2005; Massacci and Zannone, 2008), revealed that the modeling phase may be very laborious and require from several weeks to several months if this description is completely unstructured. Indeed, at this stage of the project industrial partners did not have sufficient expertise in the use of methodology to draw requirements models on their own, and, at the same time, requirements engineers did not have the full knowledge of the application domain. (3) To bridge this gap, industrial partners were required to represent the description of scenarios in a semi-structured manner. This semi-structured scenario description was then partially revised by both domain experts and requirements engineers. (4) The final step involves the graphical representation of requirements and their revision on the basis of domain experts' knowledge and the feedback provided by the formal analysis techniques supplied by the methodology. The outcome of this phase (and of the elicitation process) is a number of graphical requirements models of the system to be developed.

Though this requirements elicitation process has provided encouraging results, its execution met a number of difficulties. In particular, the industrial partners initially considered the step of representing requirements in a semi-structured manner as duplicating the work: their claim was that they had already provided the domain description in natural language. Therefore, we are interested in developing methods and tools that support analysts from the very first stages of the extraction of requirements from existing textual scenario descriptions. This will surely increase the acceptance of the methodology by facilitating the interaction between system designers and stakeholders. The objective of this paper is to report our experience in the application of such techniques.

In this paper, we present the application of a tool supported method for the transformation of unstructured natural language requirements specifications into a semi-structured form, with the intent of supporting the Secure Tropos methodology during the requirements elicitation phase. For document analysis we have used Cerno (Kiyavitskaya et al., 2006), a lightweight semantic annotation method that leverages highly efficient techniques and tools adapted from the software analysis and markup domain. Unlike Natural Language Processing (NLP)-based semantic annotation methods (Handschuh et al., 2002; Nobata and Sekine, 1999; Vargas-Vera et al., 2002), Cerno does not use any linguistic information to infer annotations. Rather, it identifies instances of concepts from a semantic model of an application domain through the use of pattern-based rules specific to such a domain. This approach allows for the development of a wide range of semantic annotation tools on top of Cerno in order to support different annotation models

and the analysis of different types of input documents. For instance, Cerno has been applied in the tourism sector (Kiyavitskaya et al., 2006; Kiyavitskaya et al., 2007b) and in the domain of regulatory documents to extract obligations and rights (Kiyavitskaya et al., 2007a; Zeni et al., 2008). The approach, however, needs a preliminary adaptation of the architecture of Cerno to each particular application domain. In the present paper, we illustrate the steps necessary to adapt the architecture in order to identify information that would populate SI* models. The resulting computer-aided support aims to assist analysts during requirements elicitation reducing the human effort spent in the analysis of textual requirements specifications, but does not pretend to completely substitute human analysts. Analysts may want to check annotated documents in order to resolve possible ambiguities in textual requirements specifications or refine them, as it is usually required by the requirements elicitation process. Nevertheless, providing analysts with a first draft of the requirements model from informal textual descriptions can largely speed up the development process because domain and modeling experts can start revising the requirements model much earlier.

To evaluate the proposed approach, we have applied Cerno to four industrial case studies that have been modeled and analyzed using Secure Tropos in the SERENITY project. These scenarios are e-Business, Smart Items, e-Government, and Air Traffic Management (Campadello et al., 2006a). They describe hypothetical situations arising from the application of new technologies. In this paper, we compare the results obtained by applying Cerno with those obtained by humans in the course of the project. The extracted requirements are then used in the

project to drive security engineers in the development of S&D patterns. In a different context, these requirements can be used to develop secure software systems.

The remainder of the paper is organized as follows. Section 2 provides insights into the Secure Tropos methodology and SI* modeling framework. Section 3 introduces the document analysis method Cerno. Section 4 presents the requirements elicitation process based on Cerno. Thereafter, Section 5 presents an evaluation of the application of Cerno to the case studies and Section 6 discusses the lesson learned. Finally, Section 7 discusses related work and Section 8 concludes the paper with some directions for future work.

2. The Secure Tropos Methodology

Secure Tropos (Giorgini et al., 2005) is an agent-oriented requirements engineering methodology tailored to model and analyze security and privacy requirements of socio-technical systems and verify their consistence with functional requirements from the early phases of the system development process. Secure Tropos has been applied to several industrial case studies (e.g., (Asnar et al., 2006; Massacci et al., 2005; Massacci and Zannone, 2008)) for the analysis of security and privacy requirements and demonstrated successful results.

The methodology adopts the SI* language (Massacci et al., 2007) for modeling requirements. This language extends i* (Yu, 1995) with concepts adequate to model security concerns. It adopts the concepts of *actor*, *goal*, *softgoal*, *task*, and *resource*; also *objective*, *entitlement*,

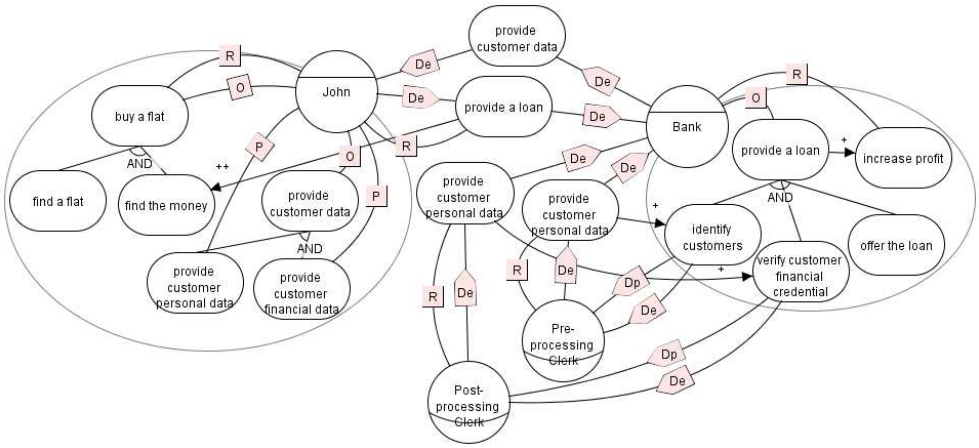

Figure 1. SI* Model

and *capability* to indicate goals, tasks, and resources that actors want to achieve, are authorized to achieve, and are able to achieve, respectively. The language uses the notions of *execution dependency* (or *delegation of execution*) to model assignments of duties between two actors – the *depender* and the *dependee* – and *permission delegation* to model the transfer of authority between two actors – the *delegator* and the *delegatee*. Finally, *trust of execution* and *trust of permission* are used to model the expectation of an actor – the *trustor* – about respectively the performance and fair behavior of another actor, the *trustee*, on a certain goal, task, or resource. SI* is based on the idea of building a model of the system that is incrementally refined and extended. Similarly to i*, goals are analyzed from the perspective of single actors using three techniques, namely *AND/OR decomposition*, *contribution analysis*, and *means-end analysis*.

To illustrate the application of the SI* modeling framework, the SI* model corresponding to a fragment of the e-Business scenario is shown in Fig. 1. In the center of the scenario story there is John, a "single"

man of 25 years old. He wants to buy a flat and needs to find money to achieve his objective. Thus, he decides to ask for a loan from his bank. The procedure adopted by the bank for providing a loan roughly consists of identifying the customer, for which the bank depends on the pre-processing clerk, verifying the customer's financial credentials, for which the bank depends on the post-processing clerk, and offering a loan. The pre-processing clerk needs to access the customer's personal data to achieve assigned duties, whereas the post-processing clerk needs both personal and financial data of the customer. Both clerks require the bank to provide the necessary information. In turn, the bank asks this information from John, who is the owner of his data. In the figure, objectives, entitlements, and capabilities are represented using request (edges marked as **R**), own (edges marked as **O**), and provide (edges marked as **P**) relations, respectively. Permission delegations and execution dependencies are respectively represented by edges labeled with **Dp** and **De**.

Once the requirements model is captured, Secure Tropos uses formal analysis techniques for the verification and validation of this model, checking its compliance with security requirements and the consistency among security and functional requirements. However, in this paper we do not present the requirements analysis process and the formal framework that supports it. The description of both can be found in (Giorgini et al., 2005). Instead, the focus of this work is the requirements elicitation process.

Secure Tropos supports system analysts during requirements elicitation by providing them with a *requirements collection schema* (RCS). This schema allows analysts to represent the description of the system

in a semi-structured way along the lines of the VOLERE methodology (Robertson and Robertson, 1999). The schema provides a tabular template, intended to bridge the gap between requirements specified in natural language and their formal specification. Accordingly, application scenarios are analyzed from the following viewpoints:

Actor Viewpoint: relevant actors (i.e., agents and roles) are described along with their objectives, capabilities, and entitlements;

Goal Modeling Viewpoint: goals are analyzed and refined into subgoals and tasks from the perspective of single actors;

Relational Viewpoint: social relations among actors are identified;

Opportunities and Threats Viewpoint: opportunities and risks for actors are identified and analyzed;

Privacy Viewpoint: privacy aspects are captured by identifying assets of actors and relating them to the actors who request them by making explicit the purpose for such requests.

Table I shows an example of the RCS by reporting a fragment of the Actor Viewpoint for the e-Business scenario from (Asnar et al., 2006).

Once the RCS is filled, the drawing of SI* models representing the system description results to be straightforward. Indeed, there is a one-to-one relationship between SI* concepts and the tables in the schema.

In summary, the requirements model generation process has been done in two steps as shown in Fig. 2. First, information is collected by manually filling the RCS. Then, semi-structured information is used to draw SI* models. However, collecting relevant facts for the template is laborious and time-consuming process. Stakeholders usually provide

Table I. Actors' Objectives

Agent/Role	**Objectives**	**Comments (if any)**
John	buy a flat	
Bank	increase profit	
Pre-processing Clerk	access customer personal data	The achievement of this goal permits the pre-processing clerk to achieve his duties, that is, to identify customers.
Post-processing Clerk	access customer data	The achievement of this goal permits the post-processing clerk to achieve his duties, that is, to verify customer's financial credential.

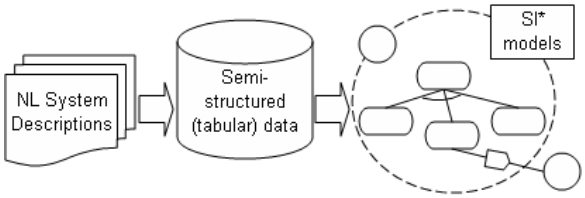

Figure 2. Requirements Model Generation in Secure Tropos

large size documents detailing the application domains together with additional specific requirements that their systems have to meet. In this setting, human designers may overlook important facts. This information can be analyzed more effectively using computer supported tools. Moreover, the company's legacy knowledge available in the form of various textual descriptions could be also involved to support analysts in the elicitation process.

The purpose of this work is to support analysts during the requirements elicitation phase by generating SI* models from scenario

description provided in natural language. In particular, we are interested in tools that can assist analysts in extracting requirements from textual descriptions and representing extracted requirements in a semi-structured form such as, for instance, the tabular form of the RCS. The following section describes the approach we are going to employ for this task.

3. The Document Analysis Methodology (Cerno)

Cerno (Kiyavitskaya et al., 2006) is a document analysis and annotation framework that uses highly efficient methods and tools adopted from the software analysis and markup domain. Its architecture is based on the *design recovery process*, i.e., the analysis and markup of source code according to a semantic design theory, which shares many similarities with the task of semantic annotation of natural language documents. The underlying technology used by Cerno is the Turing eXtender Language (TXL) (Cordy, 2006), a generalized parsing and structural transformation system, developed to support computer software analysis and source transformation tasks.

The approach discriminates between domain-dependent and independent components of the annotation process and thus allows for easy adaptation to different application domains and tasks. Although an initial effort is needed to adapt Cerno to a specific domain, this activity does not require large quantities of training data. Neither expertise in linguistics nor programming skills are needed from the analyst. An example of how Cerno can be adapted for the analysis of new domains

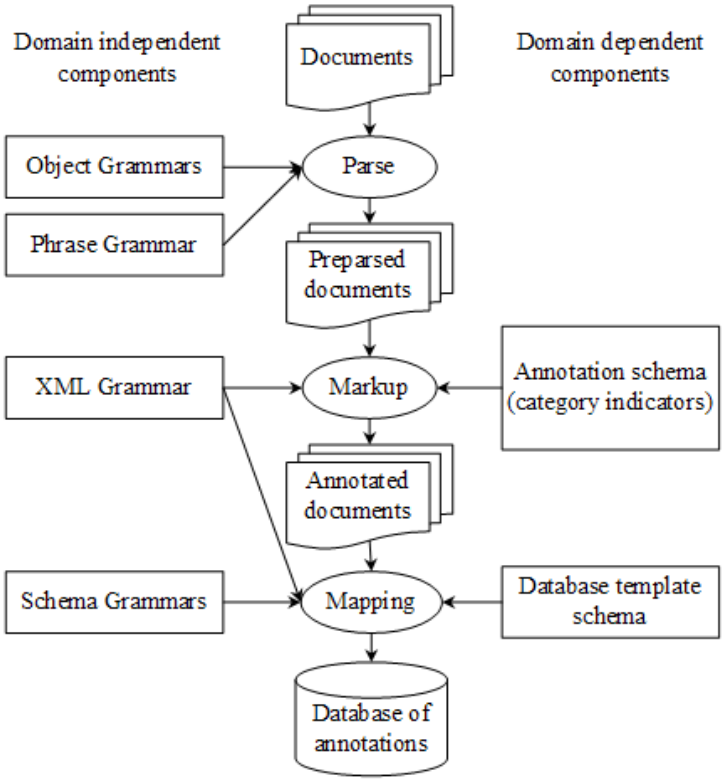

Figure 3. The Cerno's Architecture

is presented in Section 4 where we reported our experience in adopting it to Secure Tropos.

The architecture of Cerno (Fig. 3) consists of a number of consequent transformations:

1. *Parse.* The tool parses an input document breaking it down into its constituents according to a predefined document grammar that is domain independent. The produced parse tree consists of structures such as document, paragraph, phrase, and word. Any of these structures can be chosen as annotation unit, depending on the purpose of annotations. At the same time, complex word-equivalent

```
% nonterminal types are enclosed in square brackets
define program
 [repeat paragraph]
end define

% sequences of one or more non-terminals are defined by using repeat
define paragraph
 [repeat sentence][repeat newline+]
end define

define sentence
 [repeat token_not_fullstop][fullstop]
end define

% vertical bars are used to indicate alternate forms
define token_not_fullstop
 [shortform] | [id] | [token]
end define

% termial symbols are used with a single opening quote
define fullstop
 '.
end define
```

Figure 4. A fragment of the document grammar

objects, such as phone numbers, e-mail and web addresses, and similar structures, are properly recognized using structural patterns of object grammars. Grammars are described in a BNF-like form using the TXL language notation as shown in Fig. 4.

2. *Markup.* This stage uses a domain-dependent annotation schema to infer annotations. The *annotation schema* specifies the annotations to be generated. It is composed of semantic tags along with their syntactic indicators: positive, which point to the presence of a concept instance, and optionally negative, which exclude its presence, i.e., they are counter-indicators. These domain-specific indicators can be derived manually, i.e., proposed by the domain experts, or semi-automatically, for instance, mined from a rich conceptual model representing the domain knowledge (if such a model is available). The processing exploits the structural pattern matching and source transformation capabilities of the TXL engine similarly to what is used for software markup to yield an annotated text. *Syntactic indicators* can be literal words and phrases, or names of parsed entities. Fig. 5 shows the syntax of the rule applied to annotate sentences. It matches each sentence and checks for the presence of the indicators in it. If one of the positive indicators from the list is present in the text fragment and no counter-indicators appear, Cerno annotates this fragment with the corresponding tag.

3. *Mapping.* This stage is optional. It is executed in the case analysts have to store extracted information in an external database. In this stage, annotated fragments are selected from all annotations according to a predefined database schema template and, then, are inserted into the database. The schema is domain dependent and represents a sort of a target template to accommodate annotations relevant to a specific task.

```
rule markupCategories
 import Categories [repeat category]
% every sentence in input is matched once
 replace $ [sentence]
  S [sentence]
% lookup of the indicators for each category of the annotation schema
 by
  S [tagWithCategory each Categories]
end rule

function tagWithCategory Category [category]  deconstruct Category
  Tag [id]: Indicators [list keyphrase];
     Negatives [list keyphrase] .
 replace [sentence]
  S [sentence]
% check that at least one positive indicator is present
 where
  S [contains each Indicators]
% check that no negative indicators is present
 where not
  S [contains each Negatives]
 by
% markup with the name of the category
  S [markup Tag]
end function
```

Figure 5. The annotation rule applied to sentences

Cerno has already been applied to a number of case studies for the analysis of differently structured documents in the Tourism domain (Kiyavitskaya et al., 2006; Kiyavitskaya et al., 2007b) as well as for

the extraction of obligations and rights from the HIPAA regulation (Kiyavitskaya et al., 2007a), demonstrating acceptable performance and scalability while yielding adequate quality results.

Our claim is that this approach can be used also to support the Secure Tropos methodology. In particular, it can be applied to natural language requirements specifications to derive abstractions for SI* concepts.

4. Applying Cerno to Secure Tropos

In our studies, we were primarily interested in obtaining scenario descriptions with embedded annotations of SI* concepts. For this purpose, the first two steps of Cerno have been applied. In order to adapt Cerno to a new domain, it is necessary to supply the components specific to that domain, i.e., the annotation schema. The complete tag set of the annotation schema is derived from the concepts of the SI* modeling language. In particular, we have identified instances of the following concepts: *actor*, *goal*, *softgoal*, *task*, *resource*, *objective*, *entitlement*, *capability*, *execution dependency*, *permission delegation*, *trust of permission*, and *trust of execution*; and also *requester*, *owner*, *provider*, *depender*, *dependee*, *delegator*, *delegatee*, *trustor*, and *trustee*.

The appropriate syntactic indicators are usually instantiated in a by-example manner, i.e., manually collecting contextual key phrases from a handful of input samples. For this purpose, in the tuning phase, a human analyst starts from an example of the system description expressed in natural language and provides a set of contextual indicators

for each concept of interest. Then, the Cerno's processing can be run and results roughly checked to verify if most of the relevant information was found. If necessary, the analysts may add or delete indicators in the wordlist files.

Before defining the list of indicators, let us consider in more detail the nature of the concepts to be identified and the issues they raise. Depending on the inter-connections between SI* concepts, they can be classified into three main sets:

- *Basic entity*: embraces all basic operating elements which form the basis of the scenario and can exist independently from any other entity, such as actor (e.g., "John" and "bank"), resource (e.g., "personal data"), goal (e.g., "buy a flat"), and task (e.g., "calculate a rating");

- *Complex relationship*: includes concepts which bind basic entities together, namely, objective (e.g., "John wants to buy a flat"), capability, entitlement, execution dependency (e.g., "the bank depends on John for obtaining his personal data"), permission delegation (e.g., "the bank delegates the permission to verify customer financial credentials to the post-processing clerk"), trust of permission, and trust of execution;

- *Specific entity role*: denotes the interpretation of basic entities involved in a complex relationship on the basis of their role – owner, trustor, trustee, delegator, delegatee, etc. For example, the post-processing clerk plays the role of delegatee in the permission delegation "the bank delegates the permission to verify customer financial credentials to the post-processing clerk".

This classification entails that the annotation of different conceptual types should be done with different granularity, where *annotation granularity* is the quantity of text associated with a certain concept. In particular, we are interested in generating annotations at the following levels of granularity:

- *sentence level*, in which descriptive statements of complex relationships are identified;
- *word level*, in which all noun phrases that describe basic and specific entities of complex relationships are identified.

Different concept types pose different problems for the extraction process. For instance, basic entities depend on the application scenario and so they should be identified for each particular scenario. We have identified basic entities and, in particular, actors, with high reliability by taking advantage of the definition section of each scenario, where the actors participating in the scenario are described. A fragment of such a description for the e-Business scenario is presented at the beginning of Fig. 6. In summary, the vocabulary of basic entities is derived by hand from the definition section and then used by the tool to automatically recognize all their occurrences in the scenario description.

The tasks of recognizing complex relationships and interpreting entity roles are more challenging. We addressed the first one by enhancing domain-dependent modules of Cerno with a set of complex pattern-based rules. The patterns are expressions combining the markup of basic entities recognized earlier and literal indicators. There is no limitation on the form of patterns, they can use as many elements as necessary and in any order. To identify such patterns, we manually

```
John is a ‘‘single’’ man 25 years old.

Peter is a BBB bank employee, he is quite new in the bank (2 years) and he is a
pre-processing clerk.

Paul is a 10 years experienced BBB bank employee and he plays the role of the
post-processing clerk.

...

The customer delegates to the BBB bank the permission to access and modify his
personal information.

If the manager has delegated to the clerk the signature of the form, Ted trusts
Paul for acting on his behalf.

The BBB bank delegates to the manager and to the clerks the right to access and
process the customer information data.
```

Figure 6. A fragment of the e-Business scenario (Campadello et al., 2006a)

analyzed some of the proposed scenarios. For example, trust relations can be identified using the following pattern: "<Actor> ... </Actor> trusts <Actor> ... < /Actor> in ...". For this specific type of text, we discovered that using only literal verb forms, omitting actor entities, is enough to identify relationships with good precision. Therefore, for the markup of SI* concepts we provided the list of patterns for Cerno in a simplified literal form. Fig. 7 lists some patterns used to markup SI* concepts. In the figure patterns are grouped with respect to concepts. Each group represents a triple of the form [Tag : Positives ; Negatives], where Tag is a label for a concept, Positives specifies the list of indicators that are searched for in text units, and Negatives specifies the list of counter-indicators. If at least one of the positive indicators is matched and negative indicators are not encountered, a text unit is annotated by the concept's label. In total, we have identified 54 patterns for complex relationships.

TrustOfExecution: `trusts , trust ; permission , right .`

TrustOfPermission: `trusts , trust ; .`

ExecutionDependency: `depend , depends , delegates , delegate ; permission , right .`

PermissionDelegation: `delegates , delegate ; .`

Entitlement: `owns , is entitled , is the owner , is the legitimate owner; .`

Objective: `wants , desires , aims , objective of ; .`

Capability : `provides, provide, is able to , are able to, capable to , can , has the capability of , has the capabilities, have the capability of, have the capabilities; .`

Figure 7. The contextual patterns for execution dependency

The order in listing the rules is important, i.e., if the same indicator appears in more than one triple, only the first match is annotated. Notice, for instance, the usage of indicator "trusts" in Fig. 7. This indicator is used for identifying two different concepts: trust of execution and trust of permission. To disambiguate between them, counter-indicators come in handy. A sentence containing indicator "trusts" is recognized by the tool as trust of execution, unless it contains a negative indicator. Specifically, if indicator "trusts" is met and no negative indicators are present, the first pattern is matched and the sentence is annotated using tag `TrustOfExecution`. The second pattern is skipped because only one markup for sentence is allowed. On the contrary, if a counter-indicator is met, the first pattern is not applicable and the tool looks for another applicable pattern. In our case, the second pattern is applied and the sentence is annotated using tag `TrustOfPermission`.

We remark that, though these patterns have been defined in the domain-dependent modules of Cerno, they are independent from a

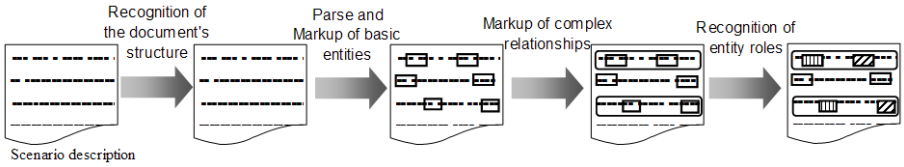

Figure 8. Scenario analysis workflow supported by Cerno

certain scenario. Therefore, they can be reused for the analysis of new application scenarios. However, we do not exclude the possibility of extending the list of patterns for the analysis of new application scenarios.

Finally, specific entity roles are interpreted on the basis of the position of basic entities in complex relationships. For instance, in the case of trust relations, given the sentence written in active voice, the first actor is annotated as `Trustor`, while the latter as `Trustee`. To realize this stage, absent in Cerno before, we augmented the framework with a phase that reuses the patterns for complex relationships to re-annotate basic entities playing key roles in the statement.

Once the vocabulary of basic entities and patterns for relations and specific entity roles are defined, the framework is ready for requirements elicitation. The extraction process is organized as follows (Fig. 8):

1. *Recognition of the document's structure.* This phase parses the scenario description and structures it according to the document grammar in Fig. 4. A fragment of the parse tree produced internally by the annotation engine is given in Fig. 9. Notice that the document is not annotated in this phase.

2. *Markup of basic entities.* This phase annotates the document using the scenario definition vocabulary. For instance, all occurrences of

```
<program>
 <repeat_line>
 ...
  <line>
   <sentence><repeat_token_not_fullstop>
    <token_not_fullstop><id>He</id></token_not_fullstop>
    <token_not_fullstop><id>is</id></token_not_fullstop>
    <token_not_fullstop><id>in</id></token_not_fullstop>
    <token_not_fullstop><id>charge</id></token_not_fullstop>
    <token_not_fullstop><id>of</id></token_not_fullstop>
    <token_not_fullstop><id>all</id></token_not_fullstop>
    <token_not_fullstop><id>air-to-ground</id>
    </token_not_fullstop>
    <token_not_fullstop><id>communication</id>
    </token_not_fullstop> <empty/>
    </repeat_token_not_fullstop>
    </sentence>
    <repeat_fullstop>
    <fullstop>.</fullstop> <empty/>
   </repeat_fullstop>
  </line>
 ...
 </repeat_line>
</program>
```

Figure 9. The internal parse tree produced by Cerno

actors in the document are identified and annotated using the tag `<Actor> ... </Actor>`.

3. *Markup of complex relationships.* This phase identifies complex standard relationships based on the domain- and input-driven patterns (Fig. 7). Cerno matches their presence and annotates full phrases with tags of complex relationships using the annotation rules (Fig. 5). In this phase, for instance, all occurrences of permission delegation in the document are identified and annotated with tag <PermissionDelegation> ... </PermissionDelegation>.

4. *Recognition of entity roles.* This phase assigns the semantic roles to the previously recognized entities depending on their position in the statement describing a relationship. Here, annotations representing basic entities are replaced with annotations representing specific entity roles. See, for instance, the interpretation rule used to recognize the actor that is the subject of a relationship in Fig. 10.

As a result, the tool generates the annotations for the concepts of interest at both word and sentence levels. Fig. 11 shows the annotation of the e-Business scenario's fragment presented at the bottom of Fig. 6.

5. Evaluation

This section describes the evaluation framework used to analyze the results of the application of Cerno to four industrial case studies.

```
% the rule identifies sentences describing delegations and stated
in active voice
function findDelegPhrase
 replace * [phrase]
  P [phrase]
 where P [contains 'delegates] [contains 'delegate]
 by
% the tag name is passed as parameter
  P [markMainRole 'Delegator] [...]
end function

% the rule matches the first actor in the fragment
function markMainRole Tag [id]
 replace * [actor]
  FirstActor [actor]
% the instance of actor will be replaced by its marked-up copy
 by
  FirstActor [markup Tag]
end function
```

Figure 10. The rules used in the interpretation phase

5.1. Evaluation Framework

The evaluation of an annotation tool can be done from different perspectives and applying various assessment criteria. For example, the tool can be evaluated in terms of its usability, processing speed, quality of output, and many other aspects. From our perspective the most important characteristics are the effectiveness of the tool in terms of quality of results and productivity. The motivation underlying this as-

```
<PermissionDelegation>The <Delegator>customer</Delegator> delegates to
the <Delegatee>BBB bank</Delegatee> the permission to access and modify his
personal information</PermissionDelegation>.
<TrustOfExecution><Condition>If the <Actor>manager</Actor> has delegated
to the <Actor>clerk</Actor> the signature of the form</Condition>,
<Trustor>Ted</Trustor> trusts <Trustee>Paul</Trustee> for acting on his
behalf</TrustOfExecution>.
<PermissionDelegation>The <Delegator>BBB bank</Delegator> delegates
to the <Delegatee>manager</Delegatee> and to the clerks the right
to access and process the <Actor>customer</Actor> information
data</PermissionDelegation>.
```

Figure 11. Annotations for the e-Business scenario

sumption is that poor quality results would require the analyst to check them, re-examining textual requirements specifications, and, actually, redo all the work by hand. Productivity aims to evaluate the tool on the basis of the efforts that analysts can save through its use.

To assess the quality of the results obtained using the proposed method, we have considered a set of standard quality metrics adapted from the information retrieval area (Yang, 1999), namely recall, precision, fallout, accuracy, error, and f-measure.

– *Recall* measures the quality of the tool in finding relevant information. It is calculated dividing the number of relevant annotations detected by the number of all relevant annotations.

– *Precision* measures the quality of the tool in not returning irrelevant information. This measure is computed dividing the number of relevant annotations detected by the number of all annotations detected.

- *Fallout* measures how quickly precision drops as recall is increased. It is equal to the number of irrelevant detected annotations divided by the number of irrelevant annotations in the collection and characterizes the degree to which a system's performance is affected by the availability of a large number of irrelevant information.

- *Accuracy* measures how well the tool identifies relevant information and rejects irrelevant one. It is computed as the sum of the number of correctly detected and correctly rejected annotations divided by the number of all annotations.

- *Error* measures how much the tool is prone to accept irrelevant information and reject relevant one. It is calculated by dividing the number of incorrectly detected and incorrectly rejected annotations by the number of all annotations.

- *F-measure* is an aggregate measure of performance. It is typically used to summarize the overall results. It combines precision and recall into a single measure and is calculated as a harmonic mean of recall and precision with equal weights, i.e., double product of these measures divided by their sum.

In order to assess the quality of output, a *reference annotated document*, i.e., the so-called *gold standard*, should be provided as the basis for comparing the desired results with those obtained by the tool. In most cases, it is assumed that any given text fragment in a collection is either pertinent or non-pertinent to a particular concept or topic. If the annotated fragment differs from the correct answer in any way, its selection is counted as an error. Thus, there is no scale of relative relevance.

The availability of the gold standard is often problematic. Ideally, we should compare the automated results against a wide range of high quality human opinions. However, in practice the cost of the human work involved is prohibitive for all but large companies and projects. Specifically, the task of manual annotation is very difficult because of a number of issues:

- annotators must be familiar with domain the document of interest;
- preliminary training and detailed guidelines are necessary for a particular annotation task;
- when non-native speaker annotators are involved, excellent language knowledge is important;
- the process is costly, time-consuming, and error-prone due to humans tiredness, lack of attention, etc.

Nevertheless, even the annotations accurately crafted by experts in a given domain, native-speakers, well acquainted with annotation process, may differ. Actually, humans can have different opinions on the same subject.

The second aspect of the evaluation focuses on the measurement of the efforts for extracting SI* abstractions from scenarios description given in natural language, when they are assisted by the tool. In other words, the evaluation framework aims to verify if the use of the tool increases the productivity of human annotators. To this intent, we consider the time spent by domain experts for requirements elicitation in the course of the project.

5.2. Evaluation Results

We have validated the proposed method by applying it to four industrial case studies that have been proposed in the SERENITY project: e-Business, Air Traffic Management (ATM), e-Government, and Smart Items.

In the course of the project, industrial partners firstly provided a general description of application domains (Campadello et al., 2006b) which was used to identify S&D issues. Such a description has been successively refined by focusing on a set of selected application scenarios. These scenario descriptions (Campadello et al., 2006a) have been used to populate the RCS and, in turn, a number of preliminary requirements models, which have been revised and refined by domain experts on the basis of their knowledge and with the support of the formal framework underlying Secure Tropos. The outcome of this process is a number of graphical models that are presented in (Asnar et al., 2006).

For the evaluation of the results obtained by applying the tool, we have used the RCS produced during the project as a reference annotation document. Given that this schema was filled and iteratively revised by several human analysts of the Serenity project, we assume that uncertainties and ambiguities usually affecting annotations produced by different humans (Kamsties et al., 2001; Sampson and Babarczy, 2003) have been resolved. Therefore, the final RCS can be used as the gold standard to evaluate the quality of automated annotations. Actually, the quality of manual annotations constitutes an upper bound for automatic document analysis. As we are initially interested largely automating the laborious work for humans, the main requirement for

our tool is to produce good quality results within reasonable time and resource limits. The obtained annotations can be then quickly checked and refined by human experts, if necessary.

Two application scenarios – e-Business and ATM scenarios – were used to tune Cerno's domain dependent components, while the other two were completely new for the tool.

5.2.1. *Quality of Results*

According to the evaluation framework, the first aspect is to compute quality rates for the annotations obtained through the tool. To assess such rates, the instances of SI* concepts identified in the description of scenarios by the tool have been compared with the instances present in the gold standard. Table II provides the evaluation of each scenario with respect to the metrics presented in Section 5.1. These numbers are average values over all instances and range between the lowest value 0 and the highest value 1.

E-Business scenario This scenario analyzes a typical loan origination process from a security perspective. When a customer requests a loan from a bank, the bank performs a number of activities to verify the identity and financial credentials of clients. The loan origination process includes those verification activities that shall be executed before selling the loan. In particular, this procedure requires several external and internal ratings in order to check the credit worthiness of customers. In this setting, it is required that information collection and transmission guarantee the confidentiality and integrity of client personal and financial data. The document describing this scenario contained 213

Table II. Overall performance rates

	e-Business	ATM	e-Government	Smart Items
Recall	0.88	0.96	0.97	0.96
Precision	0.98	1.00	0.94	0.80
Fallout	0.04	0.00	0.00	0.03
Accuracy	0.91	0.99	0.96	0.89
Error	0.09	0.01	0.04	0.04
F-measure	0.93	0.98	0.96	0.88

sentences (2144 words). The requirements model specifies 10 actors that were expanded with a total of 65 model elements – 37 goals, 11 softgoals, 13 tasks, and 4 resources. These 65 elements were linked by a total of 89 links (including execution dependency, permission delegation, trust of execution, trust of permission, decomposition).

Comparing the instances identified by the tool with those present in the reference model, we noticed that Cerno was able to find large part of relevant information (88%) and nearly all extracted information (98%) was correct. However, the error measure is quite high, about 9%. Although this scenario was used to tune the tool, average recall and error scores were affected by the fact that some instances were missing. This is mostly because the original text contained many conjunction constructions, which were not properly handled by the tool. An example of this issue has been shown in the third sentence in Fig. 11, where the manager is properly annotated as the delegatee of the permission

delegation, but the clerk is not. We further discuss this problem in Section 6.

ATM scenario This scenario presents an ATM system. It has been described as an aggregation of services provided by ground-based Air Traffic Controllers. Controllers have the responsibility to direct aircraft on the ground and in the air. Their first task is to maintain a separation between aircrafts by preventing them to come too close to each other horizontally and vertically. Secondly, controllers must ensure a correct flow of traffic, providing pilots with information helpful for the flight, such as weather conditions. This scenario focuses, in particular, on re-sectorization and partial airspace delegation processes due to an unplanned increase of air traffic. Since failures in ATM systems results in life loss, particular attention has been addressed to dependability aspects of the system. The length of the description of this scenario was 514 sentences (6580 words). The requirements model specifies 7 actors that were expanded with a total of 119 model elements – 34 goals, 5 softgoals, 55 tasks, and 25 resources. These 119 elements were linked by a total of 152 links (including execution dependency, permission delegation, trust of execution, trust of permission, decomposition).

The estimated results demonstrate high performance rates for recall, about 96%, and excellent precision equal to 100%. Error rate is very low, about 1%. This means that most of the instances were retrieved by the tool and all of them were correct. Recall in this scenario was mainly affected by the presence of coordination conjunctions. For instance, the tool annotated the phrase "<TrustExecution> <Trustor>Robert</Trustor> trusts in the ability of <Trustee>

`Paula` </`Trustee`> to evaluate the acceptability of the incoming traffic flow and to define the requirements for the partial airspace delegation </`TrustExecution`>"as one instance of trust in execution. On the other hand, human analysts in the course of the projects have represented this information as two distinct trust relationships: one trust relationship for "evaluate the acceptability of the incoming traffic flow" and another trust relationship for "define the requirements for the partial airspace delegation". This reveals that the tool did not actually miss relevant information, rather it was not able to annotate the document at the same level of granularity as human analysts did. More specifically, while the tool annotates complex relationships at sentence level (see Section 4), humans were more flexible in this respect identifying relationships at subphrase level. Therefore, the tool settings need to be revised to distinguish multiple instances of relationships in one phrase. In this scenario, fallout rate is less then 1%. It shows that precision of results remains practically unaffected with augmentation of unrelated information. The quality of the results for this case study is not surprising given that it was used to tune the tool.

E-Government scenario This scenario describes the use of technology by governmental agencies to support citizens in accessing on-line government services and making on-line transactions. In particular, it presents a fiscal portal and the on-line services it offers to citizens and companies. These services include on-line declaration and payment of the VAT, on-line support for filling fiscal forms, income tax calculation, etc. The scenario shows how the use of technology allows agencies to simplify administrative procedures for citizens, companies, and local

authorities, and to enable citizens to control the use of their personal data by public bodies. At the same time, the scenario shows how the use of technologies spawns new security issues, such as the identification and authentication of tax-payers and anonymous consulting of the tax information database, which are the focus of this scenario. The length of the description of this scenario was 177 sentences (2023 words). The requirements model specifies 6 actors that were expanded with a total of 50 model elements – 37 goals, 2 softgoals, 6 tasks, and 5 resources. These 65 elements were linked by a total of 61 links (including execution dependency, permission delegation, trust of execution, trust of permission, decomposition).

The tool has achieved good results both in terms of recall and precision, equal to 97% and 94%, respectively, with error rate about 4%. This means that the tool was able to retrieve almost all relevant information and to correctly discard major fraction of irrelevant information. The recall rate was mainly decreased due to the fact that several instances of trust relationships appearing in the goal standard do not have a counterpart in the textual scenario description. Similarly to the ATM case study, fallout score approaching to zero points out that precision of results is not damaged by the availability of large amount of irrelevant information.

Smart Items scenario This case study analyzes a health care domain where patients are monitored by a sensor network. In this setting, patients are equipped with smart devices which observe their health conditions 24 hours a day and transmit collected data to a health care center. This scenario presents a number of emergency situations where

these devices handle the patient's health troubles. Since medical data are recognized by law as sensitive data, several privacy issues arise from this case study, which have been analyzed from a legal perspective. The scenario description was composed of 389 sentences (4397 words). The requirements model specifies 14 actors, 12 of them were expanded with a total of 79 model elements – 58 goals, 9 softgoals, 6 tasks, and 6 resources. These 79 elements were linked by a total of 122 links (including execution dependency, permission delegation, trust of execution, trust of permission, decomposition).

In this experiment, the tool has shown very high recall, about 96%. However, the precision rate is relatively low, about 80%. This detriment is caused mostly by the reason that the type of delegation[2] and trust relationships (i.e., execution or permission) was incorrectly determined in some cases. In particular, generic trust statements were interpreted by domain experts as both trust of execution and trust of permission.

5.2.2. *Productivity*

The second aspect of the evaluation framework focuses on the measurement of the time required to analyze the four scenarios manually and compares it with the time spent by applying the proposed approach. A couple of weeks of one person's work were spent to adapt Cerno to the new domain. Once the tuning phase was completed, the annotation process is almost instant: the time spent by Cerno for processing textual scenario descriptions took 20-34 msec for all case studies on Intel Pentium 4, 2.60GHz, Ram 512 MB, running Windows XP.

[2] Recall that execution dependency is also called delegation of execution.

In comparison, in the context of the project the requirement elicitation phase required about 3 months for each case study. However, this time was spent not only to fill the RCS from informal scenario descriptions and draw graphical requirements models, but also to iteratively revise requirements models. The requirements elicitation phase for each case study involved the work of groups of 3-5 people, from which 1 or 2 were experts in the use of the methodology while the others were experts in the specific application domain.

6. Lessons Learned

The evaluation results has revealed a number of drawbacks and, most importantly, indicated useful hints on how the tool can be improved. One of the problems was that due to the use of conjunction constructions in the informal description of scenarios (especially in the e-Business scenario), which were not properly handled by the tool, affecting all evaluation rates and, in particular, recall and precision rates. In particular, the tool was unable to identify some items listed in the gold standard. This issue is not easy to handle as conjunctions can appear between different structural elements of sentence generating several different readings. The problem of alternative interpretations of a phrase due to the use of conjunctions is known as *coordination ambiguity* (Resnik, 1999). Resolving such ambiguities has been recognized as one of the most difficult problems in NLP. Some authors proposed only to notify a requirements engineer of the presence of such ambiguities (Chantree et al., 2006). We propose two ways to address this problem:

in the short term, domain experts can be asked to avoid the use of conjunction constructions; in addition to this, we plan to extend the set of heuristic rules applied by Cerno in order to better deal with these situations. The application of these solutions will significantly increase the quality of rates. Indeed, the avoidance of conjunctions reduces the ambiguity in natural language scenario descriptions: the fewer conjunctions are used, the less is the probability of introducing structural ambiguities (Kamsties et al., 2001). On the other hand, the use of more sophisticated heuristics can help to annotate information at a finer level of granularity.

Another difficulty for the tool is a classical problem in NLP: incomplete specifications due to unresolved pronoun coreferences (Hirst, 1981). Consider, for instance, the following statements: "<actor>The bank</actor> is the largest bank in ... <Capability>. It provides a wide range of services: ... </Capability>", where "it" (in the second statement) refers to the bank. Though the requirement statement is identified and annotated as "capability" because of the verb pattern "provides", the tool is not able to associate the pronoun to the bank. This drawback requires revising the document by making explicit the subject of every statement. As a partial solution, such problems can be avoided from the beginning by explicitly asking stakeholders to avoid using pronouns in writing requirements specifications. Alternatively, the RCS can be filled with incomplete knowledge and analysts are required to specify missing information. In this case heuristics can be defined to suggest, for instance, a candidate actor from the previous sentence in the text.

The other problem that normally makes difficult the automated construction of a model from textual requirements specifications is multiple identities of entities. For instance, in the e-Business scenario "Paul" and "the post-processing clerk" are the same person.[3] This issue can be handled relatively easily by Cerno through total substitution of all occurrences of multiple names for each entity by a single label. As we discussed earlier, all terms and names are defined in a short dictionary and used consistently throughout each scenario description.

The analysis has also pointed out discordance between assumptions made by human analysts and by the tool. This mismatch should be appropriately coordinated in future. For instance, the evaluation of results has shown that the precision rate is quite low in some scenarios due to the ambiguity in the specifications of delegation and trust relationships. The explanation is partially the lack of indicators that clearly distinguish between their types, but not only. Sometimes, generic trust statements were interpreted by human analysts (especially in the Smart Item scenario) as both trust of execution and trust of permission. In contrast, this assumption was not valid in the tool since it is not true in general. As consequence, the accuracy rate was lower than expected.

The accuracy rate has also suffered because of missing information. However, one should notice that the final RCS, which has been used as a gold standard, was not directly derived from the informal scenario description, but it is the outcome of iterative revisions of requirements specifications on the basis of the knowledge of domain experts. Actually, domain experts found it more convenient to revise, first, the

[3] For the sake of simplicity, it was assumed in the project that each role is played by exactly one agent.

RCS and, then, graphical requirements models rather than providing a new informal scenario description. Therefore, some parts present in the informal description of the scenario can be more detailed in the RCS; also, other parts can be omitted in the schema since domain experts found out that those parts were not relevant for the purpose of the project. This problem was particularly evident, for instance, in the e-Government and Smart Item scenarios, which have been considerably revised since the first draft.

The evaluation of productivity has shown that the tool guarantees substantial time gains for the requirements elicitation process. This result becomes even more significant if we take into account that the identified syntactic indicators for SI* concepts and, in particular, complex relationship patterns can be reused in future projects.

Certainly, human involvement in the requirements elicitation process cannot be completely substituted by the tool. Rather, the tool is intended to support analysts in the analysis of scenario descriptions provided in natural language by collecting and organizing information relevant to an application domain in a semi-structured format. This information still needs to be manually revised by domain experts by adding or removing information according to the purpose and the level of detail required by the project as well as to solve ambiguities in the textual documentation. The recovery time depends on the quality of results. Table II shows that the tool is able to annotate most of relevant information and discard irrelevant one. Thereby, we expect that the effort necessary to recover information would be limited.

7. Related Work

The problem of requirements elicitation is well-known in Requirements Engineering. Several approaches have been proposed to cope with this issue, especially in agent-oriented and goal-oriented communities. For instance, the Tropos methodology (Bresciani et al., 2004) provides a graphical environment to capture requirements of a system and the environment where it will operate to ease designers and stakeholders to understand each other. Another preeminent proposal is KAOS (van Lamsweerde et al., 1991). The aim of KAOS is to provide a constructive assistance during requirements engineering activities, from the elicitation of the objectives of the system and its integration into the environment, to the formal definition of the specifications of the system. In particular, van Lamsweerde and Willemet refined the previous work by presenting a systematic method for identifying system goals and requirements from informal interaction scenarios (van Lamsweerde and Willemet, 1998). This method generates a set of goal specifications in temporal logic. However, these approaches do not provide facilities for extracting requirements from documents specified in natural language. The analysis of the domain and deriving abstractions of modeling primitives is completely manual.

Tools based on text analysis for conceptual modeling from informal requirements specifications have been already proposed in a large number of studies. In particular, the large effort has been made in the natural language processing field since the early '80s. The document translator by Cordes and Carver (Cordes and Carver, 1992) was one of the first attempts to elaborate a systematic approach for deriving

object-based design from natural language requirements. The work of Goldin and Berry (Goldin and Berry, 1997) interprets the problem of finding general abstractions in requirements specifications as a problem of finding common sub-fragments among a set of sentences, thus reducing it to the traditional signal processing problem. Their method is supported by a prototype tool, AbstFinder, which find abstractions in natural language text to be used in requirements elicitation. Similarly to our approach, AbstFinder is lightweight as it does not involve any additional knowledge bases or annotated training data. However, we are interested in identifying specific terms in textual requirements specifications rather than finding general abstractions.

Numerous works suggested employing complex linguistic processing to facilitate requirements specification and verification. Among such tools are, for instance, COLOR-X (Burg, 1996) and NL-OOPS (Mich and Garigliano, 2002). Recently, Ambriola and Gervasi proposed Cooperative Interactive Requirements-Centered Environment (Circe), a modular expert system developed to support the modeling and analysis of requirements expressed in natural language (Ambriola and Gervasi, 2006). Circe applies a series of successive transformations to the input specifications in order to obtain requirements models. However, full natural language processing is a large problem area still containing several open research issues. In contrast, our work approaches the requirements elicitation process in a lightweight manner by relying on domain specific indicators retrieved, for example, from some document samples. Thereby, an initial human effort is required to tune Cerno's domain-dependent components. However, the result of this effort can

be reused during the application of the elicitation process to other projects.

Another approach to facilitate requirements acquisition is reformulating requirements in restricted natural language (Fuchs et al., 1998). However, we do not consider such an approach in this work as our initial goal was to reuse existing knowledge in the form of informal text descriptions and scenarios for requirements elicitation. Kaiya and Saeki (Kaiya and Saeki, 2005) proposed a requirements analysis method based on domain ontologies and a thesaurus. Although this method does not support full natural language processing, it allows the detection of incompleteness and inconsistency in requirements specifications, assessment of the quality of the document, and prediction of requirements changes.

Several works in aspect-oriented requirements engineering have been approaching the task of identification and separation of concerns using text analysis methods. To assist developers in identifying aspects from software requirements documentation, the Theme methodology (Baniassad and Clarke, 2004) provides facilities for the visualization and analysis of requirements and the modeling of aspects in UML. Aspects are identified using action views that show how actions are related to each other. Developers have to identify a list of key actions by looking at the document and picking out sensible verbs. The framework then performs lexical analysis of the document and generates action views. Differently from our approach, key actions are specific to a certain document and cannot be reused for the analysis of other requirements documents. The EA-Miner tool (Sampaio et al., 2005) supports separation of aspectual and non-aspectual concerns and their relationships

by applying natural language processing techniques to requirements documents. The tool reduces human effort in identifying abstractions such as base concerns, early aspects, and crosscutting relationships. It applies part-of-speech and semantic tagging to markup the elements of input documents. The result is represented as Java objects and can then be refined by human analysts. The refined model can be translated in XML or DOC formats. Along similar lines, in (Cole et al., 2006) the authors suggest a linguistic approach to extract explicit causal relations from text. This tool identifies subject-verb-object triples based on part-of-speech and syntactic parse of sentences. Then, it applies several heuristic rules to determine the triples that are causal relations. This tool, however, is intended to be a block of a more complex system rather than a stand-alone application. Instead, our approach does not employ linguistic tools for document analysis and is based on a set of domain- and context-driven indicators that point out to the relevant requirements abstractions. Stone and Sawyer (Stone and Sawyer, 2006b; Stone and Sawyer, 2006a) applied lightweight NLP techniques to address the problem of identifying the so-called tacit knowledge-based requirements, i.e., requirements derived from implicit built-in knowledge about a problematic domain. In particular, they applied Latent Semantic Analysis (LSA) techniques to compare requirements specifications with source documents from which the requirements are derived. Requirements that do not have a counterpart in the documents are classified as tacit or poorly specified knowledge. Differently from this work, our objective is the extraction of relevant facts from textual scenario description, rather than tacit requirements.

Our work relates well to the proposal by Breaux et al. (Breaux et al., 2006) for extracting rights and obligations from legislation documents and, in particular, from HIPAA privacy rules. As the first step of regulation analysis, it is required to identify phrases containing information on rights and obligations. Their solution uses normative phrases, i.e., contextual indicators derived from the analysis of the regulation. Similarly, our approach is based on the assumption that SI* concepts can be reliably extracted using lightweight contextual patterns. The difference lies in the degree of automation. The authors provided a systematic way to detect fragments describing rights or obligations, however all the work is left to humans. Recently, a computer-aids support has been presented in (Kiyavitskaya et al., 2007a), where the authors used Cerno for annotating rights and obligations in the HIPAA regulation.

8. Conclusions and Future Work

The challenge addressed in this paper is the tool supported extraction of relevant facts from natural language descriptions to feed SI* models. To cope with this issue, we have used Cerno, a tool based on software code analysis and markup techniques, to support analysts during the phase of generating requirements models. The tool identifies and annotates instances of SI* concepts in the text, thus providing a useful information for drawing requirements models.

To evaluate the tool, we have applied it to four industrial case studies. The experience gained during this work has also pointed out how different stakeholders can describe similar situations using different

natural language constructions. Thereby, our future work will include additional investigation devoted to the tuning phase in order to improve the results of the annotation phase. This entails the analysis of new case studies focusing on the identification of the contextual indicators for SI* concepts in requirements specifications.

Another research problem we plan to tackle is the automation of the second step of the model generation process. In particular, we are implementing a tool for the automatic translation of semi-structured tabular data into SI* models and vice versa. From one side, this will allow us to reduce the time and efforts needed to produce a first draft of requirements models. From the other side, this will allow us to maintain the consistency among different representations of requirements specifications.

Finally, we are investigating the relationships between the notions of right and obligation proposed in (Breaux et al., 2006) and SI* concepts. The objective of this work is to facilitate extraction of constraints imposed by the legislation and represent them in a framework where formal tools are available for model checking.

Acknowledgements

This work has been partially funded by EU Commission, through the SERENITY project and by the FIRB program of MIUR under the TOCAI.IT project.

References

Ambriola, V. and V. Gervasi: 2006, 'On the systematic analysis of natural language requirements with Circe'. **13**(1), 107–167.

Asnar, Y., R. Bonato, V. Bryl, L. Compagna, K. Dolinar, P. Giorgini, S. Holtmanns, T. Klobucar, P. Lanzi, J. Latanicki, F. Massacci, V. Meduri, J. Porekar, C. Riccucci, A. Saidane, M. Seguran, A. Yautsiukhin, and N. Zannone: 2006, 'Security and privacy requirements at organizational level'. Research report A1.D2.1, SERENITY consortium.

Baniassad, E. and S. Clarke: 2004, 'Theme: An Approach for Aspect-Oriented Analysis and Design'. In: *Proc. of the Int. Conf. on Software Engineering.*

Breaux, T. D., M. W. Vail, and A. I. Antón: 2006, 'Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations'. In: *Proc. of RE'06.* pp. 46–55, IEEE Press.

Bresciani, P., P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini: 2004, 'TROPOS: An Agent-Oriented Software Development Methodology'. *JAAMAS* **8**(3), 203–236.

Burg, J. F. M.: 1996, *Linguistic Instruments in Requirements Engineering.* IOS Press.

Campadello, S., L. Compagna, D. Gidoin, P. Giorgini, S. Holtmanns, J. Latanicki, V. Meduri, J.-C. Pazzaglia, M. Seguran, R. Thomas, and N. Zannone: 2006a, 'S&D Requirements Specification'. Research report A7.D2.1, SERENITY consortium.

Campadello, S., L. Compagna, D. Gidoin, S. Holtmanns, V. Meduri, J.-C. Pazzaglia, M. Seguran, and R. Thomas: 2006b, 'Scenario selection and definition'. Research report A7.D1.1, SERENITY consortium.

Chantree, F., B. Nuseibeh, A. de Roeck, and A. Willis: 2006, 'Identifying Nocuous Ambiguities in Natural Language Requirements'. In: *Proc. of RE'06.* pp. 59–68, IEEE Press.

Cole, S. V., M. D. Royal, M. G. Valtorta, M. N. Huhns, and J. B. Bowles: 2006, 'A Lightweight Tool for Automatically Extracting Causal Relationships from Text'. In: *Proc. of IEEE Southeastcon 2006.* pp. 125–129.

Cordes, D. W. and D. L. Carver: 1992, 'An Object-Based Requirements Modeling Method'. *Journal of the American Society for Information Science* **43**(1), 62–71.

Cordy, J. R.: 2006, 'The TXL source transformation language.'. *Sci. Comput. Program.* **61**(3), 190–210.

Fuchs, N. E., U. Schwertel, and R. Schwitter: 1998, 'Attempto Controlled English - Not Just Another Logic Specification Language'. In: *Proc. of LOPSTR'98*, Vol. 1559 of *LNCS*. pp. 1–20, Springer-Verlag.

Giorgini, P., F. Massacci, and N. Zannone: 2005, 'Security and Trust Requirements Engineering'. In: *FOSAD 2004/2005*, Vol. 3655 of *LNCS*. Springer-Verlag, pp. 237–272.

Goldin, L. and D. Berry: 1997, 'AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation'. **4**(4), 375–412.

Handschuh, S., S. Staab, and F. Ciravegna: 2002, 'S-CREAM – Semi-automatic CREAtion of Metadata'. In: *Proc. of EKAW'02*, Vol. 2473. pp. 358–372, Springer-Verlag.

Hirst, G.: 1981, *Anaphora in Natural Language Understanding: A Survey*, Vol. 119 of *LNCS*. Springer-Verlag.

IBM: 2007, 'IBM Rational RequisitePro'. http://www-306.ibm.com/software/awdtools/reqpro/. Accessed 21-Dec-2007.

Kaiya, H. and M. Saeki: 2005, 'Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach'. In: *Proc. of The 5th Int. Conf. on Quality Software*. pp. 223–230, IEEE Press.

Kamsties, E., D. Berry, and B. Paech: 2001, 'Detecting Ambiguities in Requirements Documents Using Inspections'. In: *Proc. of WISE'01*. pp. 68–80.

Kiyavitskaya, N., N. Zeni, L. Mich, T. D. Breaux, A. I. Anton, and J. Mylopoulos¡: 2007a, 'Extracting Rights and Obligations from Regulations: Towards a Tool-Supported Process'. In: *Proc. of ASE'07*. pp. 429–432, ACM Press.

Kiyavitskaya, N., N. Zeni, L. Mich, J. R. Cordy, and J. Mylopoulos: 2006, 'Text Mining Through Semi Automatic Semantic Annotation.'. In: *Proc. of PAKM'06*, Vol. 4333 of *LNCS*. pp. 143–154, Springer-Verlag.

Kiyavitskaya, N., N. Zeni, L. Mich, J. R. Cordy, and J. Mylopoulos: 2007b, 'Text Mining Through Semi Automatic Semantic Annotation'. In: *Proc. of ENTER 2007*.

Massacci, F., J. Mylopoulos, and N. Zannone: 2007, 'An Ontology for Secure Socio-Technical Systems'. In: *Handbook of Ontologies for Business Interaction*. The IDEA Group.

Massacci, F., M. Prest, and N. Zannone: 2005, 'Using a Security Requirements Engineering Methodology in Practice: The compliance with the Italian Data Protection Legislation'. *CSI* **27**(5), 445–455.

Massacci, F. and N. Zannone: 2008, 'Detecting Conflicts between Functional and Security Requirements with Secure Tropos: John Rusnak and the Allied Irish Bank'. In: *Social Modeling for Requirements Engineering*. MIT Press. To appear.

Mich, L. and R. Garigliano: 2002, 'NL-OOPS: A Requirements Analysis tool based on natural language Processing'. In: *Proc. of 3rd Int. Conf. on Data Mining*. pp. 321–330.

Nobata, C. and S. Sekine: 1999, 'Towards Automatic Acquisition of Patterns for Information Extraction'. In: *Proc. of Int. Conf. of Computer Processing of Oriental Languages*.

Ratchev, S. M., E. Urwin, D. Muller, K. S. Pawar, and I. Moulek: 2003, 'Knowledge based requirement engineering for one-of-a-kind complex systems.'. *Knowl.-Based Syst.* **16**(1), 1–5.

Resnik, P.: 1999, 'Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language'. *Journal of Artificial Intelligence Research* **11**, 95–130.

Robertson, S. and J. Robertson: 1999, *Mastering the requirements process*. ACM Press/Addison-Wesley Publishing Co.

Sampaio, A., R. Chitchyan, A. Rashid, and P. Rayson: 2005, 'EA-Miner: a tool for automating aspect-oriented requirements identification'. In: *Proc. of ASE'05*. pp. 352–355, ACM Press.

Sampson, G. and A. Babarczy: 2003, 'Limits to annotation precision'. In: *Proc. of EACL'03*.

Stone, A. and P. Sawyer: 2006a, 'Identifying tacit knowledge-based requirements'. *IEE Proceedings - Software* **153**(6), 211–218.

Stone, A. and P. Sawyer: 2006b, 'Using Pre-Requirements Tracing to Investigate Requirements Based on Tacit Knowledge'. In: *Proc. of ICSOFT '06*, Vol. 1. pp. 139–144, INSTICC Press.

van Lamsweerde, A., A. Dardenne, B. Delcourt, and F. Dubisy: 1991, 'The KAOS project: knowledge acquisition in automated specification of software'. In: *Proc. of the AAAI Spring Symposium Series*. pp. 59–62.

van Lamsweerde, A. and L. Willemet: 1998, 'Inferring Declarative Requirements Specifications from Operational Scenarios'. *TSE* **24**(12), 1089–1114.

Vargas-Vera, M., E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna: 2002, 'MnM: Ontology driven semi-automatic and automatic support for semantic markup'. In: *Proc. of EKAW'02*, Vol. 2473. pp. 379–391, Springer-Verlag.

Yang, Y.: 1999, 'An Evaluation of Statistical Approaches to Text Categorization'. *Inf. Retr.* **1**(1-2), 69–90.

Yu, E. S. K.: 1995, 'Modelling strategic relationships for process reengineering'. Ph.D. thesis, University of Toronto.

Zeni, N., N. Kiyavitskaya, J. R. Cordy, L. Mich, and J. Mylopoulos: 2008, 'Annotating Regulations Using Cerno: An Application to Italian Documents'. In: *Proc. of SREIS 2008*. pp. 1437–1442, IEEE Press.