# Analysis of Multilateral Software Confidentiality Requirements

by

Adeniyi Onabajo

B.Sc, University of Lagos, Nigeria, 1998
M.Sc, University of Victoria, 2003

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

# Analysis of Multilateral Software Confidentiality Requirements

by

Adeniyi Onabajo
B.Sc, University of Lagos, Nigeria, 1998
M.Sc, University of Victoria, 2003

**Supervisory Committee**

Dr. Jens Weber, Supervisor (Department of Computer Science)

Dr. Daniela Damian, Departmental Member (Department of Computer Science)

Dr. Daniel German, Departmental Member (Department of Computer Science)

Dr. Issa Traore, Outside Member (Department of Electrical and Computer Engineering)

**Supervisory Committee**

Dr. Jens Weber, Supervisor (Department of Computer Science)

Dr. Daniela Damian, Departmental Member (Department of Computer Science)

Dr. Daniel German, Departmental Member (Department of Computer Science)

Dr. Issa Traore, Outside Member (Department of Electrical and Computer Engineering)

## ABSTRACT

Ensuring privacy and confidentiality concerns of data owners is an important aspect of a secured information system. This is particularly important for integrated systems, which allow data exchange across organizations. Governments, regulatory bodies and organizations provide legislations, regulations and guidelines for information privacy and security to ensure proper data handling. These are usually specified in natural language formats, contain default requirements and exceptions, and are often ambiguous. In addition, interacting concerns, which are often multilayered and from different stakeholders, e.g., jurisdictions, need to be considered in software development.

Similar to other security concerns, analysis of confidentiality concerns should be integrated into the early phase of software development in order to facilitate early identification of defects - incompleteness and inconsistencies, in the requirements. This dissertation presents research conducted to develop a method to detect these defects using goal models which support defaults and exceptions. The goal models are derived from annotations of the natural language sources. A prototype tool is also developed to support the method.

The evaluations conducted indicate the method and tool provide benefits, including distinguishing requirement interferences and conflicts, exception handling, and navigation between annotated documents and the goal models.

Although current limitations of the method include a manual user driven annotation step, the method provides features that assist in early analysis of confidentiality requirements from natural language sources.

# Table of Contents

# List of Tables

# List of Figures

# List of Definitions

# *Acknowledgement*

I would like to express my sincere gratitude to my supervisor, Dr. Jens Weber, for his tremendous support and guidance during the research leading to this dissertation. His invaluable advices helped in shaping this research.

I thank members of my supervisory committee: Dr. Issa Traore, Dr. Daniela Damian and Dr. Daniel German, as well as Dr. Barbara Paech, the external examiner, for their insightful feedbacks.

I am grateful to Dr. Thomas Santen and Seda Gürses for their helpful comments at the early phase of the research. I appreciate the time and effort of Andrew McNair and Glen McCallum in providing comments on the dissertation, and thank members of the Netlab/PPCI/Symbioses research group for their assistance.

Finally, I would like to acknowledge the wonderful support of my friends and family.

# *Dedication*

To my parents and siblings

# Chapter 1

# Introduction

## 1.1 Motivation

Increasing occurrences of data misuse and confidentiality breaches emphasize the need to consider security concerns at the early phase of software system life-cycle. Negative impacts of insecure handling of confidential data include financial loss perpetrated through identity theft, social stigmatization from disclosure of personal data, e.g., health records, and threats to personal safety. The information age, facilitated by advancement in distributed computing and communication, is characterized by trends towards electronic data processing in various aspects of modern society. This includes online banking, e-services by government agencies, electronic medical systems, as well as social interactions over the internet. However, potential benefits, such as fast and effective service delivery, are overshadowed by concerns over data confidentiality.

Government legislations e.g., Personal Information Protection and Electronic Documents Act (PIPEDA) in Canada [6], Health Insurance Portability and Accountability Act (HIPAA) in the U.S [10] and Data Protection Act in the UK [17] and industry guidelines require organizations, which use personal information to adhere to principles of information privacy and security. By extension the information systems used by the data custodians should also conform to these principles. Government legislation represents a generic framework in which individual stakeholders or stakeholder groups can define more concrete privacy and security goals. For example, professional bodies such as colleges of physicians, establish standards of practice and conduct which address different issues in-

cluding the privacy of patient information. Interactions between the different stakeholder goals or with legal framework can lead to inconsistencies, which need to be harmonized for secure system development.

Stakeholders and software developers have realized the importance of systematic requirement analysis in the software development process and there are on-going efforts to integrate security requirement analysis early in the development life-cycle [96]. Confidentiality requirements analysis needs to be done at the early phase of development and continued throughout the system life-cycle.

Multilateral requirements engineering enables consideration of the viewpoints of different system stakeholders. It allows identification of the different and possibly competing requirements from the stakeholders involved, and assists in achieving a balanced set of requirements [108]. Appropriate formalisms and methods should support representation and analysis of the multilateral confidentiality requirements.

In complex applications, such as electronic medical records systems, the task of harmonizing the different requirements can be challenging. This is because of the inherent difficulty of identifying all confidentiality related stakeholder goals early on in the system life-cycle. These complex applications are open-ended in terms of: (1) breadth - new information is discovered or become relevant, (2) depth - finer-grained details are discovered or become relevant, and (3) complexity - new relationships are discovered or become relevant [109]. As a result, systems are often built with general confidentiality goals that represent the lowest common denominator of "future-proof" requirements. For example, clinical information systems typically just differentiate between two different uses of patient health data: (1) clinical use (also called primary use) and (2) research use (also called secondary use).

Requirements elicitation and analysis is typically performed in an iterative process of specification, evaluation and refinement. Each pass of the requirement phase provides an opportunity to consider new concerns from stakeholders or revisit existing requirements. Methods should be able to identify inconsistencies early so that these can be resolved or

marked as sources of inconsistencies. Confidentiality requirements can be subject to ongoing modifications because stakeholder requirements could change given certain scenarios. For example, systems can be designed to support different consent based models for specifying confidentiality requirements of the stakeholders [53].

The evolving nature of confidentiality requirements, where the set of requirements is incomplete at a given time can be supported by modelling the requirements as defaults, which can be refined with more specific requirements as additional information is available. For a given scenario, the more specific or overriding requirements hold, and in the absence of such requirements the defaults are observed.

Analysis which considers specificity relationships between multilateral requirements e.g., requirements from different jurisdictions, should be adopted in order to address the concerns of all stakeholders. These relationships often indicate prioritization among requirements.

## 1.2 Problem Exploration and Definition

The challenges of multilateral confidentiality requirements analysis can be observed in different domains. The following discussion illustrates these issues with requirements in the medical domain.

### 1.2.1 Confidentiality Requirements in Practice

The existence of multilayered stakeholder confidentiality concerns is a motivation for analysis based on requirement stratification. For example, privacy concerns in a national integrated health solution, such as the Canadian Health Infoway's [5] Electronic Health Record Infostructure (EHRi), are from different jurisdictional levels (federal, provincial and local authorities), and stakeholder groups thus making stratified analysis suitable.

Canada Health Infoway (Infoway) [5] was established in 2001 to facilitate the development and adoption of interoperable Electronic Health Record (EHR) solutions. As part of

its ongoing work it has developed an initial set of Security & Privacy requirements specification and an architecture blueprint for the pan-Canadian EHR [2]. The specification is in natural language format and careful analysis is required before implementation, particularly with regards to its interaction with requirements from other stakeholders.

An analysis of the specification with regards to other stakeholders requirements, such as the requirements from jurisdictions, clinical settings and individuals reveals interesting interactions.

i) The following requirement is specified in the Infoway requirement specification:
"*Except where inappropriate (e.g. specifically exempted by law or professional code of practice), organisations connecting to the EHRi, and organisations hosting components of the EHRi should obtain the knowledge and consent of each patient/person for the collection, use or disclosure of his or her PHI - and where required by law, must obtain the knowledge and consent of each patient/person for the collection, use or disclosure of his or her PHI.*"
Consent could be expressed, implied or deemed. Expressed consents include actions, which specifically authorize the collection, use or disclosure of personal information, e.g., a signature. Implied consents can be determined by action/inaction of individuals e.g., an individual presenting himself/herself to a physician in private practice. Deemed consent implies that the data custodian is permitted to act as if consent was given and there is no right to withhold or withdraw consent. However, these rights are available in an implied consent.
This requirement can be further refined by Canadian jurisdictions i.e. provinces, or patient stakeholder groups since it is a recommendation for obtaining consent for collection, use or disclosure of PHI (Personal Health Information). For example, the Ontario Personal Health Information Act (PHIA) requires having knowledgeable consent for collection, use and disclosure of PHI [13], while Alberta's Health Information Act (Section 58(2)) [1] requires that healthcare providers consider the expressed wishes of the patient, who is the subject of the information, in deciding on

disclosure of health information. A jurisdiction could also adopt deemed consent for certain scenarios, hence explicit knowledge is not obtained.

Medical facilities might require implied consent from patients to provide medical care. This scenario could occur in facilities where different clinicians are on-call, hence a patient is not guaranteed seeing the same care provider each time. In such situations, consent is desired for all the clinicians in order to provide care without hinderance.

Patients or groups of patients can also refine the requirement. Patients could opt that their PHI is not shared with third parties. However, refinements cannot be in violation of legal requirements which stipulate that healthcare practitioners have to record PHI or some part of it, e.g., Section 19(2) of the PHIA: "*If an individual places a condition on his or her consent to have a health information custodian collect, use or disclose personal health information about the individual, the condition is not effective to the extent that it purports to prohibit or restrict any recording of personal health information by a health information custodian that is required by law...*" [13]. Examples from the United States include statutes from the state of Oregon which require medical facilities to report cases of cancer to its Health Division, while in the state of Washington cancer diagnosis are reported to its cancer registry.

The above examples highlight the norm in specifying privacy and confidentiality requirements: requirements can be specified as general expected cases, i.e., defaults, which can be further refined or they could be irrefutable e.g., mandatory legal requirements.

ii) The purpose for which data is collected or used is important for privacy requirements. The Infoway specification includes the following requirements:

1) "*Organisations connected to the EHRi and organisations hosting components of EHRi must: a) identify all the purposes for which PHI will be collected, used and disclosed at or before the time it is collected and b) make a reasonable effort to inform patient/persons of these purposes, in a readily understandable*

*manner prior to collecting their PHI*"

2) "*Organisations connecting to the EHRi or organisations hosting components of the EHRi should only collect PHI necessary to fulfill the purposes that they have identified*".

These requirements are aimed at ensuring that genuine effort is made to ensure that patients/persons understand the purpose for data collection, usage and disclosure. Often only the main categories of the purposes are provided, and sub-categories are considered as they become available or relevant. It might also not be possible to identify the purpose at or before data is collected, e.g., in an emergency or during increased risk to public health, which can necessitate the use of medical information for previously unspecified purposes; an outbreak of a new contagious disease could trigger the need for urgent studies to control its spread thereby requiring data collection, usage and disclosure.

Refinement of requirements can be based on the purpose for which the information is to be collected, used or disclosed. The following was specified for a medical facility for fund raising purpose:

"*...We may disclose medical information to a foundation related to the hospital so that the foundation may contact you in raising money for the hospital. We only would release contact information, such as your name, address and phone number and the dates you received treatment or services at the hospital. Please write to us at ... if you wish to have your name removed from the list to receive fund-raising requests ...*"

This requirement can be refined for patients who do not want their contact information shared with a third-party for fund raising purpose.

iii) Patients can have restrictions on use or disclosure of sections of personal information or on who can have access to the information. These consent directives should be adhered to even in an integrated system such as the EHRi. The following is an example of restrictions on data and individuals who might have access:

"*You have the right to request a restriction or limitation on the medical information*

*we use or disclose about you for treatment, payment or health care operations. You also have the right to request limit on the medical information we disclose about you to someone who is involved in your care or payment of your care, like a family member or friend. For example, you could ask that we not disclose information about a surgery you had.*

*We are not required to agree to your request. If we do agree, we will comply to your request unless the information is needed to provide emergency treatment, or if the disclosure is required by law.*"

While patients can specify limitations on data use and disclosure, the medical facility may not observe these restrictions for emergency purposes or where it is legally mandatory to use or disclose information. These are examples of exceptions specified for requirements.

### 1.2.2 Observations

The following observations, which need to be considered during analysis of multilateral confidentiality requirements, are made from the scenarios described in Section 1.2.1:

i) confidentiality requirements can be specified as defaults, which could be refined for more specific scenarios.

ii) refinements of defaults can be based on the information entity or concerned stakeholders of the requirements.

iii) defaults can have exceptions, which are specified by stakeholders.

iv) defaults allow easy evolution of the requirements. The set of requirements is usually incomplete at a given time due to continuous consideration of new concerns from stakeholders.

The observations and discussions above can be used in defining a default requirement. A *default requirement* is a requirement which represents a typical or general case but can be overridden (have exceptions). A default requirement is usually specified with the use of

key words such as “Generally”, “Except ...”, “Unless ..”.

Existing requirements engineering methods support modelling stakeholder and requirement relationships, and performing inconsistency analysis to identify conflicts. However, there is limited support for representation and analysis of defaults. A lack of support for defaults can result in classification of exceptions as conflicts because the defaults are deemed as mandatory requirements which are contrary to the exceptions. For example, given the following requirement:

“*Except for physicians, clinicians should not access medical records of a patient’s family members*”. The requirement specifies a default that clinicians should not access the medical records of the family members of a patient. However, an exception is made for physicians. Clinicians comprise physicians, nurses, etc.

A method which does not support defaults and exceptions will conclude that the exception for physicians is a conflict because it contradicts the general requirements for clinicians. In order to avoid the classification of exceptions as conflicts of defaults, requirements engineering methods may alternatively represent defaults and exceptions by explicitly enumerating each specific requirement. Therefore, the default and the exceptions for the example are represented explicitly as the following:

i) Physicians can access medical records of a patient’s family members.

ii) Nurses should not access medical records of a patient’s family members.

iii) *A*s should not access medical records of a patient’s family members.

iv) *B*s should not access medical records of a patient’s family members.

...

n) *Z*s should not access medical records of a patient’s family members.

where A, B, ..., Z are also clinicians.

As previously indicated, requirement analysis is usually an iterative process where additional information is often identified (Section 1.1). In the example above, if additional stakeholder groups are identified as clinicians or existing stakeholder groups are deemed not to be clinicians, the list of explicitly enumerated requirements for each default has to be

updated. However, for a method with default representation, only the clinician stakeholder group needs to be updated. Therefore, supporting defaults provides more compact and more easily manageable representations [29]. Furthermore, representations with defaults provide a closer mapping to the original natural language specification.

Extraction of structured representations (models) from natural language specifications is needed because confidentiality requirements are usually specified in natural language formats [103]. Most requirement engineering methods with tool support create models with a visual editor without any link to the natural language source. This makes it difficult to trace the models to the actual sources. Providing support for navigating between the natural language and corresponding formal representations will assist the requirement analysis process by providing a means to trace models to sources. Extraction of models directly from the natural language sources helps in establishing traceability links which can be used to support navigation between models and original sources.

## 1.3 Research Goals

The observations in Section 1.2.2 motivate the research presented in this dissertation.

The goal of this research is to develop a method and tool to analyze multilateral confidentiality requirements with support for the following:

i) Representation of confidentiality requirements with defaults

   As indicated in Section 1.2.2 defaults are used in natural language requirements. A method which does not express defaults can be used to model defaults if all relevant requirements to the defaults (including exceptions) are known. All the relevant requirements are explicitly modelled and these makes mapping between the natural language requirements and the model representation cumbersome.

   However, not all relevant requirements are known at a given time. In a multilateral scenario, different stakeholder requirements are identified at different times. Modelling some requirements without complete knowledge of all other relevant require-

ments requires a method with support for defaults.

ii) Direct model extraction from natural language sources

Extraction of models directly from the natural language sources will help in establishing traceability links between the models and the natural language sources. The links can be used to provide navigation between models and original sources during and after model extraction.

iii) Identifying incompleteness and inconsistencies

Ambiguity or omissions in requirements specifications can have critical implications for the development process and the system. The incompleteness analysis is aimed at supporting the analyst by detecting omissions from the requirements representations. The identified omissions can then be clarified.

Similar to omissions, inconsistencies such as conflicts in requirements can result in unwanted system behaviour or vulnerabilities. Early detection of inconsistencies can prevent these outcomes, and it can save the time and cost for redesign and re-implementation.

iv) Providing a flexible exception handling technique which distinguishes between conflicts and exceptions

Defaults requirements can have exceptions. With support for expressed defaults, the method should distinguish between conflicts and exceptions. Exceptions override defaults, and are often more specific than the corresponding defaults. Establishing specificity relationships between requirements during analysis can be used to infer exceptions to default requirements. Conflicts are identified when such exceptions are not permitted.

## 1.4 Approach

The method developed to realize the goals of this research is called CREE - Confidentiality Requirement Elicitation and Engineering. CREE is aimed at providing support for analy-

sis of natural language confidentiality requirements. Analysis is performed with models, which are derived from annotations of natural language confidentiality requirements.

In addition, a tool called CREE-tool, is implemented to support the method.

Thus CREE provides the following:

i) Representations of confidentiality requirements, including default requirements, from natural language sources.

ii) Analysis to identify incompleteness.

iii) Analysis to identify inconsistencies, i.e., conflicts, based on specificity (stratified) requirement relationships and default requirements. The method distinguishes between conflicts, and allowed exceptions to default requirements.

### 1.4.1 Methodology

CREE is developed by exploring different methods for requirements engineering and security requirement engineering and how the methods support or do not support default requirements and model extraction from natural language specifications.

CREE is based on goal-oriented methods and it provides extensions to support default goal representation and creation of goal models from natural language sources. In order to extend the goal-oriented methods we conduct a qualitative study to identify elements for expressing the confidentiality requirements. The result of the qualitative study is integrated with concepts from goal-oriented methods to develop the CREE meta model. CREE goal models are derived by annotating the requirements sources with the elements of the meta model.

Formal definitions of concepts of the CREE meta model and a proof theory for analysis of goal models are specified. The definitions facilitate translation of goal models to logic program representation, which is used for analysis. The analysis is realized with a reasoning engine which supports default reasoning.

CREE-tool is used to create goal models through annotation of requirements sources

with the meta model concepts. The derived goal models are analyzed with the reasoning engine provided with CREE-tool.

### 1.4.2 Evaluation

We evaluate the CREE approach using two case studies and a user study (Chapter 5). The first case study evaluates the feasibility of using the CREE method to derive goal models through annotation of natural language sources, and the analysis of the goal models. This case study also explores the application of CREE in identifying interactions between two sets of requirements from different jurisdictions. The second case study evaluates CREE's inconsistency analysis with its support for defaults and exceptions. This case study is a comparative study with two other related approaches.

The user study was conducted to evaluate CREE-tool's annotation approach for extracting goal models from the natural language sources, in particular the ease of using CREE-tool's annotation, as well as its navigation and traceability features. The emphasis of the user study was to evaluate the ease of using CREE-tool's annotation technique to create goal models.

## 1.5 Contributions

This research contributes to the requirements engineering domain, in particular confidentiality requirements. The CREE method demonstrates the feasibility of using default requirements representation and default reasoning to analyze requirements. The method adopts goal-oriented requirements engineering concepts and provides extensions: default goals and specific goal types for information confidentiality concerns. The default goals and default reasoning enable analysis for detecting conflicts and exceptions in the requirements.

This research also contributes to the field of requirement model extractions from natural language sources. CREE uses semantic annotations to derive representations from

natural language confidentiality requirements sources. The annotation process creates goal model structures, which are used for the formal analysis. Elements from the extended goal concepts and their relationships are used as annotation elements.

CREE-tool was also developed to support the CREE method. CREE-tool supports goal model creation through annotation, and it provides formal analysis of the created models.

## 1.6 Organization of Dissertation

This dissertation is organized into the following chapters.

Chapter 2 (Foundation) of this dissertation provides background information on concepts essential to the research. These include requirements engineering, requirement stratification and default reasoning.

In Chapter 3 (Related Work), related methods are discussed with respect to support for defaults. Approaches for extracting formal representations from natural language requirements are also explored.

Chapter 4 (Confidentiality Requirements Elicitation and Engineering) describes the CREE method and CREE-tool. Details of the development of the CREE meta model for goal model creation, the formal definitions of CREE analysis, and the implementation of CREE-tool are discussed.

Chapter 5 (Evaluation) presents the evaluations performed to demonstrate the feasibility of CREE. The chapter describes findings, including limitations identified from the evaluations performed.

Chapter 6 (Conclusion) gives a summary of the research, and describes the research contributions. It also discusses current limitations, which could be the basis of future research work.

# Chapter 2

# Foundation

This chapter provides background information on topics in the areas of confidentiality, requirements engineering and default reasoning which are relevant to the method developed in this research.

## 2.1 Confidentiality

ISO standard *ISO/IEC 27002:2005 - Information technology - Security techniques - Code of practice for information security management* [11] defines confidentiality as "ensuring that information is accessible only to those authorized to have access". Confidentiality is also defined as the "concept of: 1) ensuring that information is accessible for reading, listening, recording or physical removal only to subjects entitled to it, and 2) that subjects only read or listen to the information to the extent permitted", where the subject may be a person, a process or an organization [78].

Confidentiality involves the concealment of information or resources from unauthorized persons or agents [37]. It is one of the key concepts which make up the traditional CIA (confidentiality, integrity and availability) triad of information security [116]. Confidentiality can be considered a prerequisite for privacy or assurance of data privacy and it denotes that only intended and authorized recipients, which could be individuals, processes or devices, should have access to the data. Unauthorized usage and disclosure is a confidentiality violation. Access control mechanisms can be used to support confidentiality. For example, cryptography can be used to scramble data in order to make it incomprehensible.

Increase in occurrence of confidentiality breaches, particularly with computing services have motivated governments and industries to provide regulations and guidelines for information protection. These require data custodians to implement proper data handling measures for personal information as well as provide recourse for data owners where there is confidentiality violation.

Confidentiality (privacy) regulations and guidelines serve as requirements for software systems. They are quality attributes, which need to be realized in addition to the functional requirements of the systems. The integration of confidentiality (privacy) regulations and concerns from individuals into software systems is a challenging task. A reason for this difficulty is that the regulations and concerns are usually specified in natural text and their interpretation can be ambiguous.

Security policies are used to define secure state of computing systems. "A security policy partitions the states of a system into a set of authorized or secure states and a set of unauthorized or non secure states" [37]. A security breach is said to occur if the system enters an unauthorized state. For confidentiality, the security policy identifies states where information leaks to unauthorized entities.

Confidentiality policies encompass information flow policies which are designed with the goal of preventing information from flowing to unauthorized users. Access control policies are used in information security infrastructure to specify which users have access to resources. Access control policies include Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role Based Access Control (RBAC).

DAC is an access control mechanism where access to an object is restricted based on the identity of subjects and/or groups they belong to. A user, usually the owner of the object, determines the object's access control policy by constraining which subjects can access it [37]. Commercial operating systems, e.g., traditional Unix and Windows use DAC.

MAC is an access control mechanism where access to an object is restricted by a system mechanism, and a user cannot override the access policy [37]. The system mechanism determines if access should be granted based on the security properties of the object and

the clearance of the subject. MAC is commonly used with military systems, where strict enforcement of the access control is critical.

The Bell-LaPadula Model is a formal model which describes access control using confidentiality classifications [34, 37]. It was developed for enforcing access control in government and military applications. A simple classification consists of a set of linearly (totally) ordered security clearances, which represent sensitivity levels, e.g., "Top Secret" (most sensitive), "Secret", "Confidential", "Unclassified" (least sensitive). A subject is assigned a security clearance and an object has a security classification from the set.

The simple classification of the Bell-LaPadula model is extended based on the "need-to-know" principle, where given a classified object, a subject has access only if there is appropriate clearance for some job function [37, 89]. The extension is realized by adding a set of categories to the security classification, e.g., NUC (Nuclear), NATO, US. The set of categories is partially ordered with the "$\subseteq$" relation. An element of the power set of categories can be assigned as the category of a subject or an object.

A security level $l$, is formed by a sensitivity level and a category set, e.g., (Secret, $\{$NATO$\}$), (Top Secret, $\{$NUC,US$\}$). Security levels can be partially ordered and hence form a lattice, based on relation "$\leq$", defined as follows: Let $c(l)$ be sensitivity level of $l$ and $C(l)$ be category set of $l$, then $l_1 \leq l_2$ if $c(l_1) \leq c(l_2)$ and $C(l_1) \subseteq C(l_2)$ [37, 89].

The Bell-LaPadula Model defines two properties for a state to be secure.

- Simple Security

  a subject S with security level $l_s$ has no read access to any object O with security level $l_o$, if $l_s < l_o$. This is commonly known as "no read up".

- *-Property

  a subject S with security level $l_s$ has no write access to any object O with security level $l_o$, if $l_s > l_o$. This is commonly known as "no write down".

Alternative views of confidentiality, such as notion of nondeductibility of information, may be more realistic in specific scenarios [37]. For example, if existence of an object needs to be concealed it is not sufficient to hide its content because a subject with low security level

can detect the existence of objects with higher security level when it is denied access.

RBAC is an access control approach where access to object is restricted to roles, which represent job functions in an organization [37, 36]. Permissions are granted to roles and the permissions are related to the objects required to perform the functions of the roles. Users are assigned to roles and acquire the roles' authorizations [36]. RBAC simplifies access control administration because authorizations are assigned by granting or revoking appropriate role membership to users. Unlike, DAC where access is specified for low level objects, RBAC permissions are function (operation) based. Role hierarchies, which reflect organizational structure, are supported in RBAC. Role hierarchies reduce the complexity of role and authorization specification through specialization of role definitions and authorizations.

RBAC has been incorporated in commercial database management systems as well as enterprise security administration systems [36]. SecureUML is a security modelling language which incorporates RBAC for access control policies (cf. Section 3.1.2). Similar to the RBAC approach, the CREE method developed in this research supports role and role membership concepts. It also supports the assignment of confidentiality objectives (which can be analogous to operations) to roles.

## 2.2 Requirements Engineering

Zave defines Requirements Engineering (RE) as follows: "Requirements Engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families." [133]. This definition emphasizes the point that real goals are the factors that drive the development of software systems. Specifying software behaviour precisely provides a means for defining the system to be built and verifying it, as well as analysis and validation of the system's behaviour. The definition alludes to the fact

that software systems change over time because real world goals and requirements change, therefore specifications will have to evolve as well [100].

Inadequate, incomplete and ambiguous requirements have critical implications not only on the quality of the software developed [35] but also on the cost [41]. Corrections due to errors in requirements result in enormous financial and time cost. Safety related errors in space programs (e.g., NASA's Voyager and Galileo programs) have also been primarily attributed to errors in functional and interface requirements [93, 123]. Surveys of many US and European companies have shown that failures of many software development projects can be blamed on requirement specification and management (incompleteness, imprecise objectives, unrealistic expectation, lack of user involvement) [123]. Iteratively eliciting and refining requirements is one of the most important processes in the software development life-cycle because it helps the developer(s) understand what is required in precise terms [46].

Requirements Engineering process adopts techniques and methods from many disciplines, including [100]:

- Computer science for assessing the feasibility of proposed system and tools for the development.
- Systems engineering for management of development life cycles.
- Cognitive and social sciences, which provide methods for requirements elicitation and modelling. Some of these are methods for understanding stakeholders' needs, conducting observations and analyzing communication patterns.

The process of RE consists of the following steps:

i) Elicitation: This is the process of identifying the system requirements. The requirements are subjected to analysis, modelling and validation to ensure that they are appropriate and comprehensive. Elicitation involves identifying stakeholders and their varying needs, system goals and how these relate to the stakeholders. The elicitation process assists in drawing attention to the specifics of the problem and defining

particular solutions. Various techniques are used in the elicitation process.

(a) Traditional methods: includes data collection methods such as questionnaire, surveys, interviews and existing documentation.

(b) Group elicitation: involves the various system stakeholders and helps to promote agreement on needs. This method includes focus groups and brainstorming sessions.

(c) Model-driven elicitation: employs a model based on the type of information required. Examples are goal-based approaches KAOS [126], i*[52] and NFR Framework [52, 71].

(d) Prototyping: This techniques is useful when an early feedback from the stakeholders is needed. Feedback from the prototype is used in further refinement of the elicited requirements.

Elicitation methods also include cognitive approaches used in knowledge-based systems, e.g., protocol analysis, in which an observer gets information about cognitive processes of accomplishing tasks through an expert who expresses his/her thoughts audibly. Contextual techniques have also been used and they include ethnographic methods such as participant observation [80] and techniques that analyze conversations and interactions for pattern recognition.

ii) Modelling and analysis: Modelling requirements provides representations that can be used in requirement analysis and further elicitation. Modelling approaches can be enterprise, data, behaviour or domain centric. Enterprise modelling focuses on the enterprise's structure and operations in terms of the goals, processes and tasks, and data processed [100, 131]. Data modelling techniques provide representations, which help to capture the data the systems represent, process and manage. Behavioural modelling is aimed at representing dynamic and functional behaviour of systems and stakeholders. Domain modelling provides abstract description of the world the system will operate in, and it allows reasoning on what is available in the defined

domain.

iii) Communication: RE also entails providing effective methods for sharing the requirements among the stakeholders. This implies having good documentation for the elicited requirements. Various specification languages and notations that range from natural to formal languages have been developed for documentation [62].

iv) Negotiation and agreement: This step ensures that the elicited requirements conform to the stakeholders' goals, that is, validation of defined requirements with stakeholders. Validation can be complicated by diverging or conflicting stakeholder goals. One approach for resolving conflicts among stakeholders' goals is requirements negotiation. Negotiation methods include attempts to identify common goals [111] or identify for each stakeholder, specific conditions, which are satisfied through negotiation (Win-Win Approach) [38, 39, 40].

v) Evolution: When application modifications are needed to address changes in the environment or functionality, new requirements will have to be specified. Therefore, there is a need to have a defined process to evolve and manage the changes in requirements as well. Requirement management aims to provide methods for requirements documentation and evolution [100].

Standards have been developed to assist the process of defining requirement specifications, e.g., the IEEE Recommended Practice for Software Requirements Specifications (IEEE Std. 830-1998), which specifies approaches for writing good software requirement specifications (SRS). The IEEE Guide for Developing System Requirements Specifications (IEEE Std. 1233-1998) provides guidance on capturing system requirements by helping analyst to have clear definitions of well-formed requirements and the appropriate methods of organizing the requirements. These two guides provide details of identifying requirements and complement IEEE Std. 12207.0-1996, which is the IEEE adaptation of the ISO/IEC Standard for Information Technology - Software Life-Cycle Processes.

### 2.2.1 Non-Functional Requirements

Non-functional requirements (NFRs) in software system engineering describe "qualities" of system functions, in contrast functional requirements specify "what" the system will do [52]. Non-functional requirements are software quality attributes, which are aimed at improving the system. These attributes include, accuracy, performance, reliability and security. They have also been termed "non-behavioural requirements" or "software constraints" [112, 52]. Sources of non-functional requirements include, environmental or system constraints (e.g., operating infrastructure or existing process) and user objectives, values and concerns.

Security requirements are traditionally considered as non-functional or quality requirements [52]. They are defined as system function constraints, which operationalize security goals [77]. Security requirements are often expressed as system functions by describing them with mechanisms to be used or in terms of a practice. For example, in ISO/IEC 15408 - Common Criteria for Information Technology Security Evaluation (commonly known as Common Criteria or CC), security requirements are specified as Security Functional Requirements (SFRs), which define rules by which a product (Target of Evaluation, TOE), e.g., software or hardware, "governs access to and use of its resources, and thus information and services controlled by the" product [1]. A criticism of these approaches is that they describe how systems are to be protected rather than why and when they should be protected [77].

Security requirements should be considered during the early phase of the development process in order to identify possible threats [96, 125]. This will help in analyzing the scope of protection needed and evaluating different design decisions. Unfortunately, information security needs are often not considered during initial requirements and the security mechanisms are added as ad-hoc features of the system, thus resulting in poor implementations that do not fulfill the desired requirements [96, 122]. This approach can be costly and

[1] CC Part 2:Security Functional Requirements, Ver.3.1, Rev. 2, CCMB-2007-09-002, Sept. 2007, p.18

risky because security requirements usually have a system-wide scope, and major modifications will be necessary. Temporary fixes can also have long-term negative impacts on the maintainability of the system [58]. Another concern of late consideration of security requirements is that most of the requirements decisions about security constraints of system functionalities are left to the discretion of developers, and this might be inadequate or inappropriate. Design decisions on security implementations should be based on clear and unambiguous specifications of the constraints on system functions. Flawed implementations based on assumptions from ambiguous security requirements can have adverse implications such as security vulnerabilities.

NFRs can be "treated" formally using two basic approaches: product-oriented and process-oriented. In the product-oriented approach formal definitions of non-functional requirements are defined and the software system is evaluated to determine the degree by which it satisfies the requirement [87]. This approach checks for software quality conformance by analyzing the final product. Process-oriented approaches are incorporated into the software development process. Unlike the product-oriented approach, the focus is on ensuring that NFRs are key factors in making software design decisions [97].

In addition, the methods adopted could be quantitative or qualitative. In general, the product-oriented techniques are quantitative, because analysis is based on numeric metrics to determine the level of satisfaction of an NFR. Process-oriented approaches can be quantitative or qualitative, however, the lack of a finished software product makes quantitative evaluation difficult.

### 2.2.2 Modelling Non-Functional Requirements

While modelling techniques have been identified as a method to elicit requirements, they can also provide additional functions in the requirement engineering process. They serve as a means of representing and organizing information from other elicitation methods, communicating as well as identifying and reasoning about inconsistencies of the requirements [80]. Methods for modelling requirements range from informal to formal methods and pro-

vide different levels of precision, which make them applicable to different analysis techniques.

Informal methods usually employ visual notation and textual language to describe and specify requirements. They are suitable for eliciting user requirements and for communication between analysts and users. However, they tend to be imprecise and ambiguous because different stakeholders can have varying interpretations of the informal specification. Formal methods are based on mathematical concepts (e.g., formal logic) and use formal notations for modelling requirements. Formal methods provide precise and concise specifications, and by using mathematical procedures, syntactic correctness and consistency of specifications can be checked [62]. Although specifications using formal methods are usually difficult to define, they can be used for automated reasoning and analysis of requirements. On the other hand, soft/informal methods provide representations that non-technical stakeholders will find usable, although these will be difficult to analyze automatically [100].

It should be noted that a formal specification is meaningless if there is no specific informal definition of how to interpret it in a domain of interest. This is done through a mapping of functions/predicates in the formal method to functions/predicates on domain objects [127]. This ensures the formal method is grounded in the domain of concern.

The decision to adapt a formal approach for modelling requirements should be made while noting factors that limit the adoption of formal methods in software engineering. The issues include (i) many developers are not familiar with the methods and (ii) minimal or lack of adequate tool support. Also of concern is the scalability of the formal approaches to large scale complex problems [92].

Successes in applications of formal methods in software development show a trend for tool support with the following features [92]:

- a specific and well-defined domain.
- a user-friendly interface.
- encapsulated formal methods concepts and/or algorithms e.g., theorem prover that is shielded from the users.

Thus, formal models and algorithms are used in creating tools (such as automated decision support), which assist in solving problems focused on a particular and well defined area. The tools provide support for human capability, which currently plays a vital role in the development process [92].

## 2.3 Multilateral Confidentiality Requirements

The need for multiple viewpoints in requirements engineering has been highlighted by different research [88, 117]. This is particularly important for complex and large-scale applications with different stakeholders who have multi layered concerns. Relevant stakeholders need to be identified in modelling security requirements because it facilitates analysis of the potential vulnerabilities of the proposed system by considering threats posed from the stakeholders' interactions with the system [90].

For confidentiality requirements the stakeholders with potential access to data might be of interest for the set of confidentiality goals. Therefore, confidentiality requirements engineering needs to be addressed from multilateral view points for practical applications [75]. For example, we often can distinguish between "data owners" and "data custodians", where the latter is entrusted with data belonging to the former in order to fulfill specific functions. Due to potential misuse, data owners have confidentiality goals of restricting data access to certain individuals or for some purposes. Furthermore, parties which might not be directly involved in the data collection and its routine usage often have interests in the data and its usage. For example, governments and their agencies provide guidelines for data usage by organizations within their jurisdiction. In addition, in order to facilitate public safety, access to personal data might override individual data usage directives, e.g., in medical epidemic outbreaks or law enforcement investigations.

Addressing confidentiality from the perspectives of different stakeholders requires identifying the specific data (or parts of data), the individual(s) or agencies who are the targets of the confidentiality concern, purposes for which the confidentiality concern is associated

and how long these concern is for. However, complete knowledge of all the goals of a stakeholder or of all relevant stakeholders might not be available at any particular iteration of the requirements engineering phase. Hence, the confidentiality requirements are subject to reviews during iterations of requirements engineering.

Stakeholder relationships are also of interest for confidentiality requirements because the goals of the requirements are to control access to personal data not only to attackers, but also to other stakeholders who do not necessarily have malicious intents. These relationships should be explicitly modelled for requirement specification. Non-permanent relationships among stakeholders, e.g., friendships, colleagues and family physician may also be the subjects of confidentiality requirements.

## 2.4 Requirements Stratification

Key concepts of a confidentiality goal include: (1) the stakeholder who has the goal, e.g., patients can have a goal to protect their health records from employers and other employees (2) the actual data object to be protected from misuse (3) the target stakeholder who is granted or denied access to the data, and (4) the purpose for which access is granted or denied. Use case analysis can be used to identify the stakeholders, data, target stakeholders and purposes of confidentiality goals. The data and the stakeholders from these scenarios might be of concern to a stakeholder. However, some stakeholders will not be involved in the use cases, these are indirect stakeholders e.g., governments.

Requirements stratification is the layering of requirements based on the specificity or hierarchical relationships between requirements. For confidentiality requirements, the hierarchical relationships can be defined using the confidentiality goal concepts, i.e., stakeholder, data object etc.

Requirements stratification can be used to determine precedence between requirements. Inferring hierarchical relationships between requirements in regulations helps to determine overlapping, overriding or conflicting requirements [103]. Requirements stratification oc-

curs when stakeholders have multi-layered confidentiality concerns, e.g., privacy concerns in a national integrated health solution, such as the Canadian Health Infoway's Electronic Health Record Infostructure (EHRi), involve many stakeholders at different levels of jurisdiction or different stakeholders affected by the concerns [5]. Stratification could also be due to the data entities. A more specific requirement could be specified for constituting elements of a data element.

Default requirements are requirements which are generally applicable but can be overridden. The overriding requirements are exceptions and are usually more specific compared to the defaults. Hence exceptions have a stratification relationship with defaults for a particular scenario. An exception for the example above could be that patients allow medical staff access to their health record. Therefore, analysis with stratification relationships can be used to determine conflicts and exceptions. Analysis of default requirements with stratified relationships can be supported with non-monotonic representation and reasoning, which supports overruling of a conclusion given contrary evidence [29].

Requirements analysis is an iterative process of eliciting and analyzing stakeholder concerns [80]. It allows refinement or review of existing requirements or elicitation of previously unknown requirements. Each phase could provide knowledge that reveal new and relevant data elements or stakeholders [109]. Therefore, requirements from later iterations might be exceptions to default requirements. Exceptions can be attributed to specific data elements or specific stakeholder(s). For example, patient stakeholders could grant consent to clinicians, which comprise physicians and nurses, for their medical records but an exception to this requirement could be to deny nurses access to family medical history.

## 2.5 Inference and Reasoning

Inference is a process of matching current data from a domain space to the existing knowledge in a knowledge-based system and inferring new facts until a solution in the solution space is reached [86]; it can be described as a process of drawing conclusion based on what

is known. Inference is applied in many areas of study, e.g., human inference is studied in psychology and inference from quantitative data in statistics.

Reasoning is the act of using logical inference to derive a conclusion from certain premises. A reasoning method which supports inference with defaults and exceptions is required for CREE's analysis.

Reasoning methods include:

- Deductive reasoning - this is a reasoning method in which given true premises, the conclusion must follow (the conclusion cannot be false). Deductive reasoning is non-ampliative, that is, it does not add to the existing knowledge base as the conclusion is self-contained in the premises. Syllogisms, such as the following:
  *All humans are mortal*
  *Socrates is a man*
  *Therefore, Socrates is mortal*
  are examples of deductive reasoning.
- Inductive reasoning- this is a reasoning method in which the premises of an argument support the conclusion, but do not ensure or guarantee it. It involves formulating laws based on limited observations of recurring patterns. Therefore, the conclusions follow with some degree of certainty. It is ampliative, as it provides further information than what was contained in the premises. An example of this method of reasoning is inductive generalization, which proceeds from a premise about a sample to a conclusion about the population:
  *A proportion Q of the sample has attribute A.*
  *Conclusion: Q of the population has A.*
  Bayesian inference, which uses probability theory as its framework for induction [22] is also a type of inductive reasoning.
- Abductive reasoning - this is a reasoning method in which the most plausible explanation for observed facts provided in its premises is inferred [42]. Hence, the antecedent, $a$, of "$a$ entails $b$" is inferred from the consequence $b$.

Reasoning methods in knowledge engineering can also be categorized as [86]:

- monotonic and non-monotonic

  An inference method is described as monotonic if new facts extend or at least do not contradict the current knowledge. In a non-monotonic inference method current knowledge might be reduced, that is inferred knowledge might be revised or retracted if no longer valid [66].

  Non-monotonic inference is applicable to default rules, which occur with inheritance hierarchies and exceptions. In the absence of contrary information, the default properties are assumed. For example, if Tweety is known to be a bird then for the statement "birds fly", it can be concluded that Tweety flies, since birds generally fly. However, Tweety could be a penguin or an ostrich and would not fly [66].

- exact and approximate

  In exact reasoning, exact solutions are produced from data, e.g., true/false or yes/no. Approximate reasoning produces solutions with degrees of approximation or applicability, e.g., classification of an object with a degree of 0.75 to a set and 0.2 to another set. Approximate reasoning is also applied in linguistic approximations (e.g., fuzzy logic, which is used to describe and reason about vagueness in concepts [86]) and reasoning about uncertainty and incomplete knowledge (e.g., possibilistic logic [61]).

As inferences can be valid or invalid, rules are defined for proper inference process, such that when applied to true or valid premises, correct conclusions can be made. Logic studies laws of valid inference and modern mathematical logic applies mathematical techniques to the representation and analysis of formal logic. Logic systems consist of basic symbols (or alphabet) for defining more complex sentences or expressions, a syntax or grammar for constructing the expressions, semantics for interpreting the system and inference rules for deriving valid inferences [86].

### 2.5.1 Default Reasoning

Non-monotonic reasoning methods support inference in which drawn conclusions can be invalidated when there is additional information or evidence [45]. Requirements engineering process is by nature non-monotonic because requirements are often elicited incrementally, with some requirements withdrawn during further review or at later iterations. Non-monotonic formalisms support revision of information and can be used in requirements engineering to address issues such as inconsistency management due to conflicting sources or concerns [29].

Non-monotonic inheritance assumes that for a body of knowledge that is organized taxonomically, subclasses inherit properties from their super classes. For example, if clinicians have access to health record then physicians also have access to health record because they are clinicians. However, there can be exceptions and this could result in complex interactions. For example, say, clinicians have access to field X of a health record, then physicians also have access to X. If there is an exception for intern physicians so that they do not have access to X, then there are potential conflicting inferences. However, interns are inferred not to have access to X based on specificity, i.e., the more specific information overrides the more generic information.

Defaults and exceptions often exist in requirements. Non-monotonic formalisms which support defaults would be suitable for managing requirements. Default reasoning methods use default rules to represent plausible conclusions or assumptions. The methods are non-monotonic because the defaults are retractable if additional contradictory evidence becomes available [29]. Two default reasoning approaches, Default Logic and Defeasible logic are described below.

**Default Logic** Reasoning often involves facts that are true in the majority of cases but not always. A common example is: *"birds typically fly"*. In standard logic this is expressed as *"all birds fly"*, which is inconsistent with the fact that penguins do not fly, or by *"all birds that are not penguins and not ostriches and ... fly"*, requiring

all exceptions to the rule to be specified. With default logic, inference rules can be formalized without explicitly specifying all the exceptions. Default logic was proposed by Ray Reiter and extends first-order logic with default rules.

A default theory consists of the pair $\langle D, F\rangle$. F is a set of logical formulae, which formalize the known facts. D is a set of default rules, each of which is of the form:

$$\frac{\gamma : \theta}{\tau} \quad where\ \gamma = prerequisite, \theta = justification, \tau = conclusion$$

$\gamma, \theta, \tau$ are sentences in a given language (e.g., first-order logic) and the default is interpreted as *"if $\gamma$ is known, and there is no evidence that $\theta$ might be false, then $\tau$ can be inferred"*. The prerequisite must be true in order for the rule to be applicable, while the justification has to be with currently known facts [107].

The rule for deciding when a conclusion of a default can be included to an argument is defined in terms of an *extension*. An extension of a default theory $\triangle = \langle D, F\rangle$, is the set of logical consequences of $S$, where $S = \bigcup_{i=0}^{\infty} S_i, S_0 = F$, and

$$\begin{aligned} S_{i+1} \quad = S_i \cup \{w(c) : \quad & \frac{\alpha(c) : \beta_1(c), ..., \beta_m(c)}{w(c)} \text{ is an instance of a default } D, \\ & \alpha(c) \text{ follows from } S_i, \\ & \beta_j(c) \text{ is consistent with } S, \forall j, 1 \leq j \leq m\} \end{aligned}$$

Therefore, an instance of the consequent of a default is used as premise of a logic argument if the instance of the prerequisite can be proven from facts and previously assumed consequents, and the justifications are consistent with the union of the consequents of the defaults [107].

The following is a representation of "birds fly, but baby birds are exceptional":

$$\begin{aligned} D = \quad & \{\frac{bird(x) \ : \ flies(x) \ \wedge \ \neg baby(x)}{flies(x) \ \wedge \ \neg baby(x)}\}, \\ F = \quad & \{bird(Tweety), baby(Polly), bird(Polly), \neg flies(Fred)\}. \end{aligned}$$

$flies(Tweety)$ and $\neg baby(Tweety)$ can be explained with $x = Tweety$ for the default. However, $flies(Polly)$ cannot be explained because $\neg baby(Polly)$ is inconsistent with $baby(Polly)$ in $F$.

**Defeasible Logic** Defeasible logic is a non-monotonic logic which supports both strict and defeasible rules, as well as priority relations on rules [94]. Defeasible reasoning aims at defining rules whose conclusions can be overturned by other rules. A defeasible theory D, is a triple (F, R, $>$), where F is a set of facts, R a finite set of rules, and $>$ an acyclic superiority relation on R.

A rule can be *strict* ($\rightarrow$, i.e., if the premise holds the conclusion is indisputable), *defeasible* ($\Rightarrow$, the conclusion could be defeated by contrary evidence) or *a defeater* ($\rightsquigarrow$, provide contrary evidence and thus used to prevent conclusions). For relation $r_1 > r_2$, $r_1$ is *superior* to $r_2$ and $r_2$ *inferior* to $r_1$ [94, 27].

The statement "Penguins are birds" can be stated formally as a strict rule:

$penguin(X) \rightarrow bird(X)$.

A defeasible rule can be used to formally state the statement, "birds typically fly":

$bird(X) \Rightarrow flies(X)$

and the indisputable statement, "penguins do not fly", is stated as

$penguin(X) \rightarrow \neg flies(X)$.

The proof theory of defeasible logic is summarized as follows [26]:

*A conclusion P can be derived if there is a rule whose conclusion is P, whose prerequisites (antecedents) are already proved or given (i.e., facts) and any stronger rule whose conclusion is in C(P) (P-complementary) has prerequisites that fail to be derived. Hence conclusion P is (defeasibly) derivable if:*

- P is a fact, or
- there is a strict or defeasible rule for P and either
  - all the rules for P-complementary literals are discarded or
  - every rule for a P-complementary literal is weaker than an applicable rule for P.

Inference from only strict rules is termed *definite inference*.

For the penguin example, although a penguin is a bird, it can be concluded that a penguin does not fly because this is a stronger statement. Superiority relations are

also used in the proof theory to make conclusions. For example, given the following defeasible rules

$r : bird(X) \Rightarrow flies(X)$

$r' : brokenWings(X) \Rightarrow \neg flies(X)$

with $r' > r$, it can be concluded that birds with broken wings cannot fly.

In addition to being both non-monotonic, defeasible logic and default logic distinguish statements that are definite (strict rules/facts) from statements with a lesser force (defeasible rules/defaults).

However, there exist substantial syntactic and semantic differences. In default logic there is no notion of priority among rules, nor of a defeater that cannot be used to make inferences. In addition, default logic permits any logical expression as a fact or in a default rule, but defeasible logic allows only rules as statements.

Another key difference, between the two logics is in their application of defaults: in default logic the applicability of a default is independent of any other default, thus given defaults:

$$\frac{: a}{a} \ and \ \frac{: \neg a}{\neg a}$$

either default may be applied while in defeasible logic with rules $\{\Rightarrow a, \Rightarrow \neg a\}$, neither rule may be applied, since each rule interferes with or defeats the other [101].

The intractable nature of many non-monotonic reasoning approaches has limited their practical applications, however, propositional defeasible logic is shown to have linear complexity [27].

Many data protection requirements are usually specified as regulations which organizations need to comply with (cf. Sections 1.1, 2.1). Commonly occurring in these regulations are hierarchies which result in overlapping, overriding or contradictory requirements (cf. Section 2.4). Representing and analyzing regulations with support for hierarchies, and identification of overlaps and exceptions remains a challenge in applying requirements engineering methods to regulations [103]. Defeasible logic with its defeasible rule and superiority relation can be used as a logical method for analyzing regulations [28]. Defeasible

logic has also been applied to the legal domain [59, 82].

#### 2.5.1.1 Deductive Validity of Defeasible Reasoning

Reasoning systems in predicate first-order logic are complete and sound, i.e., they produce all and only valid conclusions which follow from premises. In defeasible reasoning a proposition is considered as warranted if and only if it is undefeated relative to the reasoning so far performed. Defeasible reasoners are generally required to produce all and only warranted conclusions [106]. This requirement is considered analogous to soundness and completeness in deductive reasoning. However, the requirement is not satisfiable by defeasible reasoners which perform sophisticated reasoning.

A set is recursively enumerable if there is an algorithm which generates all and only members of the set. Deductive reasoners can draw all and only valid conclusions because the set of valid conclusions is recursively enumerable. However, for defeasible reasoning, the set of warranted propositions is not generally recursively enumerable. In order to defeasibly conclude $P$, a defeasible reasoning system using first-order language needs to show that $\neg P$ is not a theorem, otherwise $P$ is defeated. However the set of non theorems of a predicate calculus is not recursively enumerable, thus $P$ will not be concluded if $\neg P$ is not shown not to be a theorem.

Defeasible reasoning systems may encounter the constraints of either not affirming a warranted proposition (by waiting to determine that a conclusion is not defeated) or affirming a justified but unwarranted conclusion (by not waiting to determine that conclusion is defeated) [106]. Defeasible reasoning systems should draw conclusions based on the current reasoning and information, and as reasoning and/or new information becomes available earlier conclusions may be withdrawn.

## 2.6 Summary

CREE supports multilateral confidentiality requirements by providing representation and analysis of confidentiality concerns of different stakeholders. Stakeholders with the concerns and other stakeholders who might be of relevance for each concern are represented in the CREE model (cf. Section 4.1.2).

Defeasible logic is adopted for formal representation and analysis in CREE. Defeasible rules represent default requirements, and the properties for specifying a requirement, e.g., stakholders, are used to infer hierarchical relationships between the requirements. The hierarchical relationships determine the specificity of the requirements, and are used as the basis for CREE's defeasible superiority relation (cf. Section 4.4.3).

# Chapter 3

# Related Work

This chapter presents related requirements engineering methods, including those which support security requirements engineering, and explores how the methods support default requirements. In addition, approaches for extracting formal representations from natural language requirements are described.

The following sections describe these approaches and a summary is provided in Table 3.3.

## 3.1 Security Specifications with UML

A number of attempts have been made to integrate security requirements in general software engineering methods. Some of these involve the use of UML extensions called *UML Profiles*. A UML profile is an extension mechanism for building UML models for particular domains or technologies. It is defined using additional stereotypes and tagged values that are applied to constructs in UML in order to describe a particular modelling problem and facilitate modelling constructs in that domain [18]. UML security modelling approaches are described in the following subsections.

### 3.1.1 UMLSec

UMLSec is an extension of UML used to model security related features, such as confidentiality and access control [84]. It assists in the integration of security requirements into system design by using the UML notation which developers are familiar with [85]. It fo-

cuses on specification and verification of security properties as well as mechanisms such as encryption. UMLSec combines a use-case driven process with a goal directed approach, hence, a security goal tree is developed along side the system specification.

UML provides extension mechanisms with labels, which could be stereotypes, tag-valued pairs or constraints. The UMLSec extension uses labels to express security requirements within UML diagrams in system specification [84, 85]. Stereotypes used include <<secure links>>, <<secrecy>>, <<secure dependency>>, <<critical>> and <<data security>>, which apply to dependencies, subsystems or contained objects of subsystems.

Stereotypes <<data security>> and <<no down-flow>> have associated constraints to ensure protection of data values or attributes which are tagged $\{secret\}$ and belong to objects with stereotype <<critical>>. This could be used to model confidentiality in the system specification [84]. Hence extensions provided by UMLSec allow expression of design choices for security requirements such as confidentiality.

### 3.1.2 SecureUML

An approach for building security languages into a model-driven software development process is proposed by Basin *et al.* and it consists of the following [33]:

i) a security language for specifying security policies,

ii) a system design modelling language to define system models,

iii) a dialect which integrates the security language and the system design language by defining integration points. For example, model elements can be depicted as protected resources of the security language.

This approach is aimed at narrowing the gaps between security models and system design models as well as between design and implementation. It allows formulated security requirements to be integrated with system design at a high level of abstraction.

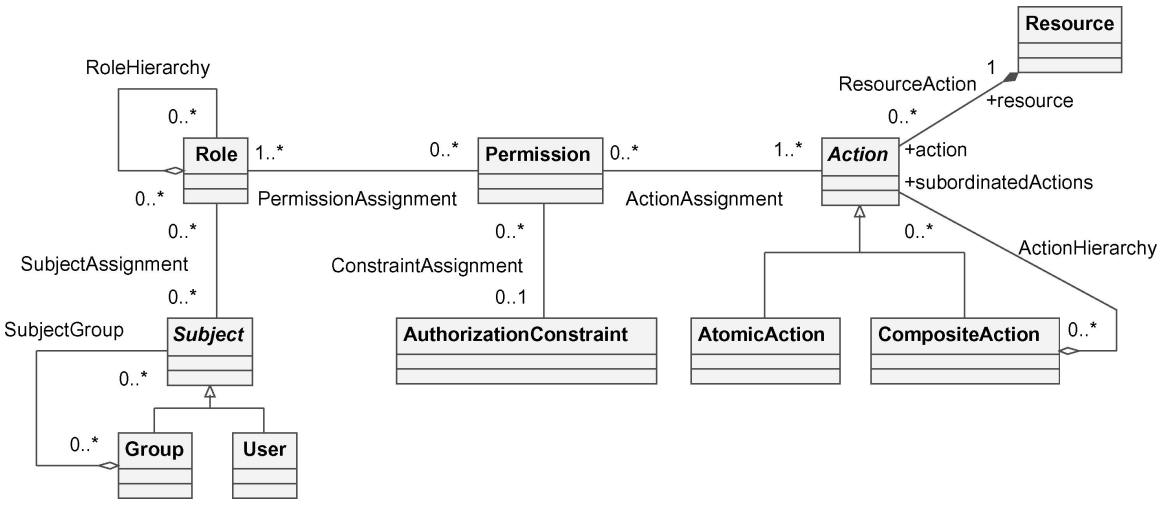


**Figure 3.1.** *SecureUML meta model [33]*

SecureUML is a security modelling language that is focused on modelling access control policies. It can be used to specify control policies for actions on protected resources, which are modelled with the system design language [33]. SecureUML is based on an extension of Role-Based Access Control (RBAC) model with additional support for authorization constraints defined with UML.

Figure 3.1 shows the SecureUML meta model. A Permission grants roles access to one or more actions, where the actions are assigned by the association ActionAssignment and the entitled roles are denoted by the association PermissionAssignment. An AuthorizationConstraint is a constraint, which makes a permission's validity a function of the system state. Constraints are specified as UML Object Constraint Language (OCL) expressions [33].

Resource is the base class of model elements that represent protected resources in the system modelling language. Operations on resources are represented by Action. Actions can be low-level, i.e., they can be directly mapped to actions of the target platform (AtomicAction) or high-level, i.e., they may not have direct counterparts on the target platform (CompositeAction). Resource and Action are generic elements and are used as a means to integrate SecureUML with different system modelling languages. Concrete resource types,

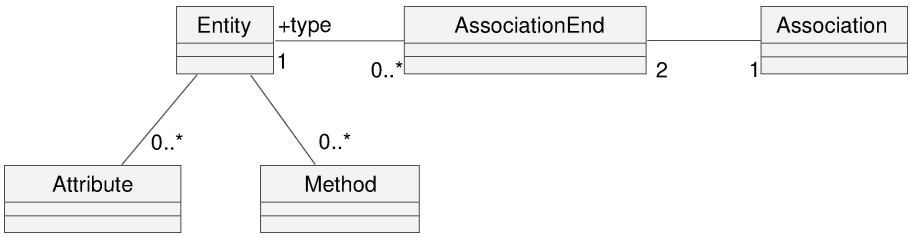


**Figure 3.2.** *A Modelling Language [33]*

actions, and the action hierarchy are defined as part of a SecureUML dialect [33].

In a component-oriented modelling language the resources could be attributes that can be read/updated or methods that can be executed [91], while in a process-oriented language protected resources can be processes with actions for activating, deactivating, terminating or resuming processes [32]. The dialect enables explicit annotation of protected resources with access control policies. To illustrate this, the simple system modelling language shown in Figure 3.2 can be merged with SecureUML to produce the dialect language in Figure 3.3. The dialect language is used to specify authorization policies as shown in Figure 3.4, which depicts the policy to (i) grant users the permission to create and read meetings (ii) grant meeting owners (specified with a constraint) the permission to update or delete a meeting and (iii) grant supervisors the permission to cancel a meeting, i.e, by executing methods "cancel" and "notify".

Access control features can also be generated from SecureUML models, thus mitigating the occurrence of errors in the realization of the access control policies [91, 32]. In comparison to UMLSec, SecureUML allows integration of existing security language policies in defining a dialect.

### 3.1.3 UMLIntr

UMLIntr is a UML profile for specifying intrusion scenarios. These scenarios are abstract descriptions of steps for gaining illegal access to protected resources [81]. Detailed descriptions of these scenarios with a specification or attack language are called intrusion

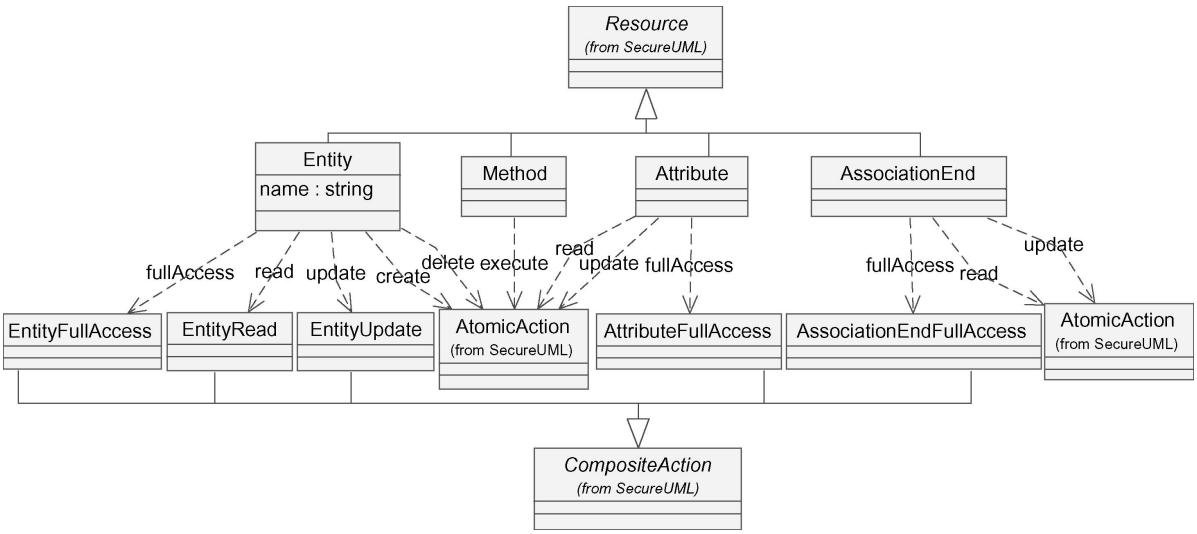


**Figure 3.3.** *A SecureUML dialect [33]*

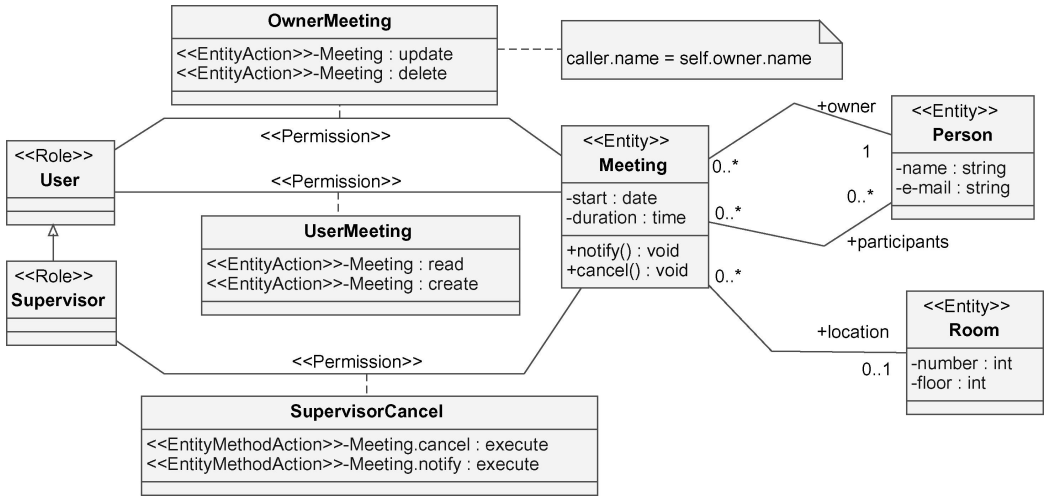


**Figure 3.4.** *Authorization Policy with dialect[33]*

signatures. The intrusion scenario specifications are transformed to signatures which can be used by intrusion detection systems.

Intrusion scenarios can be identified during the requirement engineering phase through services provided by the system. They are represented by the extended use case diagrams of UMLIntr. Other phases of the development process, e.g., design and implementation are also supported by the framework. The UMLIntr profile extends UML with the use of

stereotypes and tagged values, which are based on an attack taxonomy that defines privileges and actions.

The stereotypes used include: <<attacker>> or <<victim>> for actors in a use case or corresponding classes in a class diagram; <<intercept>>, <<alter>>, <<use>>, <<probe>>, <<deny>> which are effects of an intrusion; <<social engineering>>, <<masquerading>>, <<misconfiguration>>, <<abuse feature>> and << implementation bug>> which indicate exploitation methods. UMLIntr tagged values include *initial-privilege* - initial access level of the attacker; *gained-privilege* - gained access level of the attacker; *skill* - skill of attacker; *action* - action taken by attacker; *method* - exploitation method; *listeners* - events required to detect attacks [81].

Figure 3.5 depicts a Smurf attack specified with UMLIntr. Part (a) of the figure is the use case diagram and part (b) is the class diagram. The attacker sends ICMP (Internet Control Message Protocol) "echo request" packets to broadcast address of many subnets, and spoofs the source address to be the intended victim's address. Machines listening on the subnets send echo reply to the victim, with a potential occurrence of a denial of service at the victim [81].

Although confidentiality requirements could be specified by indicating potential "attackers" who could "use" or "probe" through "social engineering", UMLIntr is more suitable for modelling system behaviour and security requirements related to intrusion detection.

### 3.1.4 Summary - UML methods

The UML approaches mainly address security from the system perspective. They model the computer system, its access control mechanisms/policies but they are limited with regards to modelling stakeholder goals. These could be attributed to the primary purpose of UML as a design modelling language. An appropriate security engineering approach should support modelling stakeholder relationships and the goals which might be based on these relationships. In addition, the methods do not support default requirements.

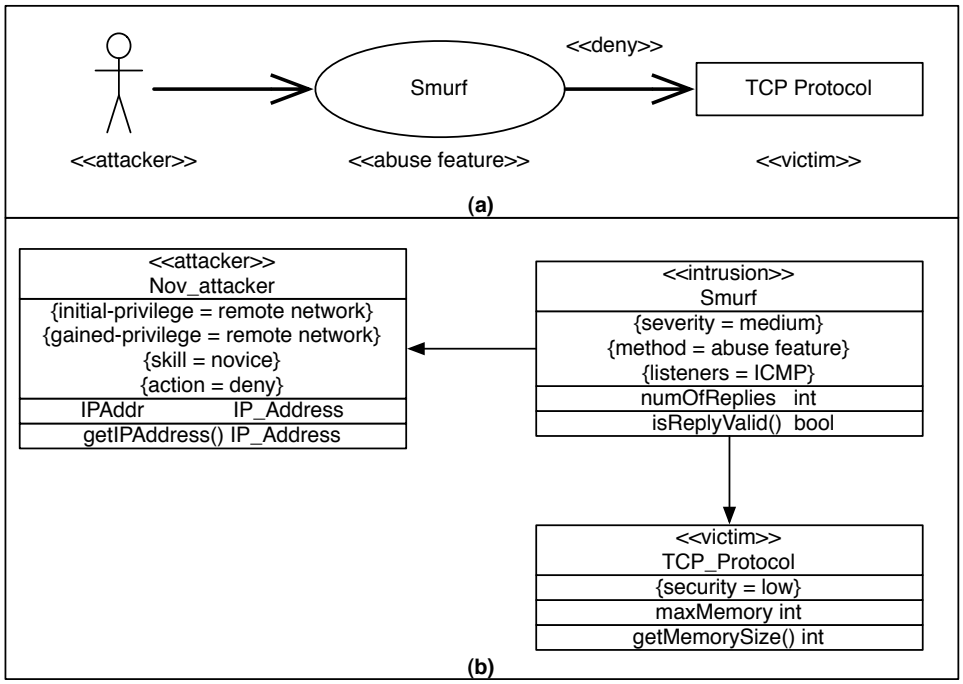


**Figure 3.5.** *UMLIntr Smurf Attack example [81]*

## 3.2 Goal-oriented Methods

Goal-oriented requirement engineering methods use goals for eliciting, elaborating, specifying, analyzing, negotiating, documenting and modifying requirements. *A goal is an objective the system under consideration should achieve* [124]. Goals may be specified at different levels of abstraction, which range from high-level concerns (e.g., improve the system performance) to low-level technical concerns (e.g., compression mechanism for improving space performance). They can be used to specify both functional and non-functional requirements. Goal identification leads to questions that address the "why", "how" and "how else" questions. Requirements are identified during the process of goal elaboration [132], hence they can be used for developing requirements specifications [24].

Goals are used in RE because they provide the following features [124, 132]:

- goal refinements provide the rationale and drive for elicitation and elaboration of lower-level requirements.

- goal refinements can provide a structure for documentation of complex requirements.
- goals provide a level of abstraction, which can be used for conflict detection and possible resolution.
- goals serve as sources for identifying requirements, and help to communicate these requirements to stakeholders. Unlike other methodologies which generally address requirements as being made up of processes and data or objects, the goal-oriented approach capture "why" and "how" relationships in terms of goals.
- goals serve as criteria for determining the relevance of elicited requirements. A requirement should address one of the objectives of the system or stakeholders.
- goals provide traceability of rationales, e.g., design decisions can be traced to initial goals.
- goals can be used for verification of requirement satisfaction.

Goal analysis use a goal graph structure, similar to AND/OR trees, with a (parent) goal refined one or more times into other (sub) goals. Using AND/OR trees involves decomposition of goals into subgoals with AND- or OR- decomposition relations. Given a goal $G$ decomposed (refined) into subgoals $G_1, G_2 \ldots G_n$, through an AND- (OR-), the goal G is satisfied if all the subgoals (at least one of the goals) are (is) satisfied. Goal decomposition and refinements can continue until tangible goals and/or observable goals are identified. A tangible goal is a goal which someone or an agent can satisfy through an appropriate course of action. An observable goal is deemed as satisfied or denied through an observation of the domain [67].

A goal graph is a pair $\langle G, R \rangle$ where $G$ is a set of goals and $R$ is a set of goal relations over $G$. If $(G_1, ..., G_n) \overset{r}{\longmapsto} G$ is a goal relation in $R$, $G_1, ..., G_n$ are source goals (subgoals) and $G$ the target goal of $r$. This can be denoted as $r((G_1, ..., G_n), G)$. Given a goal relation $r((G_1, ..., G_n), G)$, where $r \in \{\text{AND, OR}\}$, and predicates $S(G), D(G)$ which denote there is at least evidence that $G$ is satisfied or denied respectively, satisfaction or deniability propagation can be defined using the following axioms [67]:

- for $(G_1, G_2, ..., G_n) \overset{AND}{\longmapsto} G$
  - i) $S(G_1) \wedge S(G_2) \wedge ... \wedge S(G_n) \rightarrow S(G)$
  - ii) $D(G_1) \rightarrow D(G), D(G_2) \rightarrow D(G), ..., D(G_n) \rightarrow D(G)$
- for $(G_1, G_2, ..., G_n) \overset{OR}{\longmapsto} G$
  - i) $S(G_1) \vee S(G_2) \vee ... \vee S(G_n) \rightarrow S(G)$
  - ii) $D(G_1) \wedge S(G_2) \wedge ... \wedge D(G_n) \rightarrow D(G)$

Methods which integrate goals and goal refinements in requirement engineering include GBRAM [24], as well as KAOS, i* and Tropos [123].

### 3.2.1 Goal-Based Requirements Analysis Method (GBRAM)

GBRAM is used in identifying, elaborating, refining and organizing goals for requirements specification [24]. It comprises techniques, including goal analysis and goal refinement activities, for application of heuristics and inquiry to analyze goals, scenarios and obstacles [25].

Goal analysis activities include [25]:

- exploration - to examine available information sources.
- identification - to extract goals from sources such as process descriptions, e.g., flow charts or Entity Relation diagrams. Agents (entities or processes which seek to achieve goals within an organization or system), stakeholders and constraints (requirements which place conditions on the achievement of a goal) should also be identified [24].
- organization - to classify goals and organize goals based on goal dependency relationships.

Goal refinement is concerned with goal evolution from the step to identify goals to the step to operationalize the goals. The activities include [25]:

- refinement - involves pruning the set of goals. Refinements occur when synonymous goals are reconciled or by removing redundancies. For example, goals "Meeting arranged" and "Meeting scheduled" are synonymous and can be reconciled [24].
- elaboration - analyzing the set of goals by considering goal obstacles and goal scenarios to identify hidden goals and requirements.
- operationalization - translating goals into operational requirements for the specification.

GBRAM heuristics are aimed at assisting analysts with prescriptive guidance to manage different levels of detail in the available information. The heuristics are defined to guide identification, classification, refinement and elaboration tasks. For example, the heuristic, "Stakeholders may express goals in terms of activities" is an identification heuristic that describes the tendencies of stakeholders to express requirements in terms of operations and actions [25].

### 3.2.2 KAOS

KAOS facilitates capturing and modelling requirements and their relationships through the use of goals and other concepts. Concepts used in the KAOS language, as depicted in Figure 3.6 include [126]:

- Object: a thing of interest in the system. This could be specialized as an entity (autonomous), a relationship (subordinate) or an event (instantaneous).
- Agent: an object which acts as a processor of some operations (actions). An agent could perform an operation or monitor/control an object.
- Operation: an input-output relation over objects, whose application defines state transitions.
- Goal: a goal is an objective, which the system needs to satisfy. Goal analysis can identify high-level goals, which provide rationale ("why") for the goal, as well as

subgoals that give clear definition of the goal. A goal graph, which is a semantic network of goals based on their relationships, is developed during the analysis [79]. AND-refinement of a goal relates the goal to subgoals, such that satisfaction of the subgoals is a sufficient condition for satisfying the goal. An OR-refinement relates the goal to alternative subgoals such that satisfying one of the refinements is a sufficient condition for satisfying the goal. Goals refinement structures form AND/OR acyclic graphs.

Goals can be classified based on the pattern of temporal behaviour required (i.e., Achieve, Avoid, Maintain, Cease) or based on the potential requirements derived, which have heuristics for goal decomposition and satisfaction, e.g., SatisfactionGoal, SecurityGoal, SafetyGoal, InformationGoal.

- Requisite: a leaf goal, which is assigned as responsibility of an agent. Requisites are operationalized by operations and objects.
- Requirement: a requisite assigned to a software agent.
- Assumption: a requisite assigned to an agent in the domain/environment.

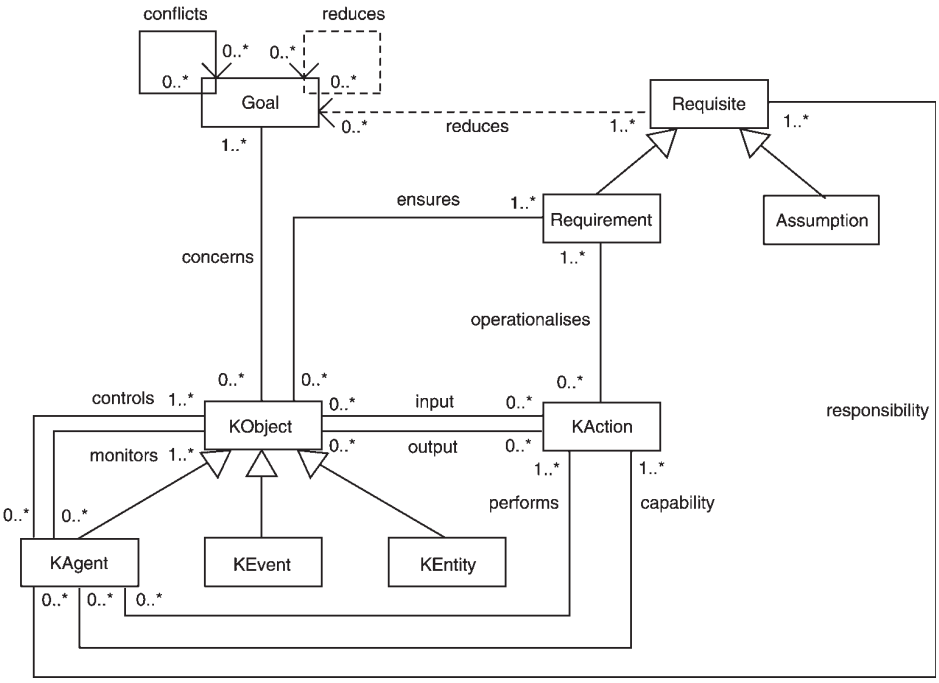


**Figure 3.6.** *KAOS Meta model [79]*

Constructs are represented with the KAOS language, which consists of [110, 126]:

i) an outer declarative layer for conceptual modelling and requirement traceability through semantic net navigation. The semantic net layer can be represented with a graphical concrete syntax or in textual form.

ii) an inner formal assertion layer to formally define the concepts and provide formal reasoning. The formal assertions are specified in a real-time temporal logic format.

The following example illustrates the use of KAOS for specification [126]:

**Goal** Achieve[ParticipantsConstraintsKnown]

**Concerns** Meeting, Participant, Scheduler

**Refines** MeetingPlanned

**RefinedTo** ConstraintsRequested, ConstraintsProvided

**InformalDef** A meeting scheduler should know the constraints of the various participants invited to the meeting within some deadline *d* after invitation.

**FormalDef** $\forall$ m:Meeting, p:Participant, s:Scheduler

$$\text{Invited(p,m)} \wedge \text{Scheduling(s,m)} \Rightarrow \Diamond_{\leq d} \text{Knows(s,p.Constraints)}$$

The goal, *ParticipantsConstraintsKnown* is a subgoal of *MeetingPlanned*, and it should eventually hold as indicated by *Achieve*. ParticipantsConstraintsKnown is refined into subgoals *ConstraintsRequested* and *ConstraintsProvided*. The goal is described with natural language statement in *InformalDef*, while *FormalDef* provides the formal assertion for the specification. The formal representation uses temporal logic formalism and the operators used include the following:

- $\Diamond_{\leq d}$ – some time in the future within deadline *d*
- $\blacklozenge_{\leq d}$ – some time in the past within deadline *d*
- $\Box_{\leq d}$ – always in the future up to deadline *d*
- $\blacksquare_{\leq d}$ – always in the past up to deadline *d*

KAOS models can be used to capture attackers, attackers' goals and capabilities, the vulnerabilities attackers can monitor or control and the attacks that satisfy their goals. These are represented as anti-models [125]. The attacker's goals are termed, *anti-goals*, and represent malicious obstacles, which threaten security goals. Initial anti-goals can be specified by negating security goal patterns for sensitive objects. Building the anti-model involves anti-goal refinement to derive leaf nodes that are either software vulnerabilities observable by the attacker or anti-requirements implementable by the attacker. New security requirements can be derived as countermeasures by applying threat resolution operators to the specification of the anti-requirements and vulnerabilities identified. Alternative countermeasures can be derived using operators such as goal substitution, agent substitution, anti-goal prevention, anti-goal mitigation, protect vulnerability (vulnerability unmonitorable by attacker) and defuse threat (anti-requirement condition uncontrollable by attacker) [125].

The framework has also been used to reason about confidentiality requirements in software development. This extension uses epistemic logic to model the knowledge of agents over time in order to represent their "memory". It uses a catalog of patterns, based on degree of approximate knowledge and knowledge timing, to capture the requirements over state variables [57].

### 3.2.3 i* Framework

The i* (*distributed intentionality*) framework proposes an agent-oriented approach to requirements engineering based on the intentional characteristics of agents. It incorporates NFR framework's softgoal and subgoal contributions for goal refinements [52].

The framework is applied in contexts that comprise multiple parties with strategic interests that might be conflicting or synergistic. These include information systems requirements engineering, business process modelling and redesign, software process modelling [131].

The i* framework process consists of (a) identifying the actors (b) identifying goals/-tasks and (c) identifying dependencies. The concepts of the framework include [90]:

- An actor - a unit, which can be ascribed intentional dependencies. This can be a role, an agent or a position. A role is an abstract actor with some responsibility (e.g., patient, administrator) while an agent is a concrete actor with particular capabilities and functions (e.g., John Doe, PDA device). A set of roles packaged to be assigned to an agent is called a position.
- Goal - a condition or state an actor would like to achieve. Goals could be softgoals, which are typically non-functional [63]. A hard goal is usually functional.
- Task - a course of action to produce a desired effect. It represents a specific procedure to be performed by an agent.
- Resource - a physical or information entity [63].

Agents attribute intentional properties (such as satisficing softgoals, achieving goals,

performing tasks, furnishing resources, beliefs) to each other and reason about strategic relationships. The agents consider alternative configurations of dependencies to assess their strategic positioning in a social context and the dependencies result in opportunities as well as vulnerabilities. Networks of dependencies are analyzed using a qualitative reasoning approach. The framework consists of two components: a Strategic Dependency (SD) and a Strategic Rationale (SR).

Strategic Dependency (SD) model describes the relationship among actors in an organizational context. It is represented as a graph with nodes representing actors, which are connected by dependency links. The depending actor is the *depender*, and the actor depended on is the *dependee*. The object of the dependency, called the *dependum* can be a softgoal, a goal, a task or a resource [52]. The SD does not model flow of entities but the intentional relationships among agents. The SD provides a means of analyzing opportunities and vulnerabilities. An example of an i* SD model is shown in Figure 3.7 (a), which shows two goal dependency links between actors "Physician" (depender in both cases) and "Medical Info. Sys" (dependee).

Strategic Rationale (SR) describes stakeholder concerns and how these are affected by various systems configurations. The SR model provides a detailed representation of the dependencies by modelling both internal and external intentional relationships of the actors. It is also represented as a graph and the framework uses qualitative reasoning for evaluating the justification network.

Tasks are connected to a goal through *means-ends* links and the goal is satisfied if any of its tasks is satisfied. A task may be detailed into subgoals, subtasks, resources and softgoals through *Decomposition* link. The task is accomplished if all its subcomponents are satisfied. High-level softgoals are refined into more specific softgoals or operationalized in terms of tasks through contribution link [131, 90].

Figure 3.7 (b) is an SR model providing further details of the SD model of Figure 3.7 (a). In the model, the soft goal, "Limited Use and Disclosure" contributes to the soft goal "Confidentiality" of medical record. The task to provide information with no personal

identifiable information is one of the means to realize the limited disclosure goal. This task depends on the medical information system's task to protect personal information.

In addition to modelling stakeholder security goals, the i* framework can be used to potentially identify the vulnerability of the system through dependency relationships. Each actor is considered a potential attacker; an attacker of this type has the legal intentions, capabilities and relationships of the actor in addition to its intentions. Incoming dependency links to this "attacker" actor are then considered for possible threats and if so, the dependency link is identified as a vulnerable point in the system. The chain effects of vulnerable points are identified for potential effects on other actors or parts of the system [90].

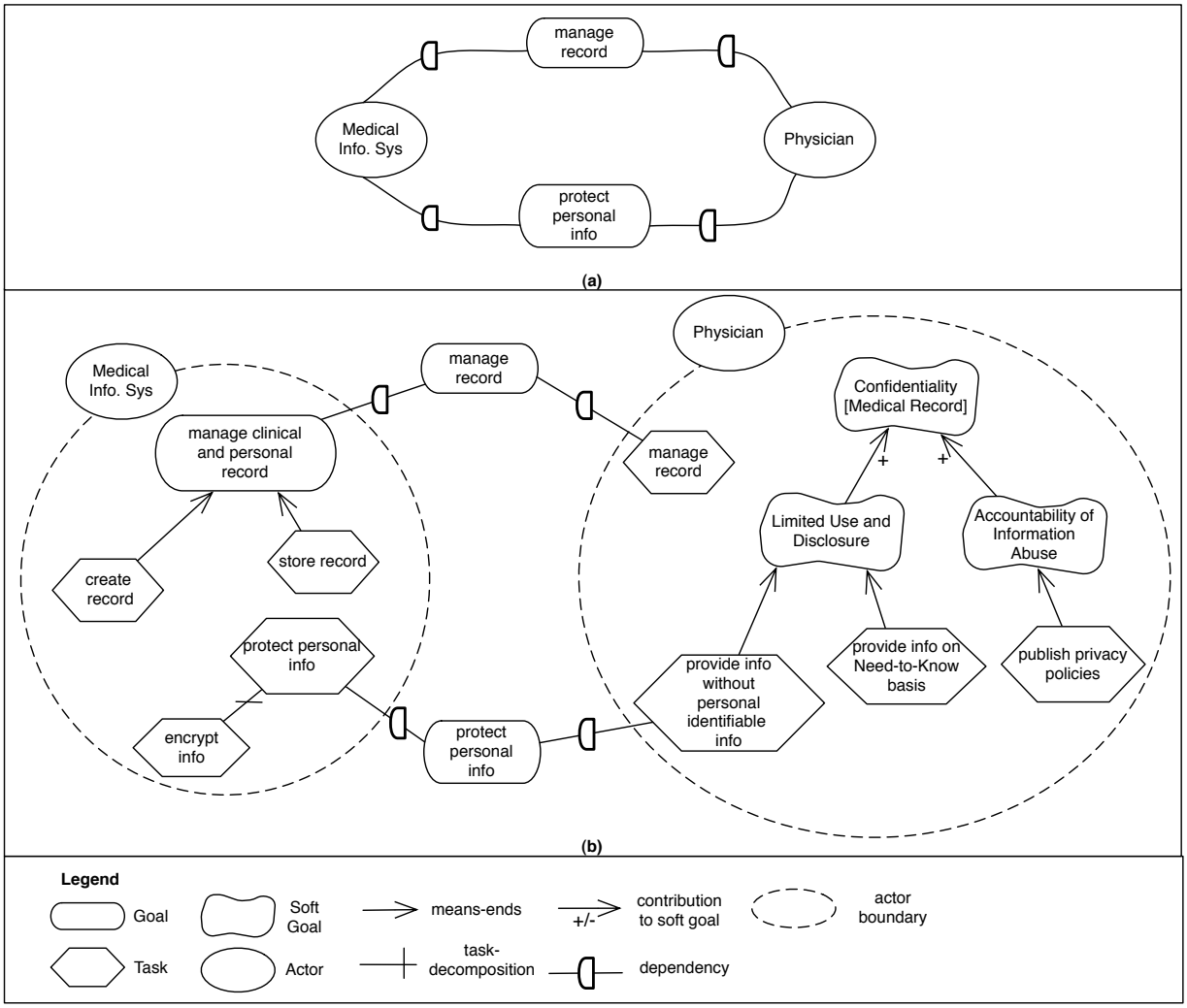


**Figure 3.7.** *i* SD and SR models*

### 3.2.4 Tropos

Tropos is aimed at developing an agent-oriented software engineering methodology, starting from early requirements. It adopts the i* framework to model early and late requirements as well as architectural and detailed design. It is supported by a variety of analysis tools based on formal methods [63]. A section of the Tropos meta model is shown in Figure 3.8.

Tropos has two key characteristics; firstly, the notion of an agent and its related no-

tions, e.g., goals and plans (equivalent to task in i*) are used in all phases of software development, from early analysis to the actual implementation. In addition, it allows for a better understanding of the environment in which the software will operate by covering the very early phases of requirements analysis and the interactions that should occur between software and human agents in this environment.

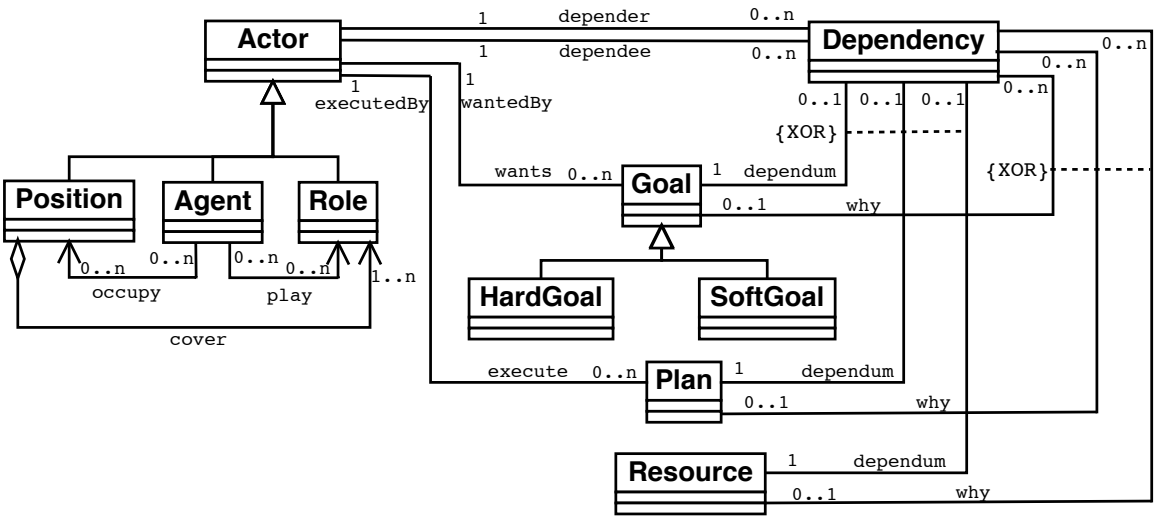


**Figure 3.8.** *Actor concept and Dependency relationship in Tropos meta model [121]*

Formal Tropos provides a formal specification language which supports temporal specification. The Formal Tropos framework combines work from research engineering and formal methods in order to support precise modelling and formal analysis of early requirements. Formal Tropos provides model checking as a verification technique [64].

#### 3.2.4.1 Secure Tropos

Secure Tropos extends Tropos by providing features that enable modelling and analysis of security requirements [68]. This is done by integrating trust, security and system engineering. The notions of ownership and offer of a service, as well as functional and trust dependencies are made explicit in Secure Tropos [69].

Ownership is a relationship between an actor and a service (a goal, task or resource),

which indicates that the actor is owner of the service. Trust is a social relationship which depicts that actor A beliefs actor B will not misuse the service which was granted to B. Delegation represents formal passage of permission [69].

The dependency relationships are refined to enable analysis of security requirements as follows:

- at-most delegation - the delegator wants the delegatee to at most fulfill a service - this is delegation of permission and the delegatee has the permission to fulfill the service.
- at-least delegation - the delegator wants the delegatee to at least fulfill a service - this is delegation of execution. In this case the delegatee gets the service fulfilled.
- at-most trust - an actor (truster) trusts another actor (trustee) will at most fulfill a service and will not abuse this trust. This applies to permission.
- at-least trust - the truster trusts the trustee will at least fulfill a service. This is trust of execution.

Permissions apply to services which are resources, while executions can be applied to goals or tasks.

Although monitoring is not a primitive concept in Secure Tropos it is used as a method to overcome the absence of trust, that is when services are delegated to untrusted actors. In this case a trusted actor is delegated to monitor the untrusted actors [69]. Monitoring can also be used for trusted actors as an audit mechanism, which is important for security-aware systems. Secure Tropos provides a formalization of these concepts to facilitate formal reasoning, and a prototype tool has been developed [70].

### 3.2.5 Misuse Cases and Goal-oriented Methods

Negative use cases or misuse cases describe unwanted behaviour in a proposed system and can be used to elicit security requirements early in the development process [23, 115]. A misuse case is defined as “a sequence of actions, including variants, that a system or other entity can perform, interacting with misusers of the entity and causing harm to some

stakeholder if the sequence is allowed to complete" [115]. A misuser is an actor which intentionally or inadvertently initiates a misuse case.

Relationships *extend, include and generalize* which are used between use cases can also be used between misuse cases. In addition, the following relationships can also be specified between use and misuse cases: i) threaten - indicates a misuse case can exploit or hinder a use case ii) mitigate - indicates a use case is a countermeasure against a misuse case [115]. Eliciting security requirements with misuse cases comprises steps to identify critical assets, threats and security requirements (or countermeasures). The security threats can be expressed with misuse cases and misusers. Risk analysis is aided by relationships identified between misuse cases. The identified security requirements are specified as security use cases which mitigate the misuse cases.

Misuse cases provide informal representations which enable easy communication between analysts and stakeholders about possible threats. Misuse cases use a multilateral approach that allows analysis from the perspective of the misuser (attacker). This approach improves the chances of identifying threats which could have been missed. Furthermore, the inherent relationship between misuse cases and use cases provides a link between functional requirements, specified with use cases, and the threats (misuse cases) and countermeasures. The link enables analysis of the functional requirements with respect to the security threats and security requirements [115].

Misuse cases can complement goal-oriented methods by providing a method to identify obstacles to elaborated goals. An obstacle captures an undesirable behaviour which obstructs some goal, e.g., a threat to a security goal. The identified obstacle acts as root of an obstacle tree for analysis. In KAOS, anti-models, which include attackers goals, i.e., anti-goals, are used for analysis (cf. Page 47). The obstacle can be addressed by different resolution strategies, e.g., obstacle tolerance, which can be realized by adding new goal(s) to mitigate the effect of the obstacle [128]. The addition of new goals is similar to adding security use cases to mitigate misuse cases [115].

### 3.2.6 Summary - Goal-oriented methods

The goal-oriented methods are better suited for requirement analysis in comparison to the UML methods (cf. Section 3.1). This is due to the goal-oriented methods' approach to identify stakeholders, stakeholder relationships and goals, and goal analysis in contrast to the system design perspectives of the UML methods. However, similar to the UML methods, existing goal-oriented methods do not support representations and analysis with default requirements and exceptions, as well as specifying stakeholders who may specify exceptions (cf. Section 1.2.2).

## 3.3 Formalizing Natural Language Requirements

Confidentiality and privacy regulations, and stakeholder concerns are usually specified in natural language. This ensures the documents are accessible to different stakeholders regardless of their technical expertise. However, the natural language documents can be complex and prone to incorrect interpretation by users including software engineers. These could result in a costly development process, e.g., due to inadequate or incorrect designs. A process which complements analysis of requirements by providing requirements extraction from natural language format will reduce ambiguity and aid analysis. This process has been identified as one of the major tasks in requirement engineering [105].

Precise and formal representations of requirements are useful to software engineers because these representations are amenable to automated analysis, e.g., validation of the specification. However, specifications should also address the needs of domain experts, who might not be familiar with the formal notations. This can be realized with a method which supports both formal and informal representations for the domain experts and the engineers.

Key factors for extraction of requirements from regulations in natural language formats include the following [103]:

- Classification and annotation

Tagging regulations with metadata provides a means of categorization of different parts of the regulation. With categorization, accessing and navigating relevant sections of the often large documents is easier. Sections could be tagged as relevant to information collection, disclosure or tagged as relevant for system-level requirements. For example, Part 1 of the Canadian Personal Information and Protection and Electronic Documents Act (PIPEDA) [6] can be annotated as relevant to protection of information collected, used or disclosed while Part 2 can be categorized as dealing with electronic compliance with paper based regulations.

Attaching relevant annotations to sections of the text can be used for different purposes, e.g., to indicate an ambiguous section for further clarifications with stakeholders. Annotations can also assist analysts by providing support for associating texts to relevant concepts for requirement extraction.

- Traceability

  Having traceability from extracted requirements to source document, and between references helps in improving analysts' understanding of the source document, as well as the extraction of relevant requirements. Traceability links also help in tracing relationships which could occur between relevant concepts defined in different sections of the document. In addition, priorities among requirements are often established through references which could be specified with traceability links. Traceability helps in managing evolution of the requirements as changes are made to the source regulation [100].

- Searching and Navigation

  The requirements engineering process is improved if analysts can search and navigate the often complex source documents in order to improve understanding. The search and navigation of the regulation can be enhanced with meta data tagging/annotation of the natural language representation. Searching could be done for specific terms or by using general concepts.

  Accessing regulations in both formal and natural language formats would also im-

prove analysis. Navigation between the formats is established with traceability between the formal and natural language representations. This allows effective comparison of the extracted requirements with the source documents.

- Prioritization and exceptions

  Extracting requirements from natural language regulations should address the inherent hierarchical nature of the regulations. Exceptions, which take precedence over normative requirements often exist in regulations (cf. Section 2.4). Therefore, the method of extracting and analyzing the requirements should assist analysts in managing the relationships among the requirements and in determining the requirements precedences, e.g., among jurisdictions.

- Data dictionary and glossary

  Having consistent terminology is critical in the design of software systems, particularly for systems complying to a regulation. A domain-specific data dictionary, which supports different stakeholders in establishing a common glossary for specification, design and compliance is needed. This will ensure consistent interpretation in the context of specific regulation(s).

### 3.3.1 Extraction from Natural Language Documents

Efforts have been made for legislation retrieval from legal documents. Moens [95] describes an approach for defining legislation retrieval models from legislative sources, which are represented as document-centric XML documents. The documents comprise mostly unstructured content, such as natural language text, as well as structured content. Tags for the structured contents mark structural metadata for different hierarchical components of the document, e.g., a statute can consist of chapters, sections and articles. Data-centric documents would however be better suited for the retrieval of goals, obligations or rights from legal documents because the tags could mark concepts such as jurisdictions, stakeholders and components of rules.

The following sub sections describe some methods for deriving formal representations of requirements specified in natural language.

#### 3.3.1.1 Restricted Natural Language Statements (RNLS)

RNLS are structured natural language statements that describe events which identify actors, actions and objects. Each RNLS has a primary actor (subject), an action and at least one object, and is defined in terms of an activity, with possibly nested activities defined as separate RNLS. Statements from the source document are restated in RNLS by decomposing into distinct activities [44].

The RNLS are used for *Semantic Parameterization*. Semantic parameterization involves mapping the RNLS into semantic models, which can be used for formal analysis [43]. The semantic models describe the relationships between concepts and map RNLS into machine readable formats. Semantic models are formally defined with the following modelling notation:

- a unary relation ($\sigma$) - to define the root or main concept.
- associative relation ($\alpha$) over concepts and parameter names - to define parameters.
- declarative relation ($\delta$) over parameters and concepts - to assign values to parameters. Parameter values could be concepts with other parameters, e.g., an activity.

Table 3.1 shows the translation of the following statements into restricted natural language statements:

"*You have the right to request a restriction or limitation on the medical information we use or disclose about you for treatment, payment or health care operations. You also have the right to request limit on the medical information we disclose about you to someone who is involved in your care or payment of your care, like a family member or friend. For example, you could ask that we not disclose information about a surgery you had.*

*We are not required to agree to your request. If we do agree, we will comply to your request unless the information is needed to provide emergency treatment, or if the disclosure*

*is required by law.*"

| |
|---|
| $\mathbf{RNLS_1}$: Medical organization may agree to ($\mathbf{RNLS_5}$) unless ($\mathbf{RNLS_9}$). |
| $\mathbf{RNLS_2}$: Medical organization may agree to ($\mathbf{RNLS_5}$) unless ($\mathbf{RNLS_{10}}$). |
| $\mathbf{RNLS_3}$: Medical organization may agree to ($\mathbf{RNLS_6}$) unless ($\mathbf{RNLS_9}$). |
| $\mathbf{RNLS_4}$: Medical organization may agree to ($\mathbf{RNLS_6}$) unless ($\mathbf{RNLS_{10}}$). |
| $\mathbf{RNLS_5}$: Patients may request limitation on ($\mathbf{RNLS_7}$). |
| $\mathbf{RNLS_6}$: Patients may request limitation on ($\mathbf{RNLS_8}$). |
| $\mathbf{RNLS_7}$: Medical organization use medical information for treatment, payment or health care operation purposes. |
| $\mathbf{RNLS_8}$: Medical organization disclose medical information to stakeholders involved in treatment or payment. |
| $\mathbf{RNLS_9}$: Medical organization use medical information for emergency treatment. |
| $\mathbf{RNLS_{10}}$: Medical organization disclose medical information required by law. |

**Table 3.1.** *RNLS representation*

In addition to the subject, action object pattern, RNLS could use patterns that give additional information about a subject or object, or patterns which follow condition keywords, e.g., "unless" [44]. Requirement (1) in (ii) of Section 1.2.1 is represented by the statements in Table 3.2. The statements have the following constraints and obligations:

Constraints:

A) Organisation connected to the EHRi

B) Organisations hosting components of EHRi

Obligations:

1) Organisations must identify purpose at or before collecting PHI.

2) Organisations must inform patient/persons of purpose in a readily understandable manner prior to collecting their PHI.

| |
|---|
| $\mathbf{RNLS_1}$: Organisations who ($\mathbf{RNLS_5}$) must ($\mathbf{RNLS_7}$) for which PHI will be collected, used and disclosed. |
| $\mathbf{RNLS_2}$: Organisations who ($\mathbf{RNLS_6}$) must ($\mathbf{RNLS_7}$) for which PHI will be collected, used and disclosed. |
| $\mathbf{RNLS_3}$: Organisations who ($\mathbf{RNLS_5}$) must inform patient/persons of purpose, in a readily understandable manner prior to collecting their PHI. |
| $\mathbf{RNLS_4}$: Organisations who ($\mathbf{RNLS_6}$) must inform patient/persons of purpose, in a readily understandable manner prior to collecting their PHI. |
| $\mathbf{RNLS_5}$: Organisations connected to the EHRi |
| $\mathbf{RNLS_6}$: Organisations hosting components of EHRi. |
| $\mathbf{RNLS_7}$: Identify purpose. |

**Table 3.2.** *RNLS for Requirement (1) in (ii) of Section 1.2.1*

The semantic model instance for obligation (1) is derived as follows:

- $\sigma$(activity1)
- associative relations
  $\alpha$(activity1, subject1), $\alpha$(activity1, action1), $\alpha$(activity1, object1)
- declarative relations
  $\delta$(subject1, organisation), $\delta$(action1, identify), $\delta$(object1, purpose).

This is represented by an activity pattern as follows:

activity [ obligation ] {
    subject = organisation
    action = identify
    object = purpose
}

For analysis, queries performed over the semantic models can be used to partially order rights and obligations based on their level of refinement. Furthermore, exceptions can

be handled by using priorities. Exceptions have higher priorities and overrule rights and obligations with lower priorities [44].

A translation of the privacy requirements of Canada Infoway's Privacy and Security requirement document [2] into RNLS is provided in Appendix B. Drawbacks of the RNLS method include:

- it is tedious for large requirement set because there is currently no adequate tool support to create the RNLS.
- it is difficult to perform semantic parameterization of the RNLS because there are no meta information provided in the RNLS translations (cf. Sections 3.3, 3.3.1.1). This lack of meta information, necessitates the need for additional steps for deriving semantic models from the RNLS.
- it produces multiple nested and/or cross referenced sentences that might not be easy to comprehend for large and complex documents. This is often caused by decomposition of an original source statement into multiple RNLS.
- it lacks support for navigating from the original text to the corresponding RNLS. Although each statement can be mapped to an index from the original text, the lack of tool support and the multiple nested RNLS derived hinder navigation to the original part of the requirement document.

Although RNLS are accessible to different stakeholders, these limitations hinder adoption.

#### 3.3.1.2 Goal Definition Language (GDL)

The GDL method proposes representations and steps for formalizing natural language goals. A motivation for the approach is to address ambiguity which occur when translating goals specified in natural language to a formal system to facilitate verification using a goal model. The goal model acts as a shared frame of reference for domain and formal method experts, and it defines a goal in terms of a *start event*, *a (pre) condition* for the event, an *end event* and some *desired behaviour* between the start and end events [119].

Therefore, the goal can be represented in a process model. The steps for translation to a target formalism are [118, 119]:

- Reduction: involves the description of the desired behaviour explicitly. The essential aspects of the goal are preserved while unnecessary information are removed. This step is performed by the domain expert. For example, the following original natural language text:
  *"The percentage of patients in the last year, with whom the possibility of breast reconstruction was discussed before mastectomy was performed"*
  can be reduced to:
  *"The possibility of breast reconstruction should be discussed with all patients prior to mastectomy"*
- Normalization: involves writing a reduced goal in the goal model structure. The goal is transformed into a normal form consisting of the four elements of the goal model, i.e.,
  $^{C}$[Condition] $^{S}$[Time-range Start] $^{E}$[Time-range End] $^{B}$[Desired Behaviour]
  Complex and nested goals can be expressed through nesting with the *Desired Behaviour* element. Assumptions and ambiguities need to be made explicit in the normal form, therefore, domain knowledge is essential. A normalization of the reduced goal above is as follows:
  $^{C}$[For women with breast-cancer],$^{S}$[after start of the medical care] but $^{E}$[before commencing mastectomy], $^{B}$[the possibility of breast reconstruction should have been discussed with the patient].
- Formalization: translates the structured natural language representation into the GDL form, which can be used by formal method experts. The GDL is a formal expression language of the goal model and keeps the same structure as the goal model. The GDL for the example above is shown below:

**Goal** Cancer Reconstruction

**Precondition**

For women with breast-cancer

**Time-specification**

**From** the start of the medical care

**Until** start of mastectomy

**Observe-during-period** $\geq 1$

discuss possibility of breast reconstruction

with the patient

GDL defines elements for Time-specification (e.g., period delimiters such as *Open-until*, *Duration*) and Behaviour (e.g., *Maintain-during-period, Avoid-during-period, Sub-goal*). The elements are defined independently of any target model, which makes them reusable with different models.

- Attachment: involves further formalization of the GDL representation from the previous step. The representation of the goal at this step uses a GDL extension, which incorporates elements of a process model. This is realized by substituting elements with corresponding concepts of the process model. A GDL extension, using guidelines developed for breast cancer and formalized in Asbru, a plan-specification language, has been developed [119].
- Translation: involves translating the GDL to the logic of a verification tool. Tools which support the temporal logic formalism expressed by the process can be used.

The GDL approach, through its goal model, provides an intuitive wrapper around temporal logic framework for goal formalization. The translation process also produces goal representations in structured natural language, which domain experts can utilize [119]. Formalized goals are used for verification of a system's behaviour (i.e., the medical guideline written as an executable program) using the temporal logic formalism.

In order to effectively model confidentiality and privacy requirements, the GDL would have to be integrated with models which support representation of the requirements' con-

cepts and their relationships (e.g., stakeholders, data, goals with exception etc.). This would involve expressing the requirements in the GDL's "Desired Behaviour" field. This integration could potentially be used for verification of system compliance with particular regulations.

#### 3.3.1.3 An Architecture for Modelling through Terminology

Cerbah and Euzenat [51] describe an architecture which uses terms as a bridge to link texts in documents and their formal representations. The architecture is based on a term extraction and management tool. It helps in improving traceability by using terms as links between formal concepts and the textual sources. Manually adding or managing traceability links can be tedious and time consuming, therefore, tool support to realize this at different phases of software development would be beneficial.

Some terms in textual sources represent concepts defined in the formal models. These terms make up an intermediary level between the texts in the documents and the formal models. These terms also serve as means of providing traceability links between the models and the documents because they can be used as sources of indexing and navigation [51].

The implemented system comprises the following components [50, 51]:

- XTERM

  This handles document management and linguistic processing. It provides terminology extraction from English or French documents with browsing features over the extracted data and the source documents.

  Terminology extraction is achieved by scanning the document to extract text fragments, which are subsequently used in linguistic processing. The extracted terms are organized hierarchically. XTERM has features for indexing and for the generation of hyperlinks from the terms to the document fragments.

- TROEPS

  The models are expressed using TROEPS, an object-based knowledge representation system. TROEPS provides functions for knowledge manipulation, such as filtering

queries (to find objects of particular concept which satisfy field constraints) and value inference (e.g., through default values). The knowledge base manipulation is done through an XML interface, and it offers a web server that facilitates the knowledge base browsing and editing.

Model generation is achieved over both components and the implementation uses XML format for structured data exchange. Class hierarchies corresponding to the term hierarchies generated with XTERM can be created in TROEPS using the TROEPS XML interface. Conversely, calls can be made from TROEPS to XTERM to display textual fragments, which correspond to a concept.

Although this implementation allows model generation from hierarchical terms identified during the term extraction process, it is limited due to explicit linguistic structural relationships used in the extraction process. For example, XTERM can identify the relationship between the terms "Flight Plan" and "Current Flight Plan", however, it cannot identify the relationship between the terms "Aircraft" and "Fighter". In addition, the system cannot generate attributes and relations between classes [51].

### 3.3.2 Structured Natural Language Specifications

Requirements could be specified in a structured natural language format by using a controlled syntax and semantics or adopting particular writing styles. Wilson [130] suggests using simple sentence structures with words and phrases based on their formal definitions. The sentences are usually made up of basic elements: entities, actions, events and conditions.

Structured natural language could also use semi-formal notations, which make it easier to translate into a target formalism for analysis. Using semi-formal notations can help to reduce problems of ambiguity, inaccuracy and inconsistency which are observed with natural language due to variations among usage of words and phrases [130].

In the following subsections, methods using structured sources as starting points for

formalized specifications are described.

#### 3.3.2.1 Stimulus Response Requirements Specification

The Stimulus Response Requirements Specification (SRRS) is designed as a method to apply a systematic process of formalization to semi-formal notations. It has been applied to the Threads-Capabilities requirement specification notation, which is used in specifying air-control systems [55]. Threads-Capabilities is a semi-formal, structured, natural language notation method developed by Raytheon Systems [54]. SRRS provides a formalization process for Threads-Capabilities, with the original notation intact but offers the benefit of a precise, unambiguous notation that could be used for automated tool support.

The Threads-Capabilities structure is based on a thread and a capability. The thread is a specification unit which describes actions performed by the system as a result of stimuli. It defines a connection through the system from an external stimulus to a response, similar to use cases, which describe what the system needs to do. A capability is a reusable mechanism, triggered internally.

The Threads-Capabilities notation consists of the following structure for the threads:

- title: a unique identifier of the thread
- overview: a high level description of the thread, indicating the processing the thread provides for valid stimuli
- stimulus: describes the external stimuli that triggers the thread
- response: describes the output events of the thread
- requirements: indicates functional requirements related to the thread. A requirement has a process for generating a class of responses for each class of stimuli
- performance: describes the average and maximum response times for the stimulus-response pair of the thread

The Thread-Capability and the SRRS notation use a black box approach to describe software requirements in terms of external inputs and external responses thereby limiting

the inclusion of design details in the specification and enhancing the suitability of the specification for testing. Groups of stimuli and responses might be related by requirements, hence the SRRS notation supports the definition of matching rules to identify a stimulus that matches a response [54, 55]. The semi-formal SRRS notation consists of the sections described above as well as a *Declaration* section, which introduces data types, data objects, functions and predicates [54].

The concrete syntax for a simple requirement in SRRS notation is as follows:
**Upon receipt of a** [ group of stimulus name ], **the** system **shall send a** [ group response name ].

A restricted vocabulary is used in the syntax, e.g., the action verbs for the response ("send" in the syntax above) [54]. The non-bold text of the syntax are filled by the author. Conditions can also be specified for a stimulus-response requirement, e.g., the condition "x" can be specified for a requirement as follows:
Upon receipt of a [ w request ], if {x} the system shall send a [ y response ].

A partial specification for a library system in the semi-formal SRRS notation is shown below [55]:

> **Title**: Search Catalogue
> **Overview**: The Search Catalogue unit describes the processing performed when an operator requests to search for an item. The results of the search are sorted in alphabetical order.
> **Stimuli**:
> 1) The Online Library System shall satisfy the requirements described below upon receipt of a [search request] from:
> a) Operator<search request>;
> b) Remote Operator<search request>.
> **Responses**:
> 1) As specified by the requirements described below, the library system shall return an [acceptance] to:

a) Operator<acceptance>.

2) As specified by the requirements described below, the library system shall send a [search response] to:

a) Operator<search response>;

b) Remote Operator<search response>.

**Requirements**:

1) Upon receipt of a [search request], if {the input is not rejected}, the system shall return an [acceptance].

2) Upon receipt of a [search request], the system shall send a [search response]. {The search response is sorted alphabetically}.

**Declarations**:

System State Components:

1) "The search response is sorted alphabetically":<bool>.

2) "the input is not rejected":<bool>.

Stimulus Response Components:

1) <search request>:<request message>.

2) <search response>:<response message>.

Local Predicates :

1) The response times for the I/O transactions described in this specification unit shall be in Class <integer> is a predicate.

**Performance**:

1) The response times for the I/O transactions described in this specification unit shall be in Class 2.

The "Local Predicates" subsection is used to introduce local predicates [54].

The formalization process involves determining error checks which should be supported by the formal version of the notation. This could include parsing errors: type checking errors and undeclared types, stimulus or response errors. However, some errors might

not be easily detected automatically, e.g., missing requirements. The syntax of the semi-formalized SRRS can be described in BNF and the semantics described with a higher order logic [54, 55] .

The SRRS method has only been applied to the Threads-Capabilities semi-formal notation, hence, its applicability to different semi-formal notations cannot be determined [55]. Furthermore, its stimuli and response concepts make the method better suited for functional requirements specifications. It may be used for non-functional requirements which can be described in terms of system functions, i.e., system responses to stimuli, however, this is not applicable to all stakeholder goals.

#### 3.3.2.2 Object-Oriented Natural Language Requirement Specification

The notation used in the Object-Oriented Natural Language Requirement Specification (OONLRS) is based on an extension of Two-Level Grammar (TLG) with object orientation [48, 47]. TLG was developed as a specification language for programming language syntax and semantics. TLG consists of two grammars which define the set of type domains and the set of function definitions operating on the domains [47].

OONLRS is aimed at deriving formal specifications with natural language-like notations that can be used by domain experts, designers and implementors. The TLG specifications can be translated to the vocabulary of a software design tool, e.g., an object-oriented design tool based on UML. However, there is no current support for automated update of the TLG specification from modifications made from design tools. Design tools could potentially generate code from the system design [48].

Type declarations are used for defining domain of functions, and they support strong typing of identifiers used in function definitions. The function definitions represent the operational part of the specification.

Domain declarations are of the form:

**Identifier-1, Identifier-2, ..., Identifier-m ::**

**data-object-1; data-object-2; ..., data-object-n.**

This denotes that the type for Identifier-1, Identifier-2, ..., Identifier-m comprises data-object-1, data-object-2, ..., data-object-n. Each data-object-i is a combination of domain identifiers (capitalized), singleton data objects, and data objects. For example, the type "Person" could be defined as: **Person :: first name String initial Character last name String**

Function definitions are of the following form:

**function-predicate.**

**function-predicate: sub-function-1,sub-function-2, ...,sub-function-n.**

where n$\geq$1 and each sub-function is a function predicate defined within the scope of the TLG specification. Function definitions could initially be given without precise domain but the method allows the user to iteratively specify the domain as well as the operations on the domain. Functions can be written at different levels of abstraction ranging from natural text to formal rule. This is illustrated below:

| | |
|---|---|
| Natural form | **Expensive parts are parts with an imported base part and cost more than $100** |
| Semi formal | **Expensive part: part with an imported base part and cost more than $100** |
| Formal rule for "Part" objects | **expensive : BasePart imported, Cost $>$ 100** |

The object oriented notion is supported by class declarations which encapsulate domain declarations and function definitions [48]. Classes are defined using the following syntax:

**class Identifier-1**

**[extends Identifier-2, Identifier-3,..., Identifier-n].**

{**instance variables and method declarations**}

**end class [Identifier-1].**

OONLRS has been used for simple natural language requirements and its object-orientation features are mostly suitable for operational requirements. Hence, it is better suited for requirements expressed as system functions. This does not make it an appropriate method

for non-functional requirements or stakeholder goals.

Adoption of translations from semi-formal or structured notations to formal representations depends on the effort (or cost) required and the benefits. The effort will be high for confidentiality and privacy requirements, which are usually specified in natural language and need to be translated to a semi-formal notation prior to the formalization process. Therefore, alternative methods which provide goal/requirement extraction and representation directly from source documents would be more viable.

### 3.3.3 Summary - Formalizing natural language requirements

The methods described in Sections 3.3.1 and 3.3.2 assist in the important process of extracting requirements from natural language sources. However, there are current limitations, i.e., lack of/minimal traceability link between sources and extracted representation, limitation to requirements expressing system functions and lack of tool support. In addition, they do not address the observations of supporting representations and analysis with default requirements, exceptions, as well as specifying stakeholders who may specify these exceptions (cf. Section 1.2.2).

## 3.4 Summary

Table 3.3 gives a summary of the methods described in this chapter. For each method, a *Yes* value in the corresponding column indicates that the method supports defaults, exceptions or extraction of representations from natural language sources. Otherwise a *No* value is specified. The table also indicates if a method has tool support.

| Method | Formal analysis | Default representation | Exception | Extraction from natural language sources | Tool support |
|---|---|---|---|---|---|
| UML methods (Section 3.1) | No | No | No | No | No |
| GBRAM (Section 3.2.1) | No | No | No | No | No |
| KAOS (Section 3.2.2) | Yes | No | No | No | Yes (no analysis [56]) |
| Secure Tropos (Section 3.2.4.1) | Yes (violations based on trust, ownership and delegation) | No | No | No | Yes |
| RNLS and Semantic Parameterization (Section 3.3.1.1) | Yes | No | Yes (with priority assignment) | Yes | No |
| GDL (Section 3.3.1.2) | Yes (verification) | No | No | Yes | No |
| SRRS (Section 3.3.2.1) | No | No | No | Yes | No |
| OONLRS (Section 3.3.2.2) | No | No | No | Yes | No |

**Table 3.3.** *A summary of related methods*

The RNLS method provides exception handling through priority assignments which determine precedence relationships among extracted sentences. This approach has the drawback of requiring a complete or global set of requirements for correct priority assignment. This is challenging for requirements in a multi-jurisdictional context, where requirements from different jurisdictions need to analyzed. Similar to the other methods which developed techniques for extracting representations from natural language sources, there is no tool support and this limits adoption in practice, especially for large documents.

Although the Tropos/Secure Tropos method currently provides tool support for goal analysis, the tool does not support extraction of the method's concepts from natural language sources. Tropos/Secure Tropos models are created by the analyst using the tool's visual editor but there is no means to navigate or trace these to the sources.

The CREE method developed in this research and described in Chapter 4 is motivated by the need to provide a method which supports analysis with defaults. CREE-tool, which is developed to support the CREE method, is used to create model representations derived from natural language sources.

# Chapter 4

# Confidentiality Requirements Elicitation and Engineering

This chapter presents the confidentiality requirement engineering method CREE, developed in the research described in this dissertation.

CREE - Confidentiality Requirement Elicitation and Engineering, supports analysis of confidentiality requirements through stratified goal models. CREE's development is aimed at addressing observations in Section 1.2.2. Requirements from different stakeholders' perspectives are often specified as expected default cases. Additional requirements, which could be exceptions to the defaults can also be specified. In addition, requirements may have hierarchical relationships, which can result in requirement overlaps, priorities and exceptions (cf. Section 2.4, Section 3.3). These relationships make stratified analysis with a reasoning method which supports defaults and hierarchical properties applicable. Defeasible reasoning supports these features, and this informed the decision to adopt defeasible logic formalism (cf. Section 2.5.1) for the CREE method.

CREE analysis is complemented by goal modelling through semantic annotation of text from source documents. The annotated texts represent instances of concept which are used in the creation of goal models and subsequent analysis. Traceability between goal model elements and the text is established through the annotations, thereby addressing one of the issues with requirement extraction from natural language regulations [103].

A tool, CREE-tool, was developed to: i) support the semantic annotation of text for creating goal models, ii) provide searching and navigation between derived goal models

and the source document, iii) create consistent annotated terms, and iv) support formal analysis of the goal models.

The CREE method consists of the following four steps as depicted in Figure 4.1:

**1) Goal Annotation**: Goals and their relationships are identified in the natural language documents and annotated with concepts from an ontology (cf. Section 4.2.1). We call this ontology the CREE Meta Model (Section 4.1.1.3).

**2) Structural Analysis**: Structures of the derived goal models are analyzed with respect to their structural completeness (cf. Section 4.4.1).

**3) Terminological Sorting**: Annotated concepts are consolidated into consistent terminologies (cf. Section 4.3).

**4) Semantic Analysis**: Goal models are analyzed for semantic inconsistency (cf. Section 4.4.2).

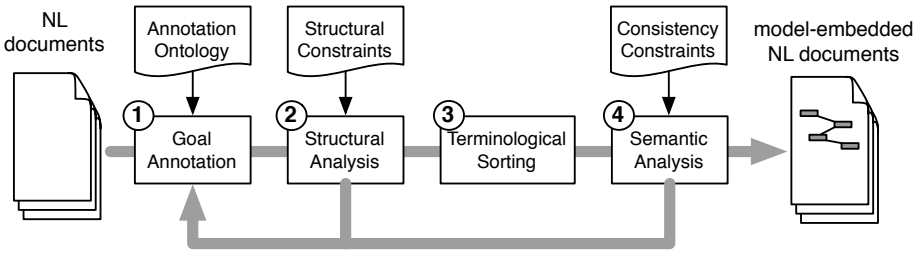


**Figure 4.1.** *Overview of the CREE process*

Iteration often occurs between the goal annotation and the analysis steps, as shown in Figure 4.1. The iteration occurs when reviews of goal annotations are needed for goal structures determined to be incomplete or inconsistent.

## 4.1 Developing CREE Meta Model

CREE meta model is developed by adopting concepts from goal-oriented methods. Models from these methods support representation of stakeholders and their relationships as well

as the stakeholders' goals. However, these methods have limitations in supporting default requirements and exceptions (Section 3.2).

Meta models of the KAOS and Tropos/Secure Tropos methods are used as inputs for defining the CREE meta model. The models are aligned to identify concepts relevant to represent confidentiality requirements. Elements representing stakeholders, information entities, and goals are adopted in the CREE meta model. In addition, a grounded theory qualitative study was conducted to explore additional properties that can be included in the model. Different sources, including natural language requirements, particularly from privacy regulations, were used in the qualitative study described in Section 4.1.1.

### 4.1.1 Confidentiality Concepts Identification

In order to effectively analyze confidentiality requirements it is critical to identify key factors that can be used in expressing the requirements. These factors as well as relationships among them not only provide a means for formal representation but also enable analysis of the requirements. For example, if a factor $X$ is deemed to be mandatory for a confidentiality requirement, analysis of the specified requirements can identify requirements without $X$.

The Grounded Theory qualitative research approach is adopted in the study to identify confidentiality requirement properties. Grounded Theory is used in developing a theory inductively and contextually from a corpus of data [60, 120, 30]. It involves data analysis to discover or label variables, which are referred to as categories, concepts or properties, and their interrelationships. The method involves constant comparison of new data to existing data or emerging theory. Data sources can be observations, interviews or literature, carefully selected, in order to extend and refine the emerging theory. The selection is based on *Theoretical Sampling* principle, where data is collected, coded and analyzed, and then a decision on next set of data to collect for enhancing the theory is made [72, 76, 104].

The study provided a method of investigating what constitute confidentiality requirements by identifying the properties from the ground up, using sources which give guidance on the handling of personal information. In addition, the experiences of practitioners were

also considered. Using this approach ensures that identified properties are important factors that are required to adequately specify confidentiality requirements.

#### 4.1.1.1 Data sources

The following types of sources were used in the study: privacy and confidentiality regulations/legislation, interviews with practitioners and literature review.

- Privacy and confidentiality regulations and guidelines.

  These sources represent governmental regulations aimed at protecting personal information, e.g., medical information. They address issues related to the collection, storage, usage and disclosure of personal information, and apply to organizations which handle personal information.

  The following legislations and guideline were used as sources:

  - PIPA

    Personal Information Protection Act (PIPA) is a provincial legislation which provides individuals in the province of British Columbia (B.C.), Canada with rights regarding their personal information in the custody or control of private B.C. organizations, including businesses, non-profit organizations and physicians' offices [4]. It requires organizations to be compliant with the rules on the collection, usage, and disclosure of personal information, as well as the right of data owners to access and correct personal information.

  - PIPEDA

    Similar to PIPA, PIPEDA (Personal Information Protection and Electronic Document Act) is a regulation for protection for personal information. However, it applies to federally regulated private sector as well as organizations involved in commercial activities within Canadian provinces [6].

  - EuroSOCAP

    EuroSOCAP is a European Commission sponsored project to address chal-

lenges expected between the easy flow of information in an information society, and ethical requirements of privacy and confidentiality. The aim is to build an accessible knowledge base and a set of standards and guidelines, which will inform professional practice as well as protect patients in the healthcare sector of the European community.

The guidelines will assist policy makers, professional organizations and the general public in understanding different issues of concern for the requirements of, and constraints on privacy and confidentiality in healthcare practice, in the evolving information society [8].

- Interviews

  Interviews were conducted with 10 physicians and health informaticians. The interviewees were asked the following set of questions:

  - *What is confidentiality?*

    This helps to understand respondents' views of what confidentiality entails in their domain.

  - *What are key confidentiality factors in practice?*

    This question is posed in order to get further details of the respondents' view of confidentiality requirements. The responses to this question also provide information useful in identifying the properties of confidentiality requirements.

  - *How would you specify confidentiality requirements?*

    This helps to identify how the different confidentiality requirement factors are used in expressing the requirements.

- Research Literature

  Publications which address privacy and confidentiality in electronic medical records and health/medical information systems [31, 49, 53, 113] were reviewed as sources for identifying appropriate confidentiality properties.

Some of the data sources used in this study are specific to medical/healthcare information domain (e.g., EuroSOCAP and interviews). This choice is based on the domain used as motivation in this dissertation (Section 1.2). As previously indicated, similar concern for confidentiality requirements can be identified in other domains, e.g., the financial domain. Although the generality of the concepts identified might be a concern, the implied bias towards the medical domain of the derived CREE meta model is balanced by more general concepts from the goal-oriented methods and from sources, such as PIPA and PIPEDA, which apply to multiple domains.

#### 4.1.1.2 Coding

The objective of the coding phase is to identify the key factors that can be used to describe confidentiality requirements from the data sources. Coding and analysis was done with the qualitative analysis tool, ATLAS.ti [3]. ATLAS.ti is a research workbench which provides features that support a systematic approach to analyzing unstructured data. It is used to manage, extract, compare, and explore pieces from large amounts of data. Its coding feature makes it suitable for conducting Grounded Theory research. All data sources were used as primary documents in ATLAS.ti for easy interpretation and management. The concepts were directly identified from the data sources as shown in Figure 4.2.

The next step involved specifying interrelationships among the concepts. Relationships "is a", "is property of", "is associated with" were specified. Figure 4.3 shows the interrelationships among the concepts.

- "is a" relationship indicates that a concept is a type of another concept. For example, "implicit consent", "explicit consent", "opt-out consent" and "opt-in consent" are types of "consent".
- "is property of" relationship indicates basic attributes used to define a concept. A "Confidentiality" concept can be described by the following attributes:
  - data of concern ("data to be protected"),

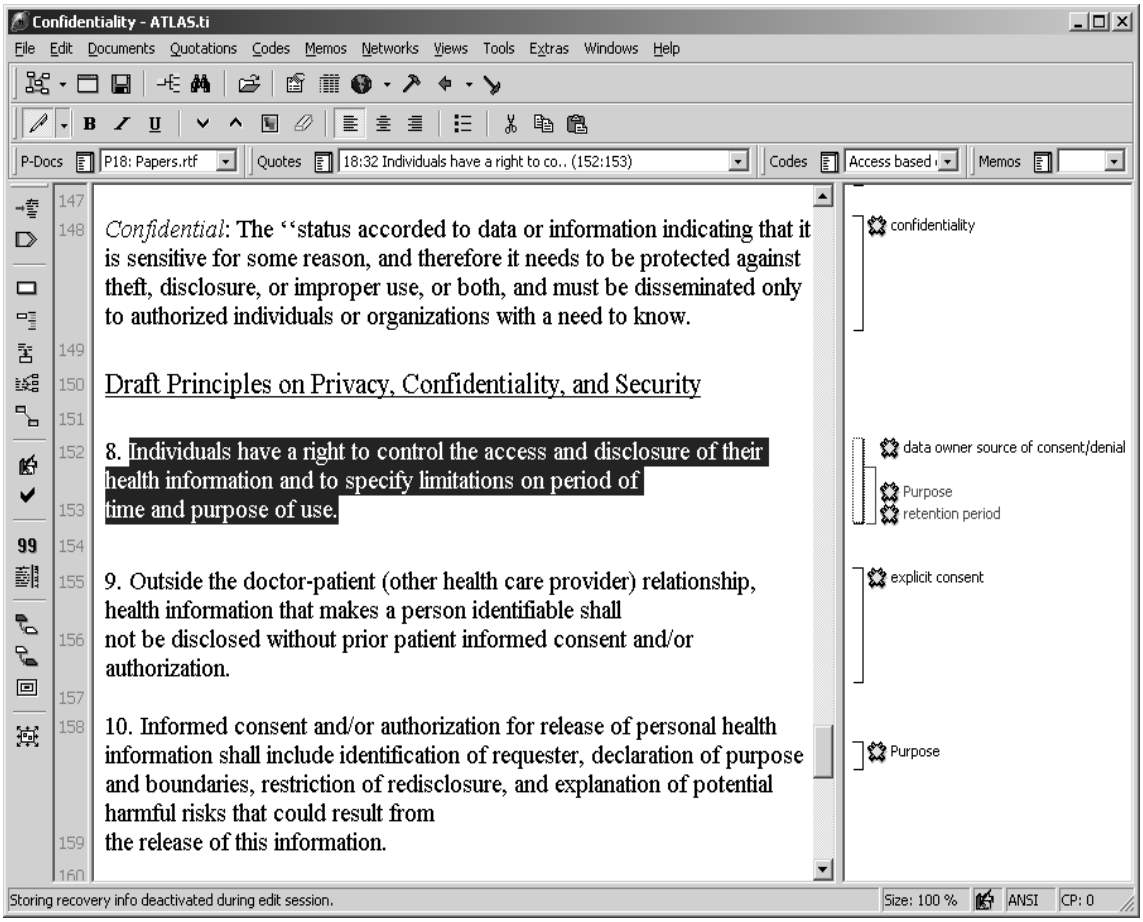


**Figure 4.2.** *Identifying Confidentiality Concepts*

  - roles of individuals accessing or using the data (“roles”),
  - granting or denial of access to the data (“consent” and “denial”)
  - individual with the confidentiality concern (“data owner source of consent/denial”),
  - an indication of possibility of exceptions or overrides (“need for exceptions”)

- “is associated with” describes associations between closely related concepts. These could be concepts that co-exist or where one is a consequent of the other. For example, in Figure 4.3 “disclosure to other parties” has an “is-associated-with” relationship with “consent”.

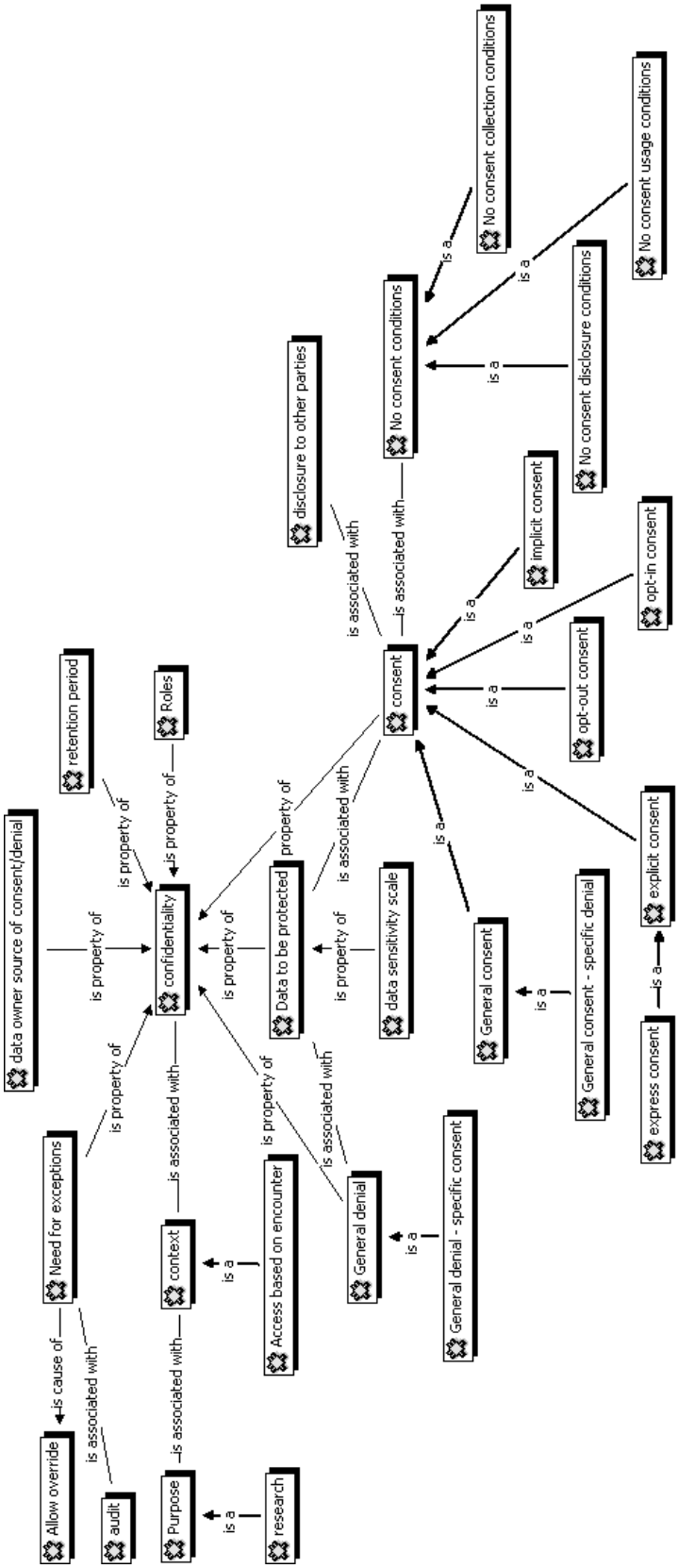


**Figure 4.3.** *Relationships of Confidentiality Concepts*

#### 4.1.1.3 Concepts of Confidentiality Requirements

Figure 4.4 shows concepts identified from the coding and analysis steps in Section 4.1.1.2.

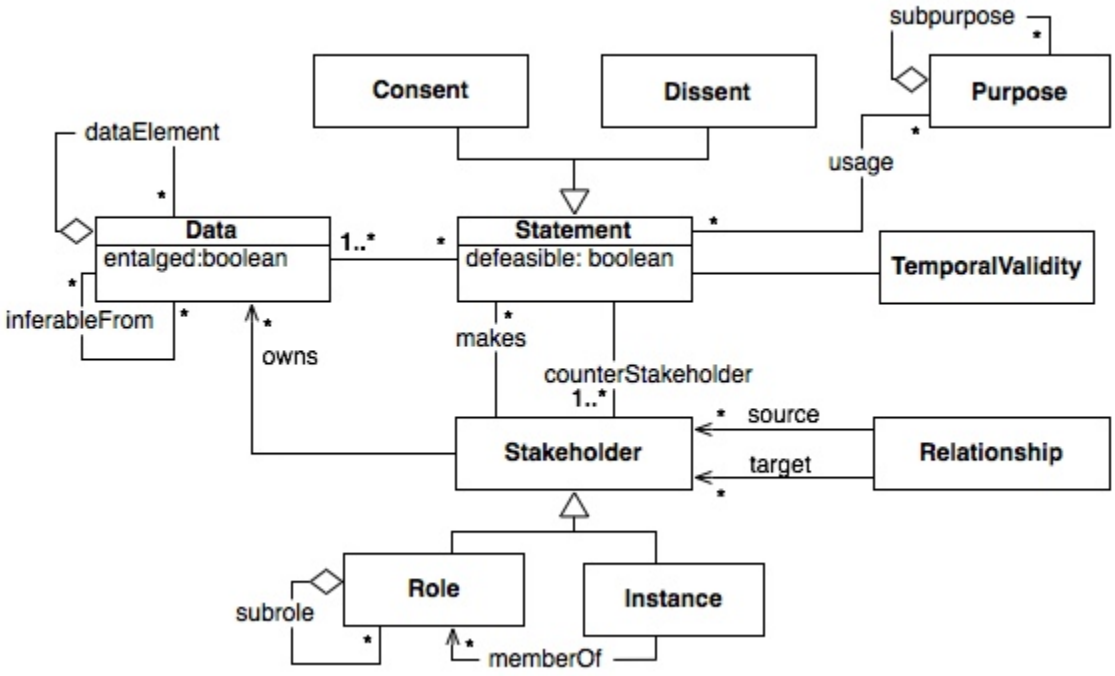


**Figure 4.4.** *Confidentiality requirement concepts*

The following are descriptions of the concepts:

- Stakeholder

  A "Roles" concept was identified during the coding process (cf. Figure 4.3). Different sources indicated individuals who are of interest in confidentiality concerns, e.g., "data owner source of consent/denial" or individuals who are targets of the confidentiality goals. These concepts are abstracted as *Stakeholder*.

  Stakeholders are identified as individuals (*Instance*) or groups (*Role*). Groups comprise instances or sub groups (sub roles). For example, the individual, "J. Doe",

can be a member of (*memberOf* association) role "Patients", while "Physicians" and "Nurses" are sub roles (subrole association) of "Clinicians". Relationships among stakeholders are important in expressing confidentiality requirements, (i.e., *Relationship* concept). This could be based on social or organizational relationships between stakeholders. For example, a relationship "Care team" could exist between patient "J. Doe" (source) and the roles "Family GP", "Referred Specialist" and "Home Nurse" (target).

- Data

  Essential to a confidentiality requirement is the data of concern. This is depicted in Figure 4.3 as "Data to be protected". The data concept is represented by a hierarchical structure, in which data elements can be further decomposed into sub elements (*dataElement*). Data elements composed of indistinguishable parts, i.e., the composing parts cannot be separated, are denoted with the *entangled* property. This usually occurs in semi-structured data. Requirements specified for entangled (semi-structured) data elements would apply to all the sub data elements.

  The concept of inference between data elements is represented by the *inferableFrom* relationship between data elements. If data element A is inferable from B, then knowledge about B can be used to deduce the existence of A. Inferrable association can be a rationale for additional requirements by the stakeholder in order to minimize undesired information leakage.

- Statement

  The *Statement* concept expresses the actual confidentiality requirement, and it could be a permission (*Consent*) or a denial (*Dissent*) of access to data. A statement is attributed to a stakeholder with the requirement (i.e., the *makes* association) and is directed to one or more stakeholders, called *counterstakeholder*. A consent gives the counterstakeholder the right to use the data, while a dissent statement indicates that the counterstakeholder should not be granted access to the data.

  Confidentiality requirements often have associated exceptions for cases where they

can be overridden. This notion is indicated in Figure 4.3 as "Need for exceptions". This is represented as the attribute *defeasible*. A defeasible statement can be overridden.

- Purpose

  Confidentiality requirements are also associated with purpose. Many legislation and laws mandate that data custodians identify the purpose for data collection, use and disclosure. In addition, the data owners have to be notified of the purpose of data usage. Purposes are usually related to features of a proposed system which can be described as functional requirements but could also potentially be other non-functional requirements.

- Temporal Validity

  Confidentiality goals are also associated with temporal validity, which indicates the length of time the statement is valid. This is illustrated by "retention period" in Figure 4.3.

### 4.1.2 CREE Meta Model

The CREE meta model, shown in Figure 4.5, is derived from the concepts identified in the study (cf. Section 4.1.1.3) and goal-oriented methods, KAOS and Tropos/Secure Tropos.

The concepts in the CREE meta model are described as follows:

- Actor - *Stakeholder* is represented as the *Actor* concept in the meta model. This is analogous to the Actor concept in Tropos. Role hierarchies and role membership are specified with *sub role* and *member of* associations respectively.

  Often, different terms are used for the same actor concept, e.g., physician and doctor are synonyms. This is specified with the *synonyms* association. An actor has a *qualSynonym* property, which is one of its synonyms or itself. The qualSynonym is the representative term for the set of synonyms.

- Goal

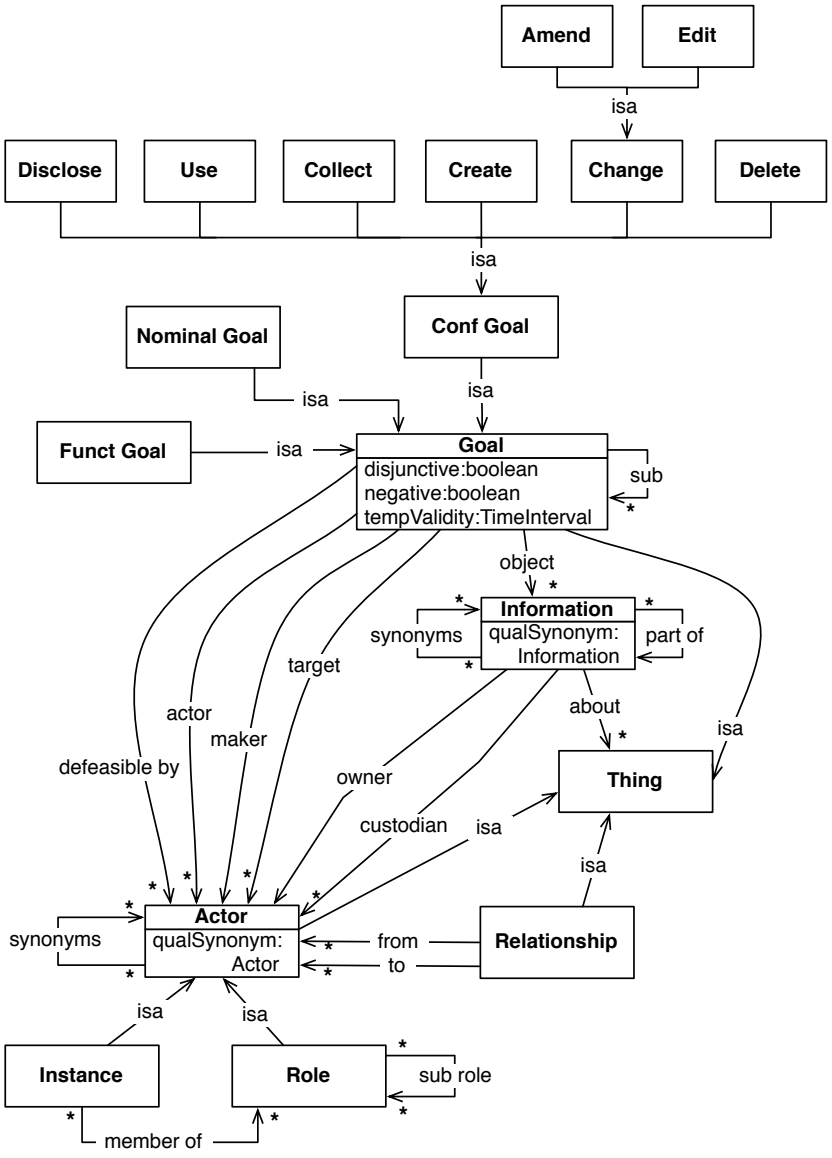


**Figure 4.5.** *CREE Meta Model*

A *Goal* represents an objective of a stakeholder. It might be functional, indicating an action the system or stakeholders should perform or non-functional. The goal concept represents the *Purpose* and *Statement* concept from the study, thus making it the key element in representing the requirements in CREE.

The Statement concept (with its sub types: Consent and Dissent) only expresses the

granting or denying of access to/use of data. It does not indicate the type of access or use. Specialized goal types in the CREE meta model, described below, can be used to specify the type of access. A goal's *negative* property indicates if it should be permitted or avoided. A TRUE value for the negative property means the goal should be avoided, otherwise the goal is permitted. Therefore, the negative property can be used to indicate granting or denial of access or use.

Goals can be decomposed into subgoals. The decomposition is represented with the *sub* association from a goal to its subgoals. This is analogous to goal decompositions in both KAOS and Tropos. CREE meta model's goal property, *disjunctive*, indicates if subgoals are an OR decomposition (disjunctive value is true) or an AND decomposition (disjunctive value is false). An OR decomposed goal is realized if any of its subgoals is realized, while an AND decomposed goal is realized if all of its subgoals are realized. CREE currently limits a goal to either AND or OR decomposition.

The actor whose requirement is represented by the goal is denoted by the *maker* association (analogous to goal owner in Tropos), while actors who should realize the goal are related to the goal by the *actor* association. Goals could also have actors as *target* (the counterstakeholder property), these are actors who the goal is directed at. The *object* of a goal is the information entity of interest. For example, a requirement by a health care authority that health care organizations can disclose anonymized personal health information (PHI) to researchers can be represented as a goal which has the authority as the maker, the health care organization as the actor, researchers as target and the anonymized PHI as the object. For concise representation, a subgoal assumes the values of the maker, actor, target and object properties of its parent goal, if the subgoal's corresponding properties are not specified.

The *defeasible* property identified in the study is represented in the CREE meta model as *defeasible by* association. The association allows representation of restriction of stakeholders who can specify exceptions to default goals. This improves the expressivity of CREE by ensuring that exceptions can be attributed to specific actors.

For example, exceptions to a goal, can be attributed to a legal requirement, therefore the goal is assigned a defeasible by value "law". KAOS and Tropos do not support the concept of default goals. A sub role or member of a role which is specified as value of "defeasible by" can also have overriding goals.

Property *tempValidity* indicates the temporal validity of a goal.

- Conf Goal

  *Conf Goal* represents a confidentiality goal. We distinguish between the following types of Conf Goal:

  a) *Use* - a goal for information handling or access. The actor of a Use goal processes the information.

  b) *Disclose* - a goal for information disclosure. Information is disclosed by the actor to the target of the goal.

  c) *Collect* - a goal for information gathering. Information is collected by the actor from the target of the goal.

  d) *Create* - a goal for information creation.

  e) *Change* - a goal for information modification by update or addition.

  f) *Amend* - a goal for information modification by addition to existing information.

  g) *Edit* - a goal for information modification by editing existing information.

  h) *Delete* - a goal for information deletion.

- Funct Goal

  *Funct Goal* is a task that actors or the system are expected to perform. It is analogous to Plan in Tropos. Details of these requirements are usually provided as functional requirements.

- Nominal Goal

  A Nominal goal represents a section or text fragment of the natural language document which contains a set of requirements. The nominal goal comprises goals which model the requirements in the section.

- Information - *Data* concept from the study is represented by *Information* in the meta model. It corresponds to the Object concept in KAOS and the Resource concept in Tropos. Hierarchical decomposition into sub parts is represented by the *part of* association. Information can be related to other concepts in the meta model and this is depicted by the *about* property, e.g., an audit log could be about a patient health information (PHI). Information can often be held by an actor (*custodian*) who does not have to be the *owner*. Similar to the Actor concept, corresponding *synonyms* and *qualSynonyms* properties are defined for information elements.

  While the data property "entangled" and data association "inferableFrom" are considered important they are not included in the CREE meta model because they are not explicitly required in the different confidentiality requirements studied. This also simplifies the current meta model.

- Relationship - the *Relationship* properties *source* and *target* from the study are renamed to *from* and *to* respectively.

- Thing - an abstract concept, it represents the different concept types in the meta model.

### 4.1.3 Goals with CREE meta model

The following examples illustrate requirements representation with CREE meta model elements. The examples also illustrate goal stratification based on information and actor properties.

- Information

  Table 4.1 shows two requirements: r1.1, which specifies that in general a hospital does not allow medical office assistants (MOA) to handle patients' personal health information (PHI). This is a default requirement as indicated by the defeasible by property. r1.2 specifies that MOA could however use demographic data section (e.g., patient name, address) of the PHI. r1.2 is an exception to r1.1 because the information

property for r1.2 is part of the information property of r1.1.

| *id* | *goal type* | *negative* | *maker* | *actor* | *Information* | *defeasible by* |
|---|---|---|---|---|---|---|
| r1.1 | use | true | Hospital | MOA | PHI | Hospital |
| r1.2 | use | false | Hospital | MOA | Demographic data | - |

**Table 4.1.** *Stratification with Information property*

- Actor

  Stratification also occurs with actors: these could be actors or targets of a goal. The actor/target for the stratified goal is either a sub role or member of the role of the higher level goal.

  In Table 4.2 the actor for r2.2, "Physician", is a sub role of "Staff", which is the actor for r2.1. Therefore, r2.2 is a stratification of r2.1 based on the actor property.

| *id* | *goal type* | *negative* | *maker* | *actor* | *Information* | *defeasible by* |
|---|---|---|---|---|---|---|
| r2.1 | use | true | Hospital | Staff | PHI | Hospital |
| r2.2 | use | false | Hospital | Physician | PHI | - |

**Table 4.2.** *Stratification with Stakeholder property*

## 4.2 Confidentiality Goal Annotation

CREE analysis is complemented by a process to create confidentiality goal models from natural language sources, such as regulations. The goal models are used for analysis of the confidentiality requirements. RNLS statements (cf. Section 3.3.1.1) could be used to extract structured statements, e.g., the translation of Canada Infoway's Privacy and Security requirements in Appendix A are provided in Appendix B. Although the RNLS translation

produces simple sentences, the challenges (cf. Section 3.3.1.1) necessitated the need for an alternative approach.

### 4.2.1 Semantic Annotation

A semantic ontology consists of concepts which represent a perspective and understanding of a domain and knowledge of the domain [98]. CREE goal models are created through semantic annotations of source documents using CREE meta model as a semantic ontology.

The annotation process is similar to the method described in Section 3.3.1.3, which supports model creation with hierarchical terms from source documents and provides traceability between the source and the model via the terms. However, CREE annotation uses a semantic ontology for model creation. Source document fragments which are deemed as instances of concepts in the ontology are annotated to create model elements. The fragments are used as labels for the created instances of the concepts. Furthermore, attribute assignments are specified for the corresponding associations or properties of the CREE meta model concepts. Traceability between the created models and source document fragments is realized by keeping track of source document fragments (textual span) that corresponds to an instance of a model element.

The annotation method provides the following benefits:

- traceability between document source fragments and models

  Annotation of the source documents with meta model concepts provides a traceability link between the created models and fragments of the source document. This facilitates navigation and browsing of the source documents and models.

- easier modification of goal model

  Identified concepts and associations between concepts can be modified on further review and as changes are deemed necessary. Modification involves removal of existing annotations and/or associations followed by the addition of new annotations and/or associations.

- easier derivation of semantic parameterizations

  Using meta model concepts as annotations during model creation means semantic parameterizations of the CREE concepts are realized with a single step in contrast to the multiple steps used with RNLS. In the RNLS method, semantic models are derived by first extracting restricted sentences and creating models from these sentences.

*CREE-tool* is developed to support the CREE method and it is implemented as a plug-in for Protégé. Protégé is an open source ontology editor and knowledge-base framework [16, 65]. In addition to supporting text annotation for creating goal models, CREE-tool enables navigation between models and the source documents. Details of the CREE-tool implementation and its usage for annotation of natural language sources are provided in Section 4.2.2.

### 4.2.2 CREE-tool

CREE-tool is an interactive graphical tool which supports source document annotation to create goal models, and analysis of the models. It is implemented as a plug-in for Protégé. CREE-tool's text annotation functionality extends an existing Protégé plug-in called Knowtator [12, 102]. CREE-tool also provides visual representation of the goal models by integrating features of Protégé's Ontoviz plug-in [14].

CREE-tool improves the goal representation process by:

- Creating CREE goal models directly from the annotations.
- Avoiding syntactic and typographic errors with semantic entities identified from source documents.
- Providing information on already annotated instances of semantic concepts.
- Browsing goal models and/or source documents.
- Navigating between created goal models and the corresponding source document fragments.

#### 4.2.2.1 Protégé and Knowtator

Protégé is an open source ontology editor and knowledge base framework. It is developed in Java and provides an extensible plug-in environment for modelling and development. There are two main methods of modelling ontologies in Protégé: Owl and Frames [16].

Protégé-OWL editor supports the Web Ontology Language (OWL), a W3C standard ontology language for promoting the semantic web [20]. Protégé-Frames editor has a user interface and knowledge server to support users in creating and storing frame-based domain ontology, customizing data entry forms, and entering instance data. It implements a knowledge model which is compatible with the Open Knowledge Base Connectivity (OKBC) protocol [19, 99]. In this model, an ontology consists of:

- classes which are organized in a hierarchy to represent a domain's concepts
- slots which are associated to classes and describe the properties and relationships of the classes
- instances of the classes - individual occurrences of the concepts that hold specific values for their properties

The Protégé-Frames model is used to represent CREE requirement goal models. CREE meta model concepts are denoted as Protégé classes, concept attributes and associations are denoted as Protégé slots, and CREE concept instances are represented by corresponding Protégé class instances.

CREE-tool's annotation feature is based on the Protégé plug-in Knowtator [12, 102]. Knowtator is designed as a text annotation plug-in and is used as a general purpose annotation tool to build annotated corpora. The corpora can be used in evaluating or training natural language processing systems [102]. Knowtator uses Protégé's ontology representation capabilities to define annotation schemas, which are subsequently used for the text annotation. CREE-tool uses the CREE meta model as its annotation schema.

Figure 4.6 shows the elements of CREE-tool's annotation schema in Protégé's Class Browser pane. The annotation schema is depicted as a class hierarchy using a tree structure.

Each node of the tree is a concept from the CREE meta model, while sub nodes represent subsumed concepts. The Class Browser also shows standard classes, which are part of the Protégé framework, and Knowtator's support classes. The support classes hold meta information such as the annotator, the path to the source documents, color assignments for annotation class types, information associating each annotated text to a class instance and its corresponding slot values.

Also shown in Figure 4.6 is the Class Editor pane (the pane on the right), which is used for defining/modifying slots (attributes) of the selected class in the Class Browser.

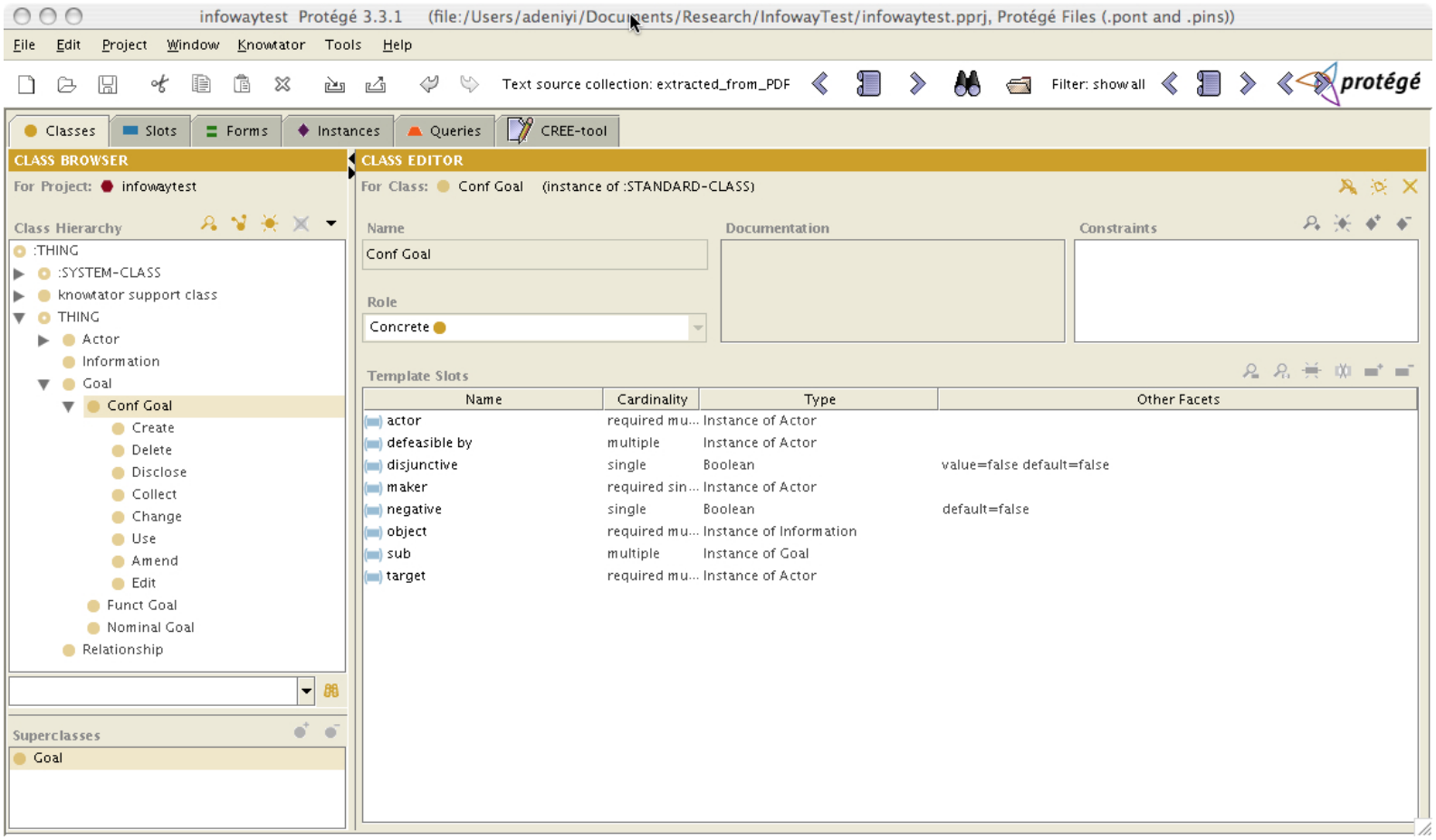


**Figure 4.6.** *Annotation schema (CREE meta model) definition in Protégé*

Knowtator could use Protégé's internal ontology model to create instances of the annotation schema classes. However, the current Knowtator implementation is limited to only creating instances of its support classes. CREE-tool creates instances of annotation schema classes from the text annotations. The instances are created from Knowtator's support class, *knowtator mention*, which holds information about an annotation (including the text span, the selected text, text source), the associated annotation class, and the slot (attribute) values. CREE-tool uses a *labeling scheme* for created instances from the annotated text by attaching a suffix, "*_n*" (where n is an integer starting from 0) for identical annotated texts.

#### 4.2.2.2 Model Visualization

CREE-tool provides an automatic visual representation of the goal models created from source document annotation. The visual notation, which consists of nodes for CREE meta model concepts and labeled edges for associations between concepts, provides an immediate feedback to the annotator. It can also be used as a means of communication with different stakeholders, because it represents the goals in a less esoteric format relative to the formal logic notation (Section 4.4.4.1).

CREE-tool visualization is realized by integrating features of Protégé's Ontoviz plug-in [14]. Ontoviz is based on Graphviz, an open source graph visualization program [9]. Graphviz provides features for creating diagrams, including options for colors, fonts, line styles, shape types and labels. It takes an input, which specifies the options for the nodes and the edges. The *dot* file format can be used to specify the elements of the input. The dot format is used to create hierarchical or layered drawings of directed graphs using a layout algorithm, which creates edges from top to bottom or left to right while attempting to minimize edge crossings [9]. CREE goal models, represented in Protégé as instances of the annotation schema classes, are specified as a dot input for Graphviz.

Node shapes for CREE meta model concepts are specified in a CREE-tool configuration file. Default shapes and edge notations are shown in Figure 4.7.

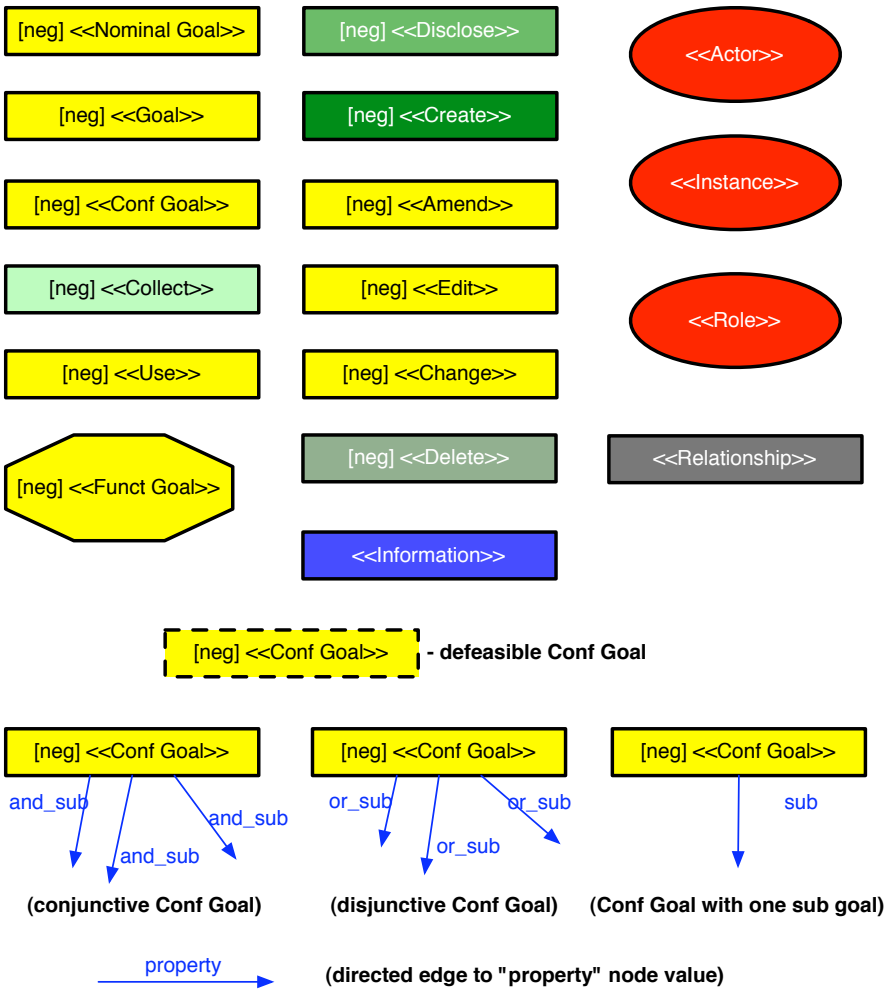


**Figure 4.7.** *CREE visualization nodes and edges*

Each node type has a stereotype-like label indicating its CREE meta model type. A *neg* label is used to denote negative goals while defeasible CREE requirement goals have nodes with a dashed outline.

Edges are directed to nodes which are values for the respective properties represented by the edges. An edge has a label which is the name of the property; an exception is an edge for the *sub* property of a goal. A disjunctive (i.e., OR-decomposed) goal has edges to its subgoals labeled *or_sub* while a conjunctive (i.e., AND-decomposed) goal has edges to its subgoals labeled *and_sub*. A goal with a single subgoal has an edge to its subgoal

labeled *sub*.

In order to reduce the verbosity of goal model representations, a property inheritance convention is used for subgoals. It is observed that subgoals often have the same maker, actor, target and object as their parent goals. Therefore, if a subgoal does not have values assigned to any of these properties it inherits the values from its parent (or parent's parent etc.).

CREE-tool allows the user to specify the depth of the visualized nodes. Traversal is done through slots (depicted as edges) of the instance the node represents, in both forward and backward directions to the node. Graphviz creates the appropriate graph representation, which is displayed in the visualization pane of CREE-tool. Figure 4.12 shows CREE-tool with the visualization pane on the bottom right.

#### 4.2.2.3 Goal Models from CREE-tool

The CREE-tool plug-in and its features are available through the CREE-tool tab in Protégé as shown in Figure 4.8.

The top left pane displays the annotation schema, while the bottom left pane lists instances of a selected annotation schema node from the top left pane. The middle pane shows a loaded source text document, where text selection and annotation is done. The top right pane displays details of a selected annotation, such as its class type and the values of its slots. The bottom right pane is used as the visualization pane for CREE-tool. Node representation of the selected annotation and its edges to slot value nodes are displayed in this pane. Additional features provided through the bottom right pane include functionality to regenerate instances of the annotation schema classes and the formal analysis of the goal models.

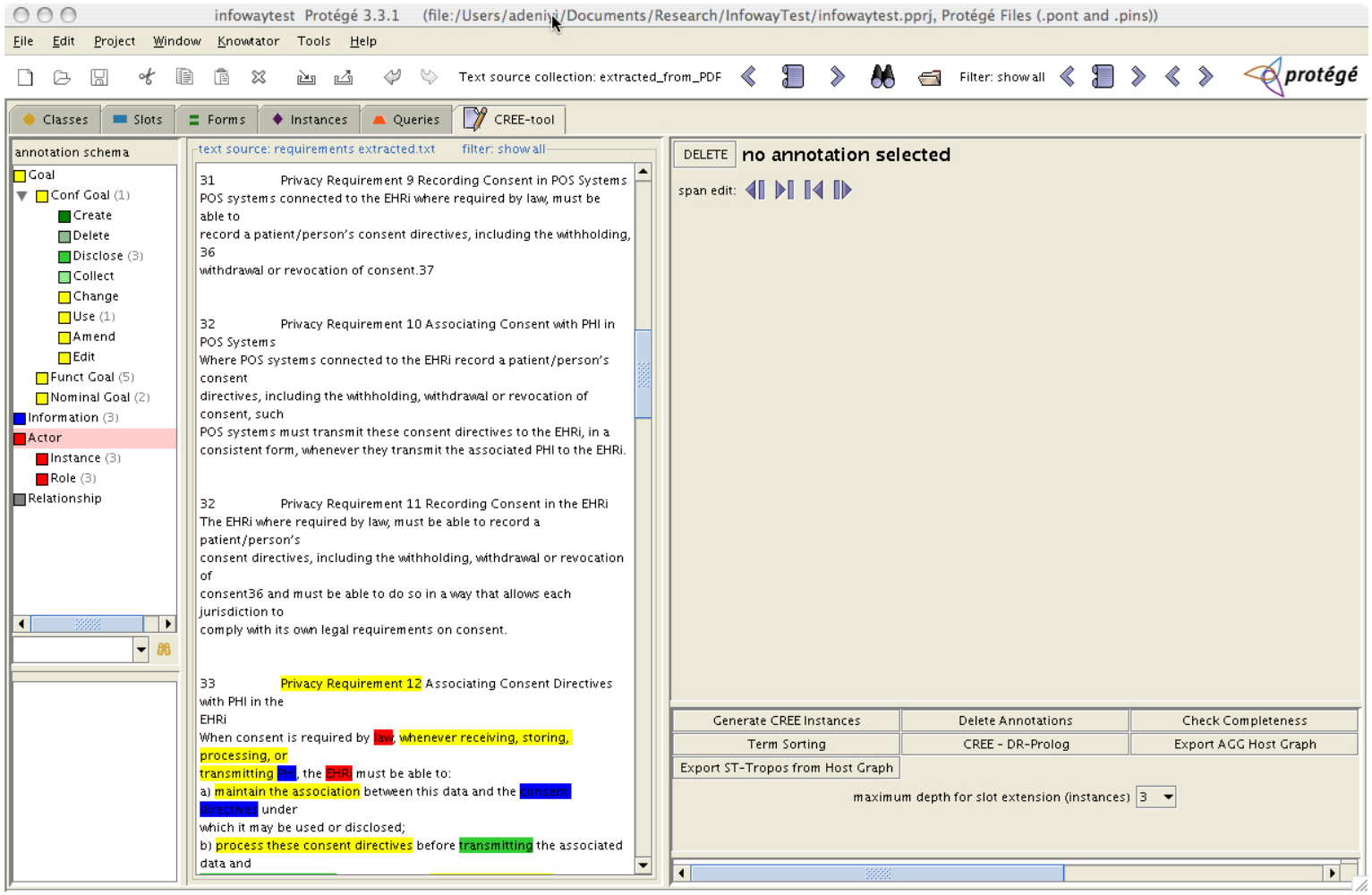


**Figure 4.8.** *Spanning the source document in CREE-tool*

### Annotating source document

Annotation is done by highlighting (spanning) text from the source document (displayed in the middle pane) and clicking on the appropriate element from the annotation schema. A context menu is displayed from which the annotator can select to create an instance of the concept element as shown in Figure 4.9.

Details of the annotated text and its corresponding annotation schema class are held by Knowtator support classes. These are subsequently used to create an instance of the corresponding annotation schema class.

### Slot assignment

Slot assignment for an annotation is realized by selecting the annotation and then clicking on the annotation which is to be assigned as a slot value. Clicking the second annotation displays a context sensitive menu which has a list of the various slot types the annotation can be assigned to. For example, in Figure 4.10, the *Instance*, "law", can be assigned to the *maker*, *actor*, *target* or *defeasible By* slot of the selected *Nominal Goal*, "Privacy Requirement 9 Recording Consent in POS Systems".

CREE-tool extends Knowtator by enabling assignments to slots which are inherited from a parent class of an annotation schema class. Knowtator restricts slot assignments to direct slots of the selected annotation's class.

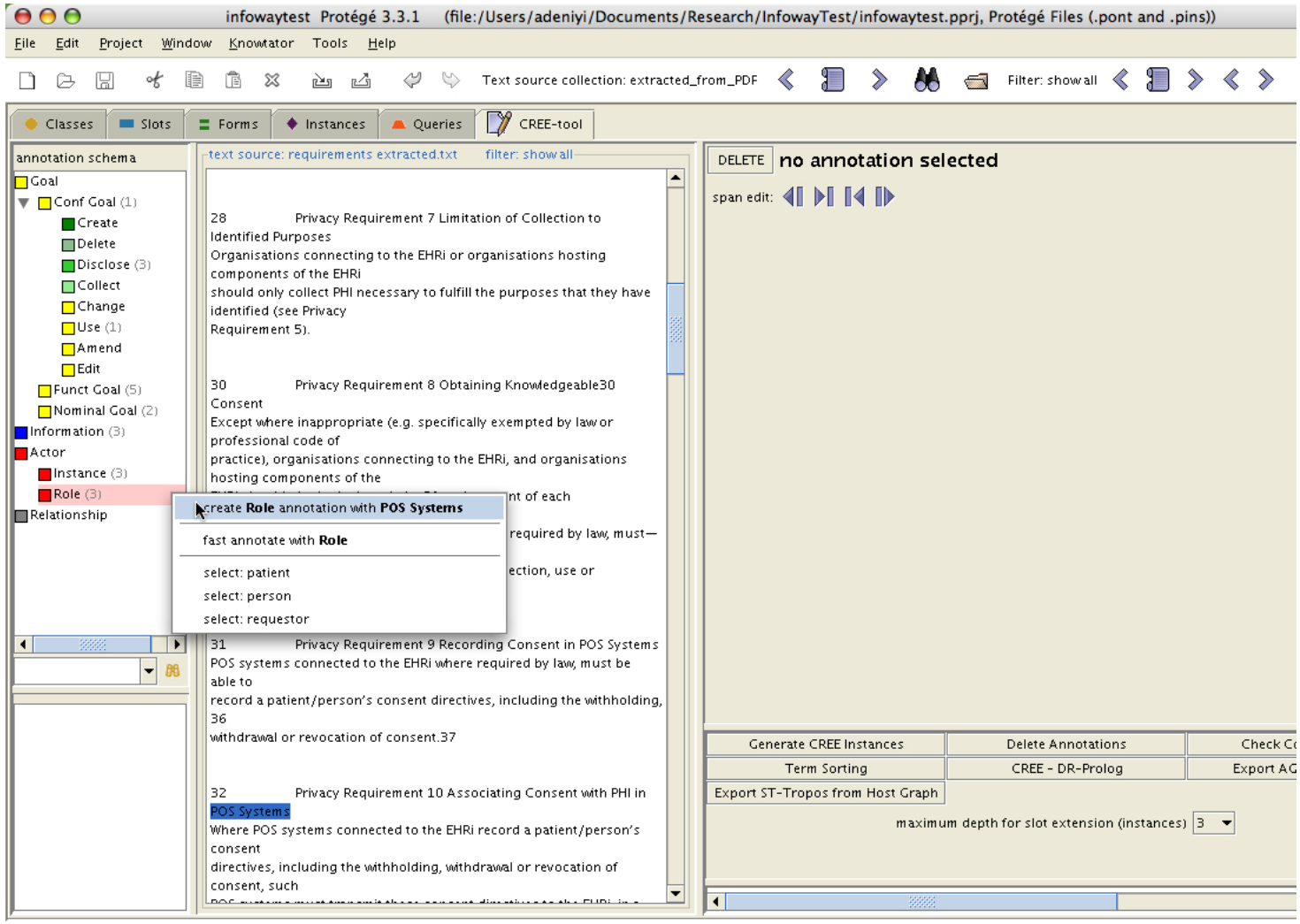


**Figure 4.9.** *Annotating text with CREE meta model concept*

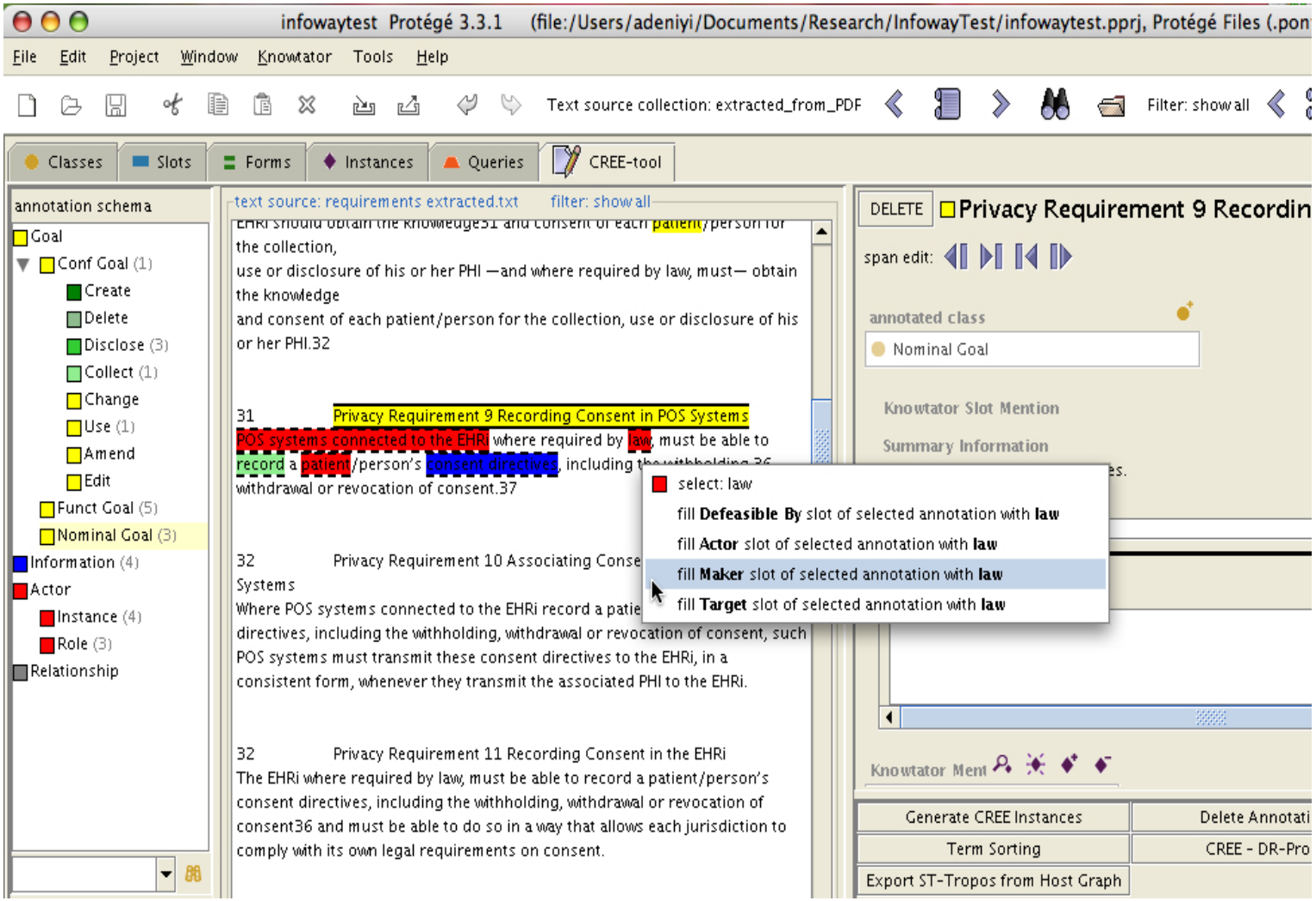


**Figure 4.10.** *Connecting annotations using slots*

### Visual notation

CREE-tool produces a visual representation of the instances of the annotation schema classes. A graph representation, as shown in Figure 4.11, is created starting from a selected annotation and using its slot values to traverse the other nodes of the graph.

The graph representation is displayed in CREE-tool's visualization pane, e.g., Figure 4.12 shows the visual representation of the selected *Nominal Goal* annotation, "Privacy Requirement 9 Recording Consent in POS Systems".

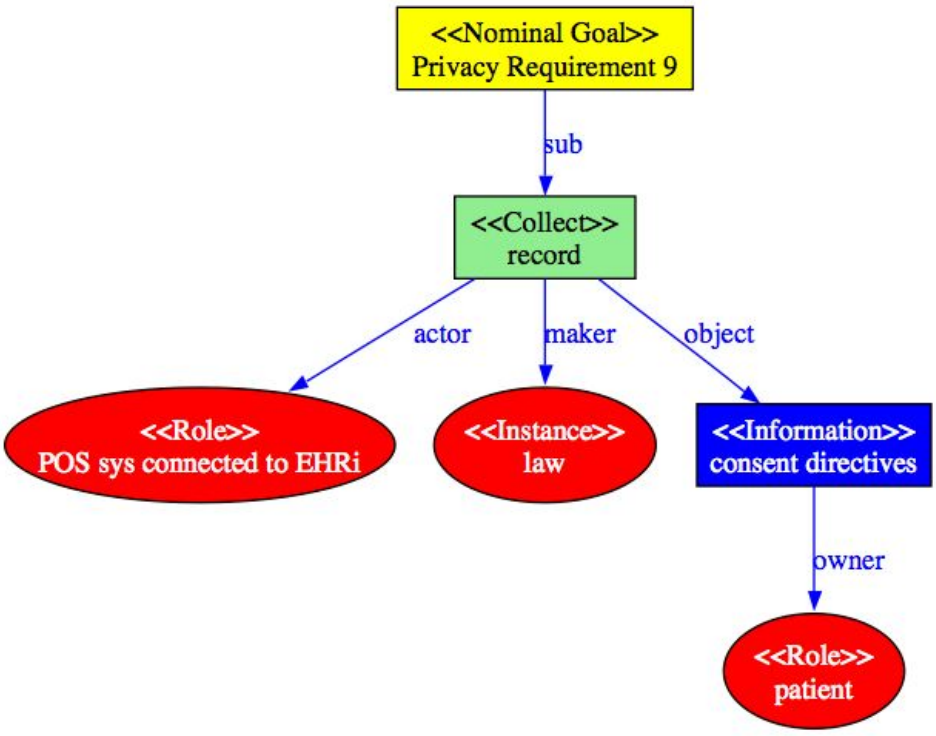


**Figure 4.11.** *Goal model from visualization pane in Figure 4.12*

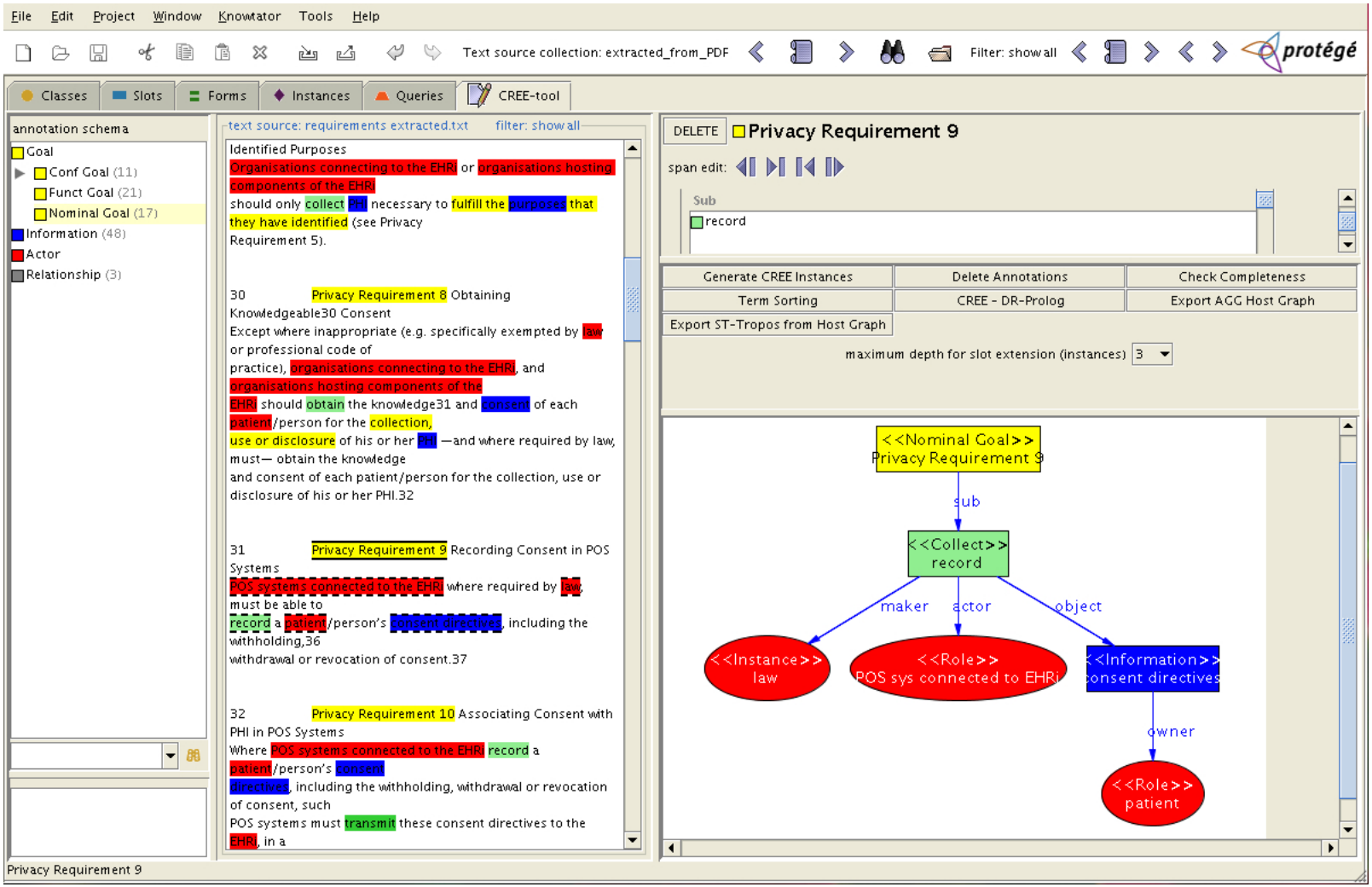


**Figure 4.12.** *CREE-tool showing visualization pane (bottom right)*

### 4.2.3 Traceability and Navigation in CREE-tool

CREE-tool provides navigation and browsing features for the annotator/analyst from both the source document and the created requirement goal model representation. Traceability links between these representations are derived from the following:

- Knowtator's annotation information, including the selected text, the associated annotation class and their corresponding slot values are stored in Protégé knowledge base as instances of Knowtator support class, *knowtator mention*. Protégé assigns each instance of a class a unique name of the format "[*projectname*]_Instance_[*n*]", where projectname is the project name and n is an integer value.
- CREE-tool creates the corresponding annotation schema instance using instances of knowtator mention class. As indicated on Page 95, CREE-tool employs a labeling scheme which assigns unique names to created annotation schema class instances. The CREE-tool labeling scheme uses labels which are more intuitive in comparison to Protégé unique names, and which can be used in the visual representation of the model elements. CREE-tool creates a persistent data structure, which maps unique Protégé names of knowtator mention instances to corresponding CREE meta model unique names.
- The visual representation of the goal model elements uses CREE-tool labels for the corresponding graph nodes.

Navigation and browsing between the text and visual representation is realized by retrieving appropriate graph node or text fragment using the traceability mappings described above. When an annotated fragment is selected from the middle text pane of CREE-tool, the appropriate graph node and its associated nodes are displayed in the visualization pane. Similarly, clicking on a node in the visualization pane selects and highlights the corresponding text fragment in the middle pane.

## 4.3 Terminological Sorting

This step of the CREE approach is aimed at compiling a consistent terminology of annotated concepts. The consistent set can be used in creating a data dictionary, which is identified as a key factor in extracting requirements from regulations specified in natural language (cf. Section 3.3). Terminological Sorting is an interactive activity in CREE-tool, and is realized by eliminating terminological conflicts due to homonyms and synonyms. CREE-tool also provides functionality to order concepts in the actor and information hierarchies (cf. the associations *member of, sub role* and *part of* in Figure 4.5).

Each entry in either hierarchy is one of the terms of a set of synonyms in the context of the requirements being analyzed, e.g., the terms "patient" and "client" may be considered synonymous in the context of the EHRi project (cf. Section 1.2.1). One of the synonyms is indicated as the *qualified synonym* term and it serves as the representative term for the set. In the previous example, "patient" can be indicated as the qualified synonym. Subsequent to the terminological sorting activity all homonyms must be eliminated among the qualified terms.

Figure 4.13 shows CREE-tool's terminology sorting window. It is launched on clicking the "Term Sorting" button in the bottom right pane of the tool. A user performs the sorting activity for annotated actor or information terms. The choice is made by selecting one of the radio buttons on the bottom left in the window There are 3 lists on the window:

- annotated terms list - the leftmost list. It shows the annotated actor or information terms.
- synonyms list - the middle list. It shows the synonyms of the selected term in the annotated terms list.
- qualified terms list - the rightmost list. It shows terms which have been designated as qualified synonyms.

Synonyms are added to an annotated term by selecting the term from the list of annotated terms and clicking the "add synonym" button. The terms to be added are selected from

a list presented to the user. The annotated term and its current synonyms are included in the set of synonyms for the new synonym. Terms are removed as synonyms for an annotated term by selecting the terms to be removed from the middle list of the window and clicking the "remove synonym" button. In addition, the terms removed are no longer synonyms for the remaining synonym terms.

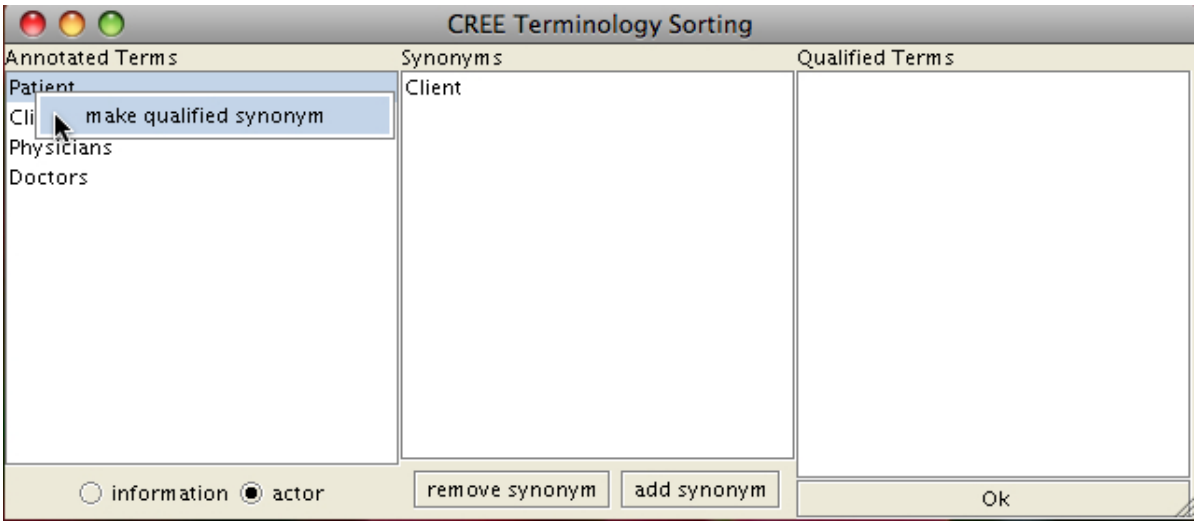


**Figure 4.13.** *CREE Terminology Sorting*

Figure 4.13 also shows the context menu used to make an annotated term a qualified synonym. On clicking the "make qualified synonym" menu item, the annotated term is set as the qualified synonym for itself and its synonyms. The qualified synonym is added to the list of qualified terms as shown in Figure 4.14.

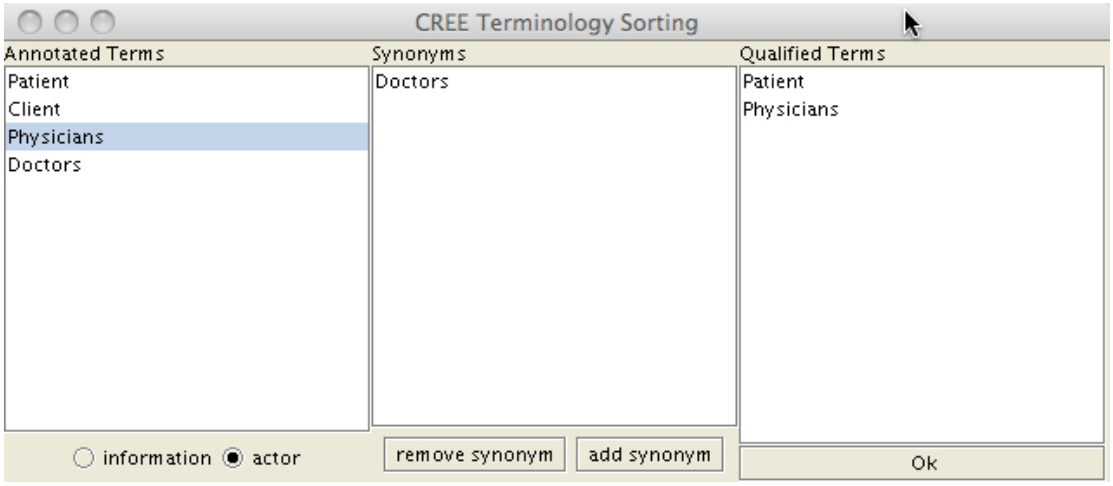


**Figure 4.14.** *CREE Terminology Sorting with qualified terms*

Terminology sorting activity consists of operations on CREE concepts. The operations, listed as 1 - 5 below, are defined using UML Object Constraint Language (OCL). The following OCL notations, (a) and (b), are used in the definitions.

a) "x.propertyname" - indicates the value of property, *propertyname*, of x. Types and properties are from the CREE meta model. For example, a.synonyms is the set of synonyms for a. Property, "type" is an OCL defined property which indicates the type for a variable.

b) @pre - indicates the value of a property prior to the operation.

1) Add synonym to a term

For a given term *b*, this operation adds another term, *a*, as a synonym of *b*. *a* also becomes a synonym to the current synonyms of b. Correspondingly, *b* and its previously existing synonyms are also added to the set of synonyms of *a* and a's existing synonyms.

**context** Actor :: addSynonym(a:Actor): OclVoid

**pre**: self.type = a.type

**post**: self.synonyms−>includes(a)

**post**: self.synonyms−>includesAll(a.synonyms@pre)

**post**: self.synonyms@pre−>forAll(t|t.synonyms−>includes(a))

**post**: self.synonyms@pre−>forAll(t|t.synonyms−>includesAll(a.synonyms@pre))

**post**: a.synonyms−>includes(self)

**post**: a.synonyms@pre−>forAll(t|t.synonyms−includes(self))

**post**: a.synonyms−>includesAll(self.synonyms@pre)

**post**: a.synonyms@pre−>forAll(t|t.synonyms−includesAll(self.synonyms@pre))

**context** Information :: addSynonym(a:Information): OclVoid

**post**: self.synonyms−>includes(a)

**post**: self.synonyms−>includesAll(a.synonyms@pre)

**post**: self.synonyms@pre−>forAll(t|t.synonyms−>includes(a))

**post**: self.synonyms@pre−>forAll(t|t.synonyms−>includesAll(a.synonyms@pre))

**post**: a.synonyms−>includes(self)

**post**: a.synonyms@pre−>forAll(t|t.synonyms−includes(self))

**post**: a.synonyms−>includesAll(self.synonyms@pre)

**post**: a.synonyms@pre−>forAll(t|t.synonyms−includesAll(self.synonyms@pre))

2) Remove synonym from a term

For a given term *b*, this operation removes another term, *a*, as a synonym. *a* is also removed as a synonym for the other synonyms of b. Correspondingly, *b* and its other existing synonyms are removed from the set of synonyms of *a*.

**context** Actor :: rmvSynonym(a:Actor): OclVoid

**post**: self.synonyms−>excludes(a)

**post**: self.synonyms@pre−>forAll(t|t.synonyms−>excludes(a))

**post**: a.synonyms−>excludes(self)

**post**: a.synonyms−>excludesAll(self.synonyms@pre)

**context** Information :: rmvSynonym(a:Information): OclVoid

**post**: self.synonyms−>excludes(a)

**post**: self.synonyms@pre−>forAll(t|t.synonyms−>excludes(a))

**post**: a.synonyms−>excludes(self)

**post**: a.synonyms−>excludesAll(self.synonyms@pre)

3) Set qualified synonym

This operation sets a term, *a*, from a set of synonyms as the *qualified synonym* for the set.

**context** Actor::setQualSynonym(a:Actor): OclVoid

**pre**: self.type = a.type

**pre**: self.synonyms−>includes(a) or self = a

**post**: self.qualSynonym = a

**post**: self.synonyms−>forAll(t|t.qualSynonym = a)

**context** Information::setQualSynonym(a:Information): OclVoid

**pre**: self.synonyms−>includes(a) or self = a

**post**: self.qualSynonym = a

**post**: self.synonyms−>forAll(t|t.qualSynonym = a)

4) Resolve a pronoun to its actual term

In natural language documents, pronouns, e.g., *it, he, them, its, this, that, who, that, which*, refer to previously mentioned terms to make sentences less cumbersome and repetitive. The term which a pronoun refers to, called the antecedent, should be used during the annotation process. If a pronoun is considered a synonym of a term, the pronoun should not be designated as a qualified synonym. This expressed as the following OCL invariants, where P is the set of pronouns.

**context** Actor inv

P−>forAll(t|t <> self.qualSynonym)

**context** Information inv

P−>forAll(t|t <> self.qualSynonym)

5) Make one term a child of another term in an hierarchy

This operation is used to create the actor and information hierarchies. Roles can be specified as sub roles of other roles, while Instances can be specified as members of a role. Similarly, composition of information entities can be specified.

**context** Role::addChildTerm(a:Actor): OclVoid
**pre**: a.oclIsTypeOf(Role) or a.oclIsTypeOf(Instance)
**post**: if (a.oclIsTypeOf(Role) then
a.subrole−>includes(self)
else
a.memberof−>includes(self)
endif

**context** Information::addChildTerm(a:Information): OclVoid
**post**: a.partof−>includes(self)

## 4.4 Analysis of CREE Models

CREE provides structural and semantic analysis of the confidentiality goal models. The analysis are discussed in Sections 4.4.1 and 4.4.2 below.

### 4.4.1 Structural Analysis

Structural analysis enables checks for omissions that could occur during the annotation process.

A goal with undefined maker, actor, target or object property is termed as incomplete (Definition 8). Inheritance of these properties by subgoals (Definition 4) allows concise representation of CREE goals. However, a goal property is undefined if none of its tran-

sitive parent goals nor itself has a value specified for the property. A property could be missing due to the following:

i) omission by analyst: an analyst could erroneously omit a property during annotation.

ii) ambiguity in source document: an analyst might not be able to identify an attribute during the annotation step or the attribute might be an assumption not explicitly indicated in the source document.

This is illustrated by the following excerpt from a requirement for an e-health initiative: "Foreign Disclosures

The Service Provider will immediately inform the Province if the Service Provider receives any order, directive, ruling, requirement, judgment, injunction, award or decree, decision, or other requirement issued pursuant to any Foreign Disclosure Laws, .... Upon receipt of a Disclosure Order, the Service Provider will not disclose any Patient Personal Information in response thereto ..."

This is annotated to create the representation shown in Figure 4.15

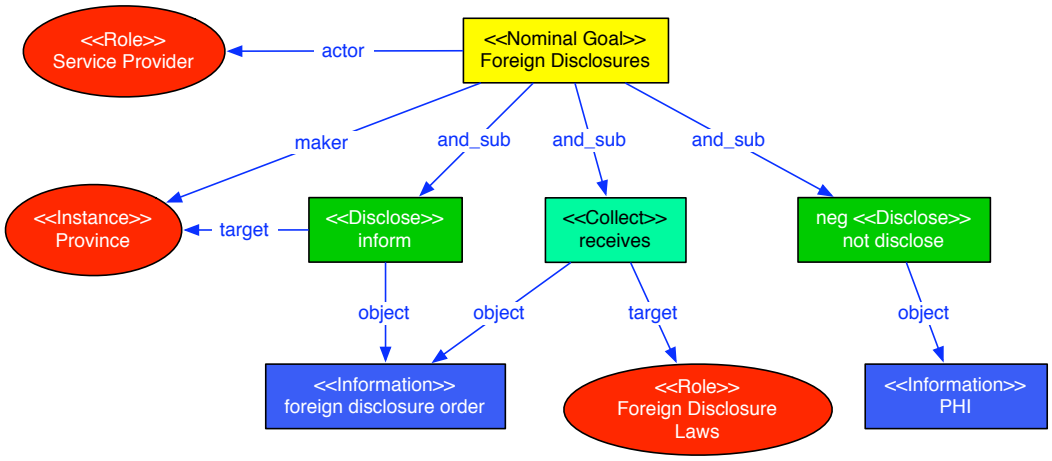


**Figure 4.15.** *Model with incompleteness*

Each of the goals has a *maker* property "Province", and an *actor* property, "Service Provider" (the subgoals inherit these properties based on the subgoal property inheritance convention). The Disclose ("inform") and Collect ("receives") goals also have *target* properties. However, the negative Disclose goal does not have a target property because further

clarification is needed to identify who the Service Provider should not disclose the personal information to. For example, an order or directive would indicate who the information needs to be disclosed to.

Structural analysis is based on goal constraints, which can be expressed as UML Object Constraint Language (OCL) invariants. An extended version of OCL proposed by Schürr [114] is used. Schürr introduces a transitive closure operator (*) for navigating over "zero or more" instances of a specified path. This operator enables a more concise definition of property subsumption by subgoals.

The expression below checks for the existence of *object, target,* and *actor* properties for each goal and enforces an acyclic goal structure. It requires that goals be either top level (nominal goals) or children of other goals, and that nominal goals have a *maker*. The *super* keyword denotes the *sub* association from a parent goal in the CREE meta model.

**context** Goal inv:
self.super*.object−>notEmpty() and
self.super*.target−>notEmpty() and
self.super*.actor−>notEmpty() and
not self.super.super*−>includes(self) and
self.super−>isEmpty() implies oclIsTypeOf(NominalGoal) and
oclIsTypeOf(NominalGoal) implies self.maker−>notEmpty()

The following expression checks for an owner for instances of Information; the super keyword denotes the *part of* association to a containing information element.

**context** Information inv:
self.super*.owner−>notEmpty()

Identifying structural incompleteness assists in correcting omissions in models. The incompleteness analysis is performed in CREE-tool by clicking the "Check Completeness" button on the right pane (cf. Figure 4.12). The incomplete goals are placed in different categories based on the missing properties, i.e., maker, actor, target and object. Analysts

can review incomplete goals and

i) assign the right property value in the case of an omission or

ii) contact domain experts for clarifications in the case of ambiguity which cannot be resolved using the source document.

### 4.4.2 Semantic Analysis

The following semantic analysis are provided.

i) Inferring requirement goals

CREE analysis provides reasoning which enables determining defeasibly inferred goals in the goal model. This is based on stratification and superiority relations among specified and derived goals. A stratification relationship exists between two goals if the respective goal types, as well as the actor, target and object properties have partial order relationships.

Figure 4.16 shows a negative Disclose goal, “not disclose” and a Disclose goal, “release” for personal health information (PHI), where the actor properties of the goals are assumed to be respectively partially ordered. The Disclose goal defeats the negative Disclose goal because they have a stratification relationship and the actor specified as the *maker* property of “release” i.e., “Province”, can specify exceptions for “not disclose” as depicted with the *defeasible by* property.

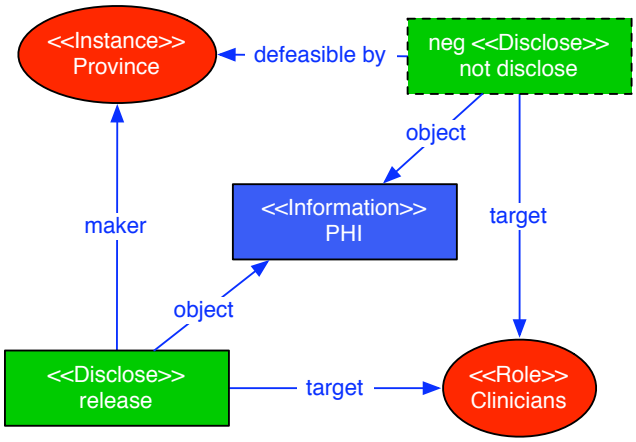


**Figure 4.16.** *Inference with defeating goals*

A goal annotation, termed a *specified* goal, for a role applies to members of the role and its sub roles. The applicable goals for the members are termed *derived* goals. However, stratified defeasible inference is relative to other goals which could prevent (defeat) the application to particular members or sub roles. Figure 4.17 extends the previous example (Figure 4.16) with the addition of the subrole "Nurses" to "Clinicians". The extension illustrates a scenario where derived goals are defeated. Let $g_1$ be the derived goal from "release' for nurses and $g_2$ be derived goal from "not disclose" for nurses. $g_1$ has a stratification relationship with $g_2$, however, $g_2$ is not concluded because a contradictory goal, i.e., $g_1$ is derived from specified goal, "release", which defeats "not disclose".

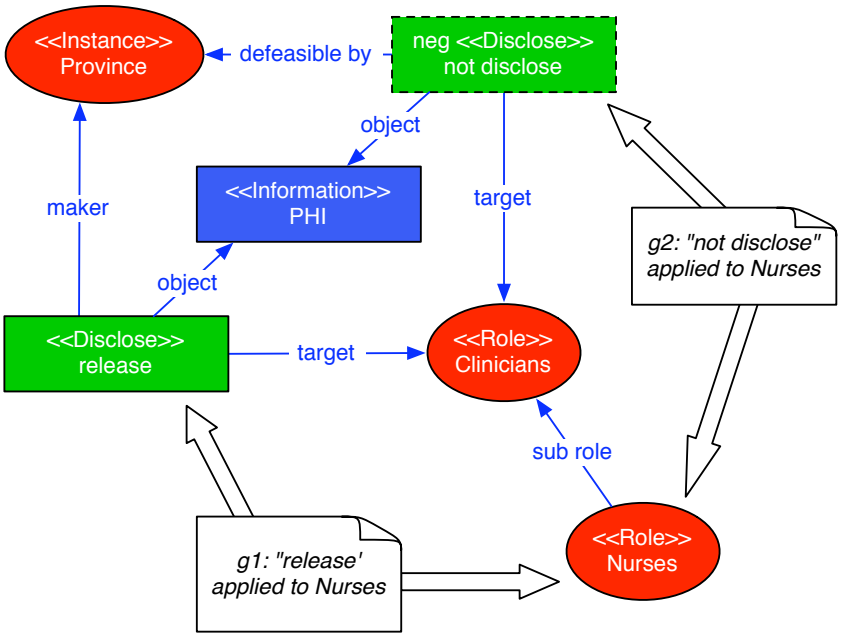


**Figure 4.17.** *Inference with derived goals*

Similarly from the model in Figure 4.18, "not disclose" is not concluded because a contrary applicable derived goal $g_1$ overrides it.

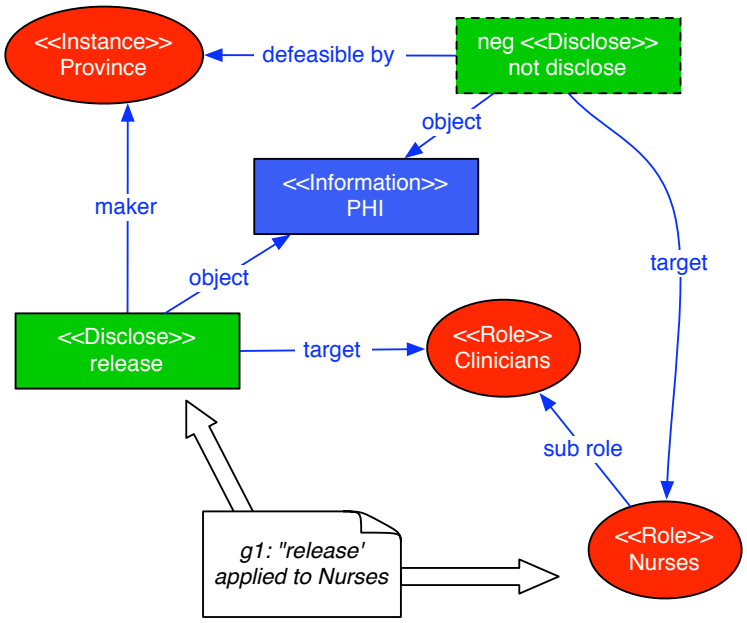


**Figure 4.18.** *Inference with specified and derived goals*

This analysis is formally realized with Definitions 9, 12, 13 and 14.

ii) Interference, Conflict and Exception identification

CREE goal models are analyzed to identify interference and conflicts among goals, and by extension determine existing inconsistencies from the original source document. Interference, conflict and exception identification are based on goal stratification, goal decomposition and permitted overrides.

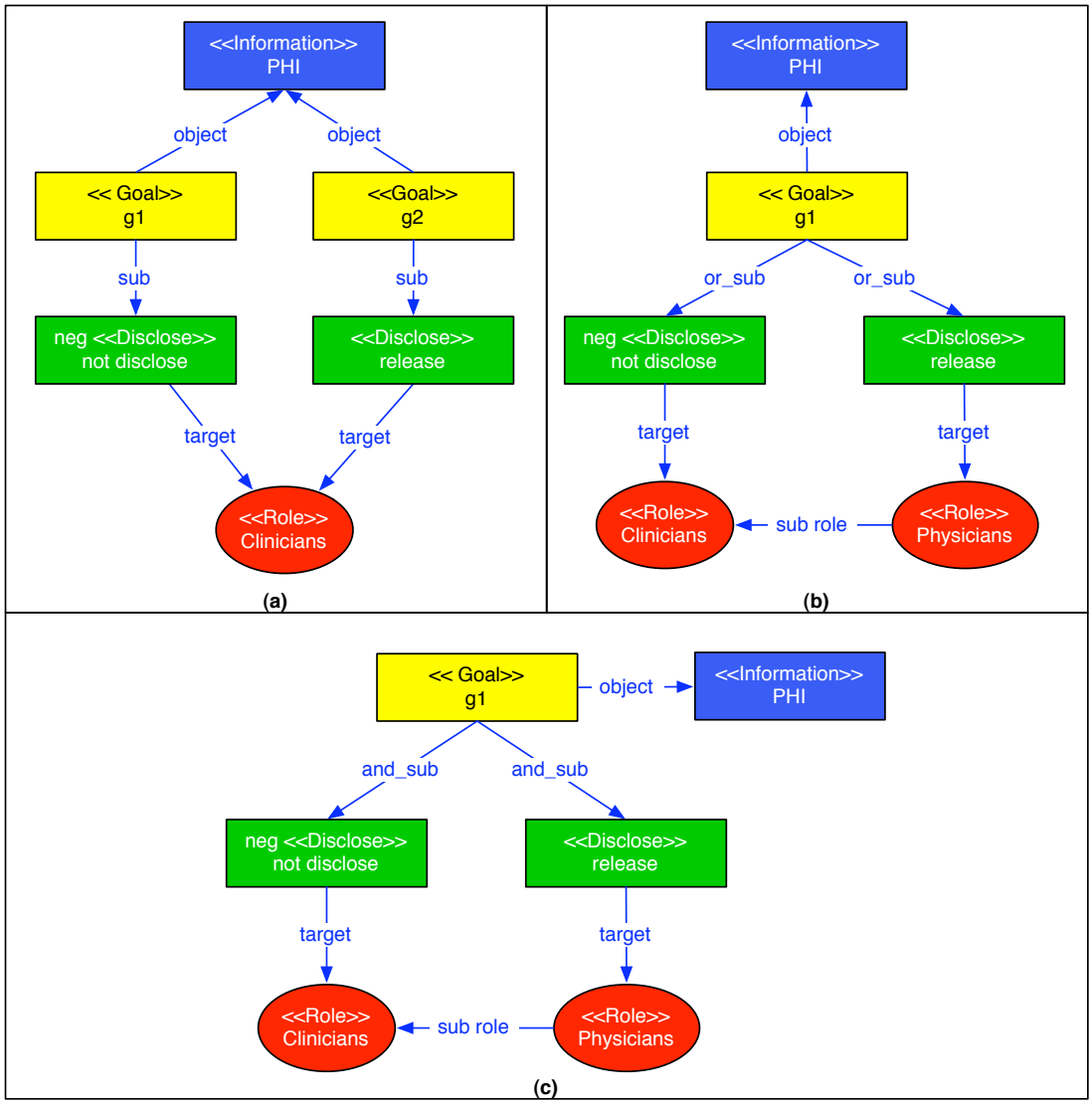


**Figure 4.19.** *Goal interferences and conflict*

Interferences occur when opposite goals have a stratification relationship. The Disclose goals in Figure 4.19 interfere. All the goals have an object property, "PHI" (Personal Health Information). For (a), goals $g_1$ and $g_2$ interfere because of the corresponding Disclose goals. In (b) and (c), goal $g_1$ is OR-decomposed and AND-decomposed respectively into interfering Disclose goals. Interferences which occur between goals in an "AND"-decomposition, such as in (c), are identified as conflicts because all the subgoals are required for the decomposed goal to be satisfied.

Exceptions are supported in CREE by indicating goals which can be overridden and the stakeholders that can specify the exceptions. The conflict in Figure 4.19(c) occurs because no exception is allowed for "not disclose". Figure 4.20 illustrates how this exception can be represented with CREE. The Disclose goal, "release" for "Physicians" is an exception to the default requirement not to disclose "PHI" to "Clinicians". Indicating the "Province", which is the *maker* for all the goals, as a stakeholder who can specify exceptions to the negative Disclose goal "not release" means the disclosure to "Physicians" is a permitted exception.

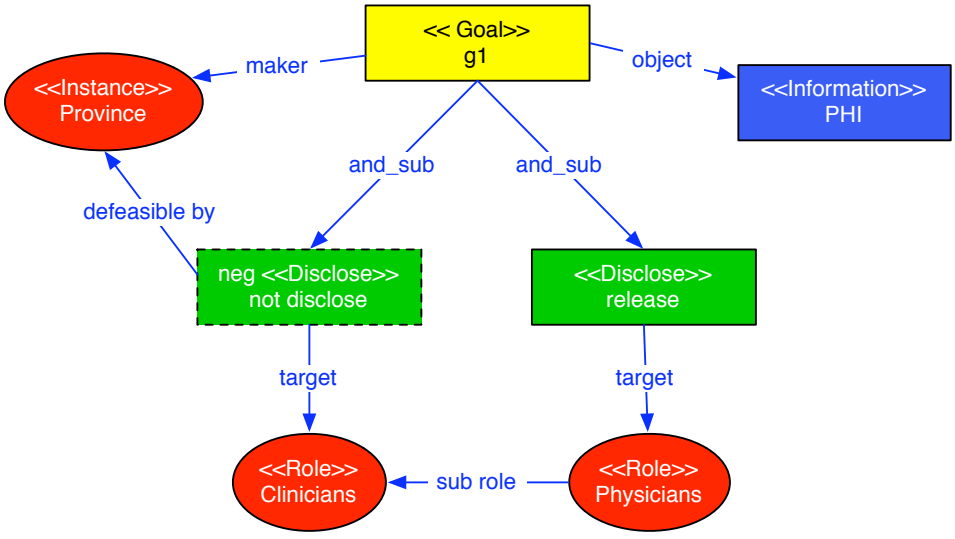


**Figure 4.20.** *Goal exception*

Definitions 9, 15 and 16 provide the formal basis for identifying goal interference,

conflict and exception.

### 4.4.3 CREE Formal Definition

In this section a formal definition of CREE is presented to support formal analysis of goal models. The formal definitions are for the domain of discourse i.e., definitions of CREE concepts (goal, actor etc.) and their relationships/properties, as well as properties for goal analysis (goal subsumption, defeating goal, goal interference and conflict). The definitions are translated into corresponding predicates used for analysis of CREE goal models in a defeasible logic formalism (cf. Section 4.4.4.1).

Section 4.4.3.1 defines the formal syntax for CREE goal models. The formal syntax is used to determine the well-formedness of CREE goal models. Section 4.4.3.2 provides definitions for precise and meaningful interpretation of goal models and Section 4.4.3.3 provides proof theory definitions used for CREE goal model analysis [127].

#### 4.4.3.1 CREE Formal Syntax

Goal models are defined with elements of the CREE meta model (Section 4.1.2) and should be consistent with the following formal definitions.

**Definition 1 (CREE Confidentiality Goal Model)** *A CREE confidentiality goal model is the 6-tuple,* $(I, A, G, \leq_I, \leq_A, Rel_L)$ *where:*

- $I$ *- a set of information elements*
- $\leq_I$ *- a partial ordering of* $I$*, where for* $i_1$,$i_2 \in I$, $i_2 \leq_I i_1$ *if* $i_2$ *is part of* $i_1$
- $A$ *- a set of actors*
  - *a* role*,* $A_R$*, is a subset of* $A$*:* $A_R \subseteq A$
  - *an* instance*,* $ins$*, is a subset of* $A$ *with cardinality 1:* $ins \subset A \wedge |ins|$*=1*
- $\leq_A$ *- a partial ordering of* $A$*, where* $a_2 \leq_A a_1$ *if* $a_1, a_2 \in A \wedge a_2 \subseteq a_1$
- $Rel_L$ *- a set of relationship labels*

  - *a relationship, $rel$, is a function, which maps relationship labels to ordered pairs of actors.* $rel$*:* $Rel_L \rightarrow A \text{ X } A$

    *For example, the relationship label, "worksfor", could have the following value* $rel$*(worksfor) =* $\{$*(drX, hospital_A), (nurseY, hospital_A), (drZ, hospital_B)*$\}$

- $G$ *- a set of requirement goals of the form* $g(neg, gt, m, a, t, i, decompose, S, D)$*:*
  - $neg \in \{$*T, F*$\}$*: indicates if the requirement goal is negative*
  - $gt \in GT =$ $\{$*nomGoal, goal, confGoal, functGoal, use, disclose, collect, create, change, amend, edit, delete*$\}$*: goal type from the CREE meta model.*

    $\leq_{GT}$ *denotes the partial order of goal types, where* $gt_2 \leq_{GT} gt_1$*, if* $gt_1, gt_2 \in GT$ *and* $gt_2 = gt_1$ *or* $gt_2$ *is a sub type of* $gt_1$*.*
  - $m \in A$*: maker for the goal*
  - $a \in A$*: actor for the goal*
  - $t \in A$*: target for the goal*
  - $i \in I$*: object (information element) for the goal*
  - $decompose \in \{AND, OR\}$*: indicates if the requirement is "AND" or "OR" decomposed*
  - $S \subset G$*: subgoals of goal* $g$
  - $D \subset A$*: actors that can override goal* $g$

In the following, auxiliary functions $neg$, $gt$, $m$, $a$, $t$, $i$, $decompose$, $S$ and $D$ refer to the equally named goal elements defined above.

**Definition 2 (Defeasible Goal)** *Goal* $g$ *is defeasible if* $D(g) \neq \emptyset$*.*

$$def(g) = \begin{cases} T & D(g) \neq \emptyset \\ F & D(g) = \emptyset \end{cases}$$

**Definition 3 (Subgoal)** *Goal* $g_1 \in G$ *is a subgoal of* $g_2 \in G$*, denoted* $g_1 \lhd g_2$*, if*

- $g_1 \in S(g_2)$ *or*
- $\exists g_3 \in G$, $(g_1 \lhd g_3) \wedge (g_3 \lhd g_2)$

$\lhd$ is extended with the reflexive relation $g \lhd g$.

**Definition 4 (Subgoal Property Subsumption)** *A goal $g_s$ whose maker, actor, target or object property is not specified ($undef$), assumes the property from its most immediate parent goal $g$, which has the corresponding property specified. Hence,*

$$m^{'}(g_s) = \begin{cases} m(g) & m(g_s) = undef \wedge g_s \lhd g \wedge m(g) \neq undef \\ & \wedge \forall g_i (g_s \lhd g_i \lhd g)\ m(g_i) = undef \\ m(g_s) & m(g_s) \neq undef \\ undef & \end{cases}$$

$$a^{'}(g_s) = \begin{cases} a(g) & a(g_s) = undef \wedge g_s \lhd g \wedge a(g) \neq undef \\ & \wedge \forall g_i (g_s \lhd g_i \lhd g)\ a(g_i) = undef \\ a(g_s) & a(g_s) \neq undef \\ undef & \end{cases}$$

$$t^{'}(g_s) = \begin{cases} t(g) & t(g_s) = undef \wedge g_s \lhd g \wedge t(g) \neq undef \\ & \wedge \forall g_i (g_s \lhd g_i \lhd g)\ t(g_i) = undef \\ t(g_s) & t(g_s) \neq undef \\ undef & \end{cases}$$

$$i^{'}(g_s) = \begin{cases} i(g) & i(g_s) = undef \wedge g_s \lhd g \wedge i(g) \neq undef \\ & \wedge \forall g_i (g_s \lhd g_i \lhd g)\ i(g_i) = undef \\ i(g_s) & i(g_s) \neq undef \\ undef & \end{cases}$$

$m^{'}, a^{'}, t^{'}, i^{'}$ denote the subsumed maker, actor, target and object of a goal.

**Definition 5 (Implied Goals)** *The set of implied goals of goal $g$, denoted $g!$ is defined as* $g! = \{r | r \lhd g \wedge decompose(g) = AND \wedge \forall y (r \lhd y \wedge y \lhd g) \Rightarrow decompose(y) = AND\}$

Implied goals of $g$ are subgoals of $g$ which are reachable through traversal of connected paths of conjunctive subgoals of $g$. $g \in g!$ follows from the extended definition of $\lhd$.

#### 4.4.3.2 CREE Semantic Definitions

The following semantic definitions form the basis for interpretation of a CREE goal model.

**Definition 6 (Disjunctive Goal)** *Goal $g$ is disjunctive if $decompose(g) = OR$. $g$ is decomposed into goals, $S(g) = \{g_1, g_2, ... g_n\}$, $g \neq g_j$, $1 \leq j \leq n$.*

**Definition 7 (Conjunctive Goal)** *Goal $g$ is conjunctive if $decompose(g) = AND$. $g$ is decomposed into goals, $S(g) = \{g_1, g_2, ... g_n\}$, $g \neq g_j$, $1 \leq j \leq n$.*

**Definition 8 (Incomplete Goal)** *Goal $g$ is incomplete if the $maker$, $actor$, $target$ or $object$ property is not specified and the property is not subsumed from a parent goal.*

*Therefore, $g$ is incomplete if*

$$m^{'}(g) = undef \vee a^{'}(g) = undef \vee t^{'}(g) = undef \vee i^{'}(g) = undef$$

Subsumption (stratification) relationship between goals is one of the basis for detecting inconsistencies. Given two goals, $g_1$ and $g_2$, $g_1$ is subsumed by $g_2$ if the goal type of $g_1$ is the same or a specialization of the goal type of $g_2$, actor and target of $g_1$ are equal or a specialization of the actor and target of $g_2$ respectively, and the object of $g_1$ is the same or a part of the object of $g_2$.

**Definition 9 (Goal Subsumption)** *Given goals $g_1, g_2 \in G$, $g_1$ is more specific than (subsumed by) $g_2$, denoted $g_1 \ll g_2$, if and only if*

- $gt(g_1) \leq_{GT} gt(g_2)$,
- $a(g_1) \leq_A a(g_2)$,
- $t(g_1) \leq_A t(g_2)$,
- $i(g_1) \leq_I i(g_2)$

$\ll\gg$ denotes the symmetric extension of $\ll$, i.e., $g_1 \ll\gg g_2 \Leftrightarrow (g_1 \ll g_2 \vee g_2 \ll g_1)$.

For an information entity, symmetric goal types can represent the same information flow between stakeholders. An information flow objective can be represented from the perspectives of two stakeholders with goals of the symmetric goal types. One of the stakeholders is the actor of the first goal and target of the second goal, while the other stakeholder is the target for the first goal and actor of the second goal.

**Definition 10 (Symmetric Goal types)** *Let $b, c \in A$, $gt_1, gt_2 \in GT$ and $gt_1 \neq gt_2$. $gt_1$ is symmetric to $gt_2$ if information flow between $b$ and $c$ can be represented by goals $g, h \in G$ such that*

$$g : gt(g) = gt_1, a(g) = b, t(g) = c$$
$$h : gt(h) = gt_2, a(h) = c, t(h) = b$$

Goals $g, h$ are symmetric goals, denoted $g \rightleftharpoons h$.

Goal types *Collect* and *Disclose* are symmetric goal types. A collect goal with actor b, and target c, is symmetric to a disclose goal with actor c and target b for an information entity.

From Definition 10, it follows that

i) $(g_1 \rightleftharpoons g_2) \Rightarrow g_1 \ll\gg g_2$

ii) $(g_1 \rightleftharpoons g_2) \wedge (g_2 \ll\gg g_3) \Rightarrow g_1 \ll\gg g_3$

**Definition 11 (Derived Goal)** *A goal $g_i$ is a derived goal, from specified goal $g$, if there exists $a(g_i)$ or $t(g_i)$ or $i(g_i)$ such that*

$$a(g_i) <_A a(g);\ t(g_i) <_A t(g);\ i(g_i) <_I i(g).$$

*$neg$ and $m$ properties of $g_i$ are the same as those of $g$.*

#### 4.4.3.3 CREE Proof Theory

**Definition 12 (Defeating Goal)** *Given goals $g_1$, $g_2 \in G$, $g_1$ defeats $g_2$, denoted $g_1$ $sup$ $g_2$, if:*

*i)* $neg(g_1) \neq neg(g_2)$

*ii)* $g_1 \ll g_2$, $def(g_2) =$*T*, $\exists\ a \in D(g_2) \wedge m(g_1) \leq_A a$

**Definition 13 (Definitely Inferred Goal)** *A goal, g, is definitely inferred if* $def(g) =$ *F.*

**Definition 14 (Defeasibly Inferred Goal)** *A goal, g, is defeasibly inferred if:*

*i) g is definitely inferred or*

*ii)* $def(g) =$ *T, and*

- $\forall g_c \in$ *C(g),* $g_c$ *is not definitely/defeasibly inferred*
  *where C(g) is set of complements of g*
- $\neg\exists g_d : g_d\ sup\ g$

**Definition 15 (Interfere)** *Goals* $g_1$ *and* $g_2$ *interfere with respect to goals* $x$ *and* $y$*, denoted as* $g_1 \dagger_y^x g_2$ *if*

*i)* $neg(x) \neq neg(y) \wedge x \ll\gg y$

*ii)* $x \lhd g_1 \wedge y \lhd g_2$

*iii)* $gt(x), gt(y) \leq_{GT}$ *"confGoal"*

The interfere definition denotes two goals that may potentially conflict due to offending confidentiality goals $x$ and $y$, which have a subsumption relationship ($\ll\gg$)

**Definition 16 (Conflict)** *Goals* $g_1$ *and* $g_2$ *conflict with respect to goals* $x$ *and* $y$*, denoted as* $g_1 \ddagger_y^x g_2$ *if*

*i)* $g_1 \dagger_y^x g_2$

*ii)* $x \in g_1! \wedge y \in g_2!$

*iii)* $\exists g! : g_1, g_2 \in g!$

*iv)* $\neg(m(g_1) \leq_A D(g_2) \vee m(g_2) \leq_A D(g_1))$

Two interfering goals conflict if the offending goals/subgoals are in exclusively *AND*-decomposed subtrees and one of the goals is not specified as an exception to the other. The last item in Definition 16 excludes scenarios where one of the goals is allowed to *defeat* the other; this is used in distinguishing between conflicts and allowed exceptions.

Definitions 15 and 16 provide the basis for goal consistency checks and give approximations for the consistency of the model. An interference indicates goals which may be inconsistent and require further review while conflicts are inconsistent.

$inf(g)$ defined below is used in Definition 17. $inf(g)$ indicates if a goal is inferred (definitely or defeasibly).

$$inf(g) = \begin{cases} T & \text{g is definitely or defeasibly inferred} \\ F & \end{cases}$$

Based on the goal satisfaction axioms in Section 3.2, the satisfaction of a CREE goal is defined below (Definition 17).

**Definition 17 (Goal Satisfaction)** *The satisfaction of goal g, denoted,* $sat(g)$ *is*

$$sat(g) = \begin{cases} inf(g) \wedge \forall g_i \in S(g)(sat(g_i) \wedge \neg(g \ddagger_y^x g_i)) & decompose(g) = AND \\ inf(g) \wedge \exists g_i \in S(g)(sat(g_i) \wedge \neg(g \ddagger_y^x g_i)) & decompose(g) = OR \\ inf(g) & S(g) = \emptyset \end{cases}$$

A goal without subgoals is satisfied if it is inferred. A decomposed goal's satisfaction is based on satisfaction propagation from its subgoals. A conjunctive goal is satisfied if it is inferred, all its subgoals are satisfied and it does not conflict with any of its subgoals. A disjunctive goal is satisfied if it is inferred and there is a subgoal which is satisfied and does not conflict with the goal.

### 4.4.4 Formal Analysis in CREE-tool

Formal analysis of goal models in CREE-tool is provided through CREE-DR-Prolog window shown in Figure 4.21. This is launched on clicking the "CREE-DR-Prolog" button in the bottom right pane of CREE-tool.

The upper left pane displays the generated logic program translation of the goal models. The logic translation (cf. Section 4.4.4.1) is used in the formal analysis of the models. For example, the highlighted non-defeasible rule in Figure 4.21 is the translation of the *collect* goal, "record" of Figure 4.11 to the logic program notation.

On the right pane of the formal analysis window, the user can select different analysis options. These include:

- inferring definite or defeasibly derived goals (Definitions 13, 14)
- identifying subgoals (Definitions 3)
- identifying incomplete goals (Definition 8)
- identifying interfering goals (Definition 15)
- identifying conflicting goals (Definition 16)

The lower left pane displays results of the analysis. Figure 4.21 shows the result of analysis to identify defeasibly derived *collect* goals.

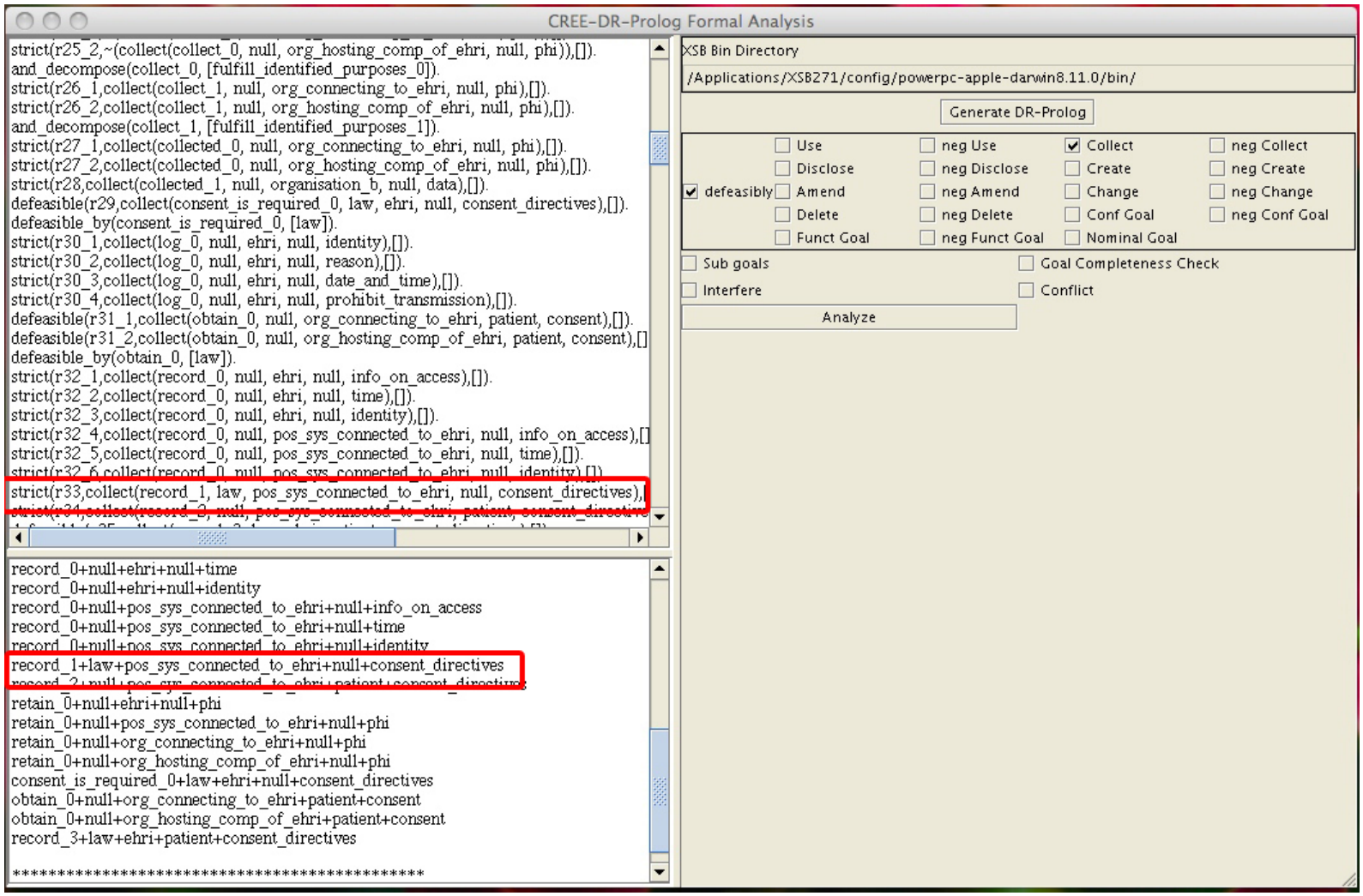


**Figure 4.21.** *CREE-tool's Formal Analysis Window*

#### 4.4.4.1 Implementation with Logic Programming

Analysis of CREE models is realized by translating the models into defeasible logic formalism (cf. Section 2.5.1).

Model instances of actors and their relationships as well as information elements are translated into defeasible logic facts while CREE goals are translated into defeasible logic rules. A defeasible goal is translated to a defeasible rule and a non-defeasible goal is translated to a strict rule.

Defeasible logic supports superiority relations between rules. Defeating relationships between CREE goals (Definition 12) establish priorities between goals and are used during analysis to infer superiority of the defeasible logic rules which represent the goals. The definitions of Section 4.4.3 are also translated to appropriate logic clauses.

An implementation of a defeasible reasoning engine is adopted to facilitate analysis. CREE goal models are translated to the format of the reasoning engine. The translation process uses predicates which are defined for CREE meta model concepts and their relationships. The predicates for the goals are used in combination with the defeasible reasoning engine's syntax for defeasible and non-defeasible rules in order to represent CREE goals. Similarly, negative goals are translated by using the appropriate operator for the implementation.

CREE-tool provides automatic translation of goal models to the formal logic notation and functionality for goal model analysis (cf. Sections 4.4.1, 4.4.2). The adopted reasoning system, as well as the predicates defined for CREE meta model concepts are described in the following subsections.

#### 4.4.4.2 Defeasible Prolog Translation

CREE-tool's default reasoning implementation is based on DR-Prolog [26]. DR-Prolog is a system for defeasible reasoning which uses XSB logic programming system [21] as its reasoning engine. XSB is a Prolog programming language dialect which extends Prolog

with HiLog (a type of higher order logic programming) as well as tabled resolution, which is useful for recursive query computation.

DR-Prolog defines the defeasible logic proof theory as a metaprogram. The metaprogram is compiled along with defeasible theories which were translated to logic programs. The following predicates are defined in the metaprogram [26], to represent the basic elements of a defeasible theory, i.e., $D = (F, R, >)$ (cf. Section 2.5.1):

- fact

  **fact**(p). for each p $\in$ F

- strict

  **strict**($r_i$, p, [$q_1$,. . ., $q_n$]). for each rule $r_i$: $q_1$, $q_2$,. . .,$q_n \rightarrow$ p $\in$ R

- defeasible

  **defeasible**($r_i$, p, [$q_1$,. . ., $q_n$]). for each rule $r_i$: $q_1$, $q_2$,. . .,$q_n \Rightarrow$ p $\in$ R

- sup

  **sup**(r, s). for each pair of rules such that r $>$ s.

The proof theory used in CREE-tool extends DR-Prolog with definitions to use goal properties, actor and information hierarchies, and to infer goal interferences and conflicts. The following subsection describes the predicates used for translation of CREE goal models into the logic program.

#### 4.4.4.3 CREE goal models in DR-Prolog

The following predicates are defined for the CREE meta model concepts:

i) actor(*Label*): represents an Actor instance.

ii) role(*Label*): represents a Role instance.

iii) instance(*Label*): represents an instance of CREE meta model Instance.

iv) member_of(*Instance, Role*): denotes role membership for a CREE instance, i.e., *Instance* is a member of *Role*.

v) is_a(*SubRole, Role*): denotes *SubRole* is sub role of *Role*.

vi) relationship(*Label, From, To*): represents Relationship concept. *Label* is the Relationship's label, *From* is list of actors, which are values of the Relationship's *from* property and *To* is list of actors, which are values of the Relationship's *to* property.

vii) info(*Label*): denotes an instance of Information.

viii) info_about(*Info, Thing*): represents an *about* association. *Info* is an instance of Information and *Thing* is an instance of a CREE meta model concept.

ix) part_of(*Part, Info*): represents a *part of* of association. *Part* is an information entity which is part of *Info*.

x) owner(*Info, Actor*): represents an *owner* association. *Info* is an information entity owned by *Actor*.

xi) custodian(*Info, Actors*): represents a *custodian* association. *Info* is an information entity held by actors specified in list *Actors*.

xii) goal(*Label, Maker, Actor, Target, Object*): represents a goal. *Label* is the goal's label, *Maker*, the goal's *maker* property, *Actor*, the goal's *actor* property, *Target*, the goal's *target* property and *Object*, the goal's *object* property.

xiii) nomGoal(*Label, Maker, Actor, Target, Object*): represents a *Nominal Goal*, with identical arguments as those for the goal predicate.

xiv) functGoal(*Label, Maker, Actor, Target, Object*): represents a *Funct Goal*, with identical arguments as those for the goal predicate.

xv) confGoal(*Label, Maker, Actor, Target, Object*): represents a *Conf Goal*, with identical arguments as those for the goal predicate.

xvi) use(*Label, Maker, Actor, Target, Object*): represents a *Use* goal, with identical arguments as those for the goal predicate.

xvii) disclose(*Label, Maker, Actor, Target, Object*): represents a *Disclose* goal, with identical arguments as those for the goal predicate.

xviii) collect(*Label, Maker, Actor, Target, Object*): represents a *Collect* goal, with identical arguments as those for the goal predicate.

xix) create(*Label, Maker, Actor, Target, Object*): represents a *Create* goal, with identical arguments as those for the goal predicate.

xx) change(*Label, Maker, Actor, Target, Object*): represents a *Change* goal, with identical arguments as those for the goal predicate.

xxi) amend(*Label, Maker, Actor, Target, Object*): represents an *Amend* goal, with identical arguments as those for the goal predicate.

xxii) edit(*Label, Maker, Actor, Target, Object*): represents an *Edit* goal, with identical arguments as those for the goal predicate.

xxiii) delete(*Label, Maker, Actor, Target, Object*): represents a *Delete* goal, with identical arguments as those for the goal predicate.

xxiv) defeasible_by(*Label, Actors*): denotes a *defeasible By* association. *Label* is the goal's label and *Actors* is a list of actors.

xxv) and_decompose(*Label, SubGoals*): denotes an AND-decomposition of a goal. *Label* is the goal's label and *SubGoals* is a list of its subgoals.

xxvi) or_decompose(*Label, SubGoals*): denotes an OR-decomposition of a goal. *Label* is the goal's label and *SubGoals* is a list of its subgoals.

Each default goal has a corresponding *defeasible_by* clause added to the logic program for its *defeasible by* association to actor(s). Similarly, subgoals are translated by adding the corresponding *and_decompose* or *or_decompose* clause.

Goal models in the logic program representations are analyzed using clauses for CREE analysis. Details of the clauses for interference and conflict are provided in Appendix D.

As indicated in Section 2.5.1.1, the validity of conclusions from defeasible reasoning systems should be based on current information (i.e., current facts and rules). A conclusion is drawn if there is no defeating evidence in the current knowledge base. However, such conclusions are withdrawn when there is additional reasoning or information to the

contrary.

Although defeasible reasoning systems do not generally produce all and only warranted conclusions due to the intractable nature of identifying all defeaters, recursively enumerating the set of defeaters in CREE is tractable because defeating goals are restricted to goals from actors who can specify exceptions, i.e., actors who have a *defeasibly_by* association.

## 4.5 Summary

CREE provides a method to represent and analyze confidentiality requirements from natural language sources. The requirements are extracted from natural language sources as goal model structures through an annotation process which involves the use of CREE meta model concepts as annotation elements. Analysis of the derived goal models can be performed to identify incompleteness and conflicts.

CREE's incompleteness analysis checks structural constraints of goal models, e.g., missing goal properties, which could be a result of omissions during goal annotations or ambiguity in the original source. Identifying these omissions supports the goal annotation process and highlights points of clarification in the natural language source.

CREE's inconsistencies analysis is used to determine interferences and conflicts in goal models. This is based on stratified relationships between goals and goal decompositions. The stratified relationships are inferred with CREE goal properties *actor, target and object*. Goals with stratified relationships interfere if they have opposite *negative* property values. Interfering goals conflict if they are conjunctive subgoals and have not been specified as possible exceptions. CREE-tool supports the annotation and analysis approach of the CREE method.

# Chapter 5

# Evaluation

This chapter describes the evaluation of the CREE approach using CREE-tool. A case study was performed to illustrate the usage scenario of CREE for extracting and analyzing goals from natural language sources. In addition, a comparative case study of CREE's inconsistency analysis with alternative methods was conducted. A user study was also performed to evaluate CREE-tool's goal extraction approach.

## 5.1 Evaluation Strategy

Two case studies using regulations and requirements in the medical domain were conducted. These requirements were used in order to get better insight of potential benefits and limitations of the CREE approach.

The first case study (Section 5.2.1) consisted of two parts. The first part involved the use of the CREE approach with privacy and security requirements of a national e-health initiative for cross-organizational and cross-jurisdictional electronic health records. In the second part of the study (Section 5.2.4) requirements from the national initiative are analyzed with privacy, security and confidentiality requirements of a provincial initiative for adoption of electronic health records. This part of the study is aimed at revealing interactions between the two sets of requirements.

The second case study (Section 5.2.5) evaluates CREE's inconsistency analysis and exception handling technique. The study compares CREE, an RNLS-based method and Tropos/Secure Tropos methods using a regulation. The RNLS-based study was conducted

by a different research group [44].

A user study evaluation (Section 5.3) was also performed. It was designed to evaluate CREE-tool's annotation approach for extracting goal models from natural language sources.

## 5.2 Case Studies

### 5.2.1 Explorative Case Study

This section describes usage scenario of the CREE approach as applied to requirements in the health care domain. The case study was conducted in order to explore the benefits and feasibility of CREE's annotation method for extracting goal models from natural language sources and how this process is aided by CREE's analysis.

The selected case is Canada Infoway's Privacy and Security requirements [2]. These provide overall privacy and security requirements for an interoperable electronic health record. The requirements are based on:

a) data protection laws and regulations

b) privacy and security best practices

c) common privacy and security issues in health care situations

The requirements reference Canadian laws and regulations and address the following [2]:

- privacy and security requirements of the Electronic Health Record Infostructure (EHRi) and those of the organisations which connect to the EHRi e.g., organisations hosting EHRi components and organisations contributing to the EHRi;
- policy and business related privacy and security requirements which impact the EHRi privacy and security conceptual architecture;
- privacy and security issues identified in the EHRi use cases;

Requirements of the chosen case consists of typical confidentiality and privacy requirements for data protection in health care situations, therefore, the study provides insight into the application of the CREE approach to these scenarios. The case also provides a multilateral context because of inter-dependencies between Infoway requirements and other jurisdictional policies.

Privacy sections of the requirements are for data protection in an EHR (Electronic Health Record) environment and are based on principles of the Canadian Standards Association's Model Code for the Protection of Personal Information [7]. The security requirements are mainly technical requirements covering security control objectives including organizing information security, asset management, human resources security, communications, security incident management, and physical and environmental security.

The study covers the privacy sections because CREE's current models and analysis techniques do not adequately support the specified security requirements. The requirements studied are listed in Appendix A. There were 29 privacy requirements, categorized as administrative and/or technical requirements. Nineteen of the requirements were selected for the case study; these included all 11 technical requirements and 8 administrative requirements relevant for data or consent. Administrative requirements dealing with procedures such as investigation or openness were not selected. Although these can be annotated with the CREE "Goal" concept, the case study is limited to requirements with data concerns.

### 5.2.2 Annotating the case study

CREE models were created for the selected Infoway's privacy requirements using CREE-tool source document annotation feature. Visual representations of the models are provided in Appendix C. Each of the privacy requirements is depicted as a CREE Nominal goal, composed of instances of CREE concepts which appropriately express the requirement.

The models were created in 7 hours. In addition, 3 hours were spent for review and modification. Table 5.1 shows the number of instances of the other concepts identified from the case study.

| CREE concept | Instances |
|---|---|
| Role | 16 |
| Instance | 5 |
| Information | 28 |
| Relationship | 2 |
| Goal (excluding nominal goal) | 62 |

**Table 5.1.** *CREE concepts from case study*

Role instances identified include “patient”, “user”, “Organisation hosting components of the EHRI”, and “Organisation connecting to the EHRi”. Instance concepts include “Infoway”, “law” and “EHRi”.

Information instances include: “communications materials”, “privacy policies and practices”, “complaints or inquiries”, “consent directives ”, “consent”, “hosted components”, “identity”, “information on accesses”, “phi” and “time”. Relationship concepts identified are “works” and “substitute decision maker”.

The following describe some of the requirement models.

- Requirement 3 (R3)

  “Privacy Requirement 3 Privacy Policy

  Organisations connecting to the EHRi and organisations hosting components of the EHRi must implement policies and practices, including:

  a) Implementing procedures to protect PHI

  b) Establishing procedures to receive and respond to privacy related complaints and inquiries;

  c) Training users and communicating to users information about the organisations privacy policies and practices; and

  d) Developing communications materials to explain to the general public the organisations privacy policies and practices ”

  R3 consists of functional goals, e.g., “*Developing communications materials*”, “*Training and communicating to users*” and “*Implementing procedures to protect PHI*”

which are modelled as CREE "*Funct Goal*" as depicted in Figure C.1. These functional goals are modelled as subgoals of the nominal goal, "*Privacy Requirement 3*".

A convention adopted in line with Definition 4 is for subgoals to inherit values for *actor*, *target*, *object*, and *maker* attributes of the parent goals, where they are not specified for the subgoals. Hence, each goal in Figure C.1 has "*org hosting comp of EHRi*" and "*org connecting to EHRi*" as its actor attribute.

- Requirement 7 (R7)

  "Privacy Requirement 7 Limitation of Collection to Identified Purposes

  Organisations connecting to the EHRi or organisations hosting components of the EHRi should only collect PHI necessary to fulfill the purposes that they have identified. "

  The CREE representation of R7 is shown in Figure C.4. R7 indicates that PHI (personal health information) should only be collected in order to fulfill identified purposes. This is modelled with a subgoal: *Use*, "fulfill identified purposes", for the collect goal.

- Requirement 8 (R8)

  "Privacy Requirement 8 Obtaining Knowledgeable Consent

  Except where inappropriate (e.g. specifically exempted by law or professional code of practice), organisations connecting to the EHRi, and organisations hosting components of the EHRi should obtain the knowledge and consent of each patient/person for the collection, use or disclosure of his or her PHI and where required by law, must obtain the knowledge and consent of each patient/person for the collection, use or disclosure of his or her PHI. "

  R8 is an example of a scenario where default requirements with exceptions are expressed in a privacy requirement. It states that consents should be obtained for the collection, use or disclosure of PHI and indicates that an exception can be stipulated by law. Therefore, a *legislature*, could specify regulations that do not require obtaining consent, thus defeating the default requirement. This is modelled with the *defeasible by* association in the CREE meta model. The CREE representation of

R8 is shown in Figure C.5. The *Collect* goal, "obtain", for collection of consent is defeasible by actor "*law*", which can stipulate goals that defeat this goal.

- Requirement 12 (R12)

  "Privacy Requirement 12 Associating Consent Directives with PHI in the EHRi

  When consent is required by law, whenever receiving, storing, processing, or transmitting PHI, the EHRi must be able to:

  a) maintain the association between this data and the consent directives under which it may be used or disclosed;

  b) process these consent directives before transmitting the associated data and block the transmission where it would violate the directives and where no exception for such a disclosure is outlined in law; and

  c) notify the requestor whenever data is blocked as in b) above. "

  In R12 the possibility of exceptions outlined in law is also expressed. A legislature, could specify requirements that permit overriding disclosure consent. Although the actor is not explicitly mentioned, the text span "*law*" is annotated with the actor concept (Figure C.9) similar to that in R8. Other concepts that may not be mentioned explicitly or in detail are often functional goals that are typically described elsewhere, e.g., in Use Case specifications or in functional requirements documentation. Such functional goals include "*maintain the association between PHI and the consent directives*" and "*respond to request*" indicated in R12.

  Part b of R12 refers to an information usage step "*process consent directives*" which results in a conditional choice based on the outcome of this usage, i.e., to allow transmission in case of positive consent or to block transmission if consent was withheld. This is modelled by two opposite functional goals ("*violate the directives*") as shown in Figure C.9, which form part of the alternatives (transmit or block).

  CREE currently abstracts from temporal properties, e.g., "*process consent directives* before *transmitting* ..." in R12 is modelled as conjunctive subgoals "*process*" and "*transmit or block*".

- Requirement 24 (R24)

  "Privacy Requirement 24 Patient/Person Access

  Organisations connecting to the EHRi and organisations hosting components of the EHRi must, upon request:

  a) inform a patient/person of the existence, use and disclosure of his or her PHI and shall give the patient/person direct access to that PHI where such access is not prohibited by legislation;

  b) respond to requests for access to a patient/persons PHI within a reasonable time and make it available in a form that is generally understandable; and

  c) allow a patient/person to challenge the accuracy and completeness of his or her PHI and have it amended as appropriate. "

  Part a of R24 is modelled using CREE concepts as shown in Figure C.16, however, Part b and Part c cannot be adequately represented. This is because these describe performance - "*respond to requests ... within a reasonable time*"; usability - "*make it available in a form that is generally understandable*" and stakeholder capability to challenge state of information - "*allow ... to challenge the accuracy and completeness ...*" which are not supported by CREE.

### 5.2.3 Analysis

The goal models were analyzed using the functionalities provided by CREE-tool. This was performed on a computer with PowerPC G4 1.67MHz processor and 2GB RAM. The analysis took 0.313s. The interference analysis took 10% of the total time while the conflict analysis took 37.7% of the total time.

The following is a discussion of the results of the analysis.

- **Incompleteness Check**

  Results of incompleteness analysis of the case study models are shown in Table 5.2.

| Check | Number of goals |
|---|---|
| Goals with unspecified maker attribute | 0 |
| Goals with unspecified object attribute | 17 |
| Goals with unspecified actor attribute | 12 |
| Goals with unspecified target attribute | 47 |

**Table 5.2.** *Incompleteness check from case study*

All the goals have a maker attribute. This is either inherited from the parent nominal goals (not shown in Appendix C) or directly specified, e.g., *ConfGoal*, "record-keeping requirements" (Figure C.14).

Only the nominal goals had no object attribute specified.

Goals without actor attribute were: *Conf Goal* - "inappropriate access use or disclosure" (Figure C.13) and "collection usage and disclosure" (Figure C.3); *Use* - "accesses" (Figure C.15); *Funct Goal* - "handling" (Figure C.17); *neg Disclose* - "transmission" (Figure C.10) and Nominal goals for R8, R9, R10, R11, R12, R18, R22.

Goals without target attribute included those which do not necessarily need one, particularly *Funct Goal*, e.g., an EHRi marking records (Figure C.15) or organisations hosting components of EHRi assessing risks (Figure C.2); as well as those which might not be explicit but could be determined on further review. For example, the target for *Collect* goal, "record" (Figure C.6) was set to "patient".

- **Goal Inference**

  Table 5.3 shows the results of goal inference analysis for the case study. The *Annotated* column indicates the number of instances of a goal type identified during annotation. *Definitely Inferred* indicates the number of goals which were definitely inferred and *Defeasibly Inferred* indicates the number of goals which were defeasibly inferred (i.e., those that were definitely inferred and those that can be defeated - Definition 14).

The *Defeated* column of Table 5.3 indicates the number of defeasible goals which were defeated. None of the defeasible goals was defeated.

| Goal Type | Annotated | Definitely Inferred | Defeasibly Inferred | Defeasible | Defeated |
|---|---|---|---|---|---|
| Use | 5 | 4 | 5 | "direct access" (Figure C.16) | 0 |
| Collect | 12 | 9 | 12 | "obtain" (Figure C.5), "consent is required" (Figure C.9), and "record" (Figure C.8) | 0 |
| Disclose | 11 | 10 | 11 | "alert" (Figure C.13) | 0 |
| neg Disclose | 2 | 2 | 2 | – | 0 |
| Conf Goal | 11 | 10 | 11 | "purposes consistent" (Figure C.12) | 0 |
| Funct Goal | 20 | 20 | 20 | – | 0 |
| neg Funct Goal | 1 | 1 | 1 | – | 0 |

**Table 5.3.** *Goal inferences from case study*

- **Interfering goals**

  The following interfering goals were identified:

  a) *Disclose* goal "notify" and *neg Disclose* goal 'block the transmission" of R12 (Page 137), which is modelled as Figure C.9.

  b) *Disclose* goal "transmitting" and *neg Disclose* goal "block the transmission"of R12 (Page 137), which is modelled as Figure C.9.

  The goals interfere because the two pairs of opposite goals ("notify" and "block the

transmission", and "transmitting" and "block the transmission") have identical actor, target and object attributes: i.e., "EHRi", "Requestor", and "PHI".

- **Conflicting goals**

  The interfering goals "notify" and 'block the transmission" of R12 (Page 137) also conflict. The goals conflict because they are interfering goals in the same goal conjunctive path (as shown in Figure C.9) and neither goal is an exception of the other.

**Case study with RNLS**

RNLS defined for the Infoway requirements are provided in Appendix B. There is currently no tool support for creating the RNLS, which made the process tedious. A further implication of the lack of tool support is the absence of browsing or navigating to the original textual fragments. Labels, such as section identifiers, could be used for the RNLS as reference to the original sections in the source document, e.g., in this study, the created RNLS had labels which indicate the nominal privacy requirement they are part of.

#### 5.2.3.1 Summary

The following observations were made from applying the CREE method and the RNLS approach to the Infoway privacy requirements.

i) Comparing the annotation method with RNLS

   Model creation with CREE-tool was faster than with the RNLS method. Although this could be attributed to the lack of tool support with the RNLS method, a key distinction is the direct semantic parameterization provided by CREE annotations. The RNLS parameterization comprises two steps.

   Creating the RNLS and the semantic parameterizations of the 19 requirements in Appendix B took approximately 60 hours of initially extracting the restricted statements and reviews. However, the initial corresponding CREE models in Appendix C were created in 7 hours. In addition, 3 hours were spent to review and modify the models.

ii) Defeating Goals

The results of the analysis did not identify any defeated goals. To further evaluate CREE-tool's default reasoning, the following goals were introduced:

(a) Law does not require EHRi to alert privacy compliance officers of inappropriate access, and

(b) Infoway does not require POS systems connected to the EHRi to alert privacy compliance officers of inappropriate access.

These are shown in Figure 5.1 as (a) and (b) respectively. The subsequent analysis produced the following results:

1) defeasible *Disclose* goal, "alert" in R19 (Figure C.13) for actor "EHRi" was defeated by (a).

2) defeasible *Disclose* goal, "alert" in R19 for actor "POS systems connected to the EHRi" was not defeated by (b) because only "law" can specify defeating goals for "alert", as indicated by the *defeasible by* attribute.

3) (a) and (b) are definitely inferred.

4) An interference was identified between *Disclose* goal "alert" in R19 and negative *Disclose goal* "alert" (Figure 5.1 (b)) for actor "POS sys connected to EHRi", target "privacy compliance officer " and object "inappropriate access".

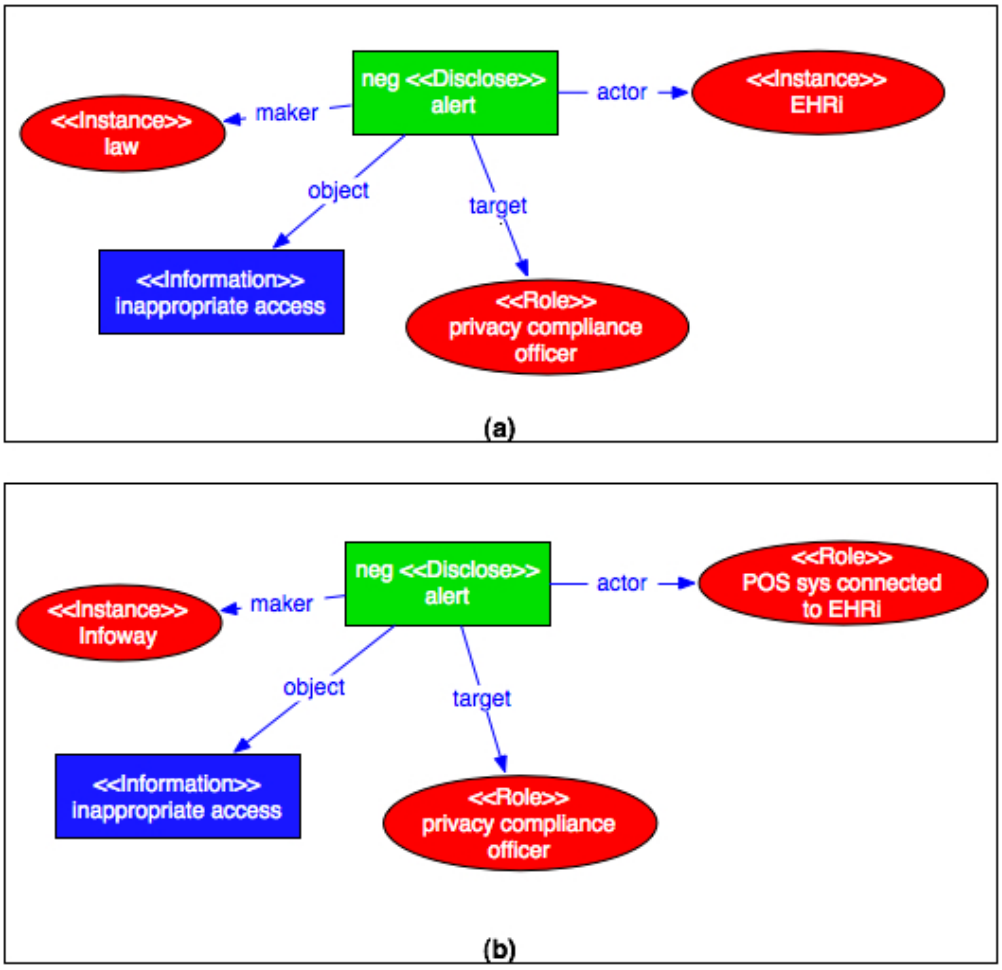


**Figure 5.1.** *Models for goals (a) and (b)*

The results show that default reasoning provided by CREE-tool can determine defeating (and defeated) goals.

c) Reviewing interferences and conflicts

The interference and conflict between "notify" and "block the transmission" is an implication of CREE's property/attribute value subsumption by subgoals and ambiguity in the source text fragment.

An unintended scenario occured from the representation shown in Figure C.9 for R9: "EHRi" can disclose ("notify") "PHI" to "requestor" if consent directives are violated. This is because no explicit detail is given on what constitutes the notifica-

tion i.e., "*notify the requestor whenever data is blocked*". Hence, the object property ("PHI") is inherited from a parent goal. "notify" passed the incompleteness check due to property inheritance.

Further review of the derived model and original document should be performed when interferences and conflicts are detected by the automated analysis. In this case, clarifying what information makes up a notification and setting this as the object value for goal "notify" addresses this interference/conflict problem as long as the object attribute for "block the transmission" is set to "PHI".

The unintended scenario triggered the occurrence of an interference and a conflict, and subsequently further clarification in the source document. This also illustrates the use of CREE's analysis to detect defects in the original document through analysis of the goals derived from the document. Overall, annotation with property inheritance for subgoals was found to be mainly problem free.

d) CREE annotation limitation

Similar to Parts b and c of R24, Requirements 22 (R22) and 25 (R25) from Appendix A were not modelled because key concepts for these requirements cannot be adequately expressed with CREE concepts. For example, R22 indicates that stakeholders involved in the Infoway EHR should "*ensure that PHI is as* accurate*,* complete*, and* up-to-date ..." These are integrity concerns which are not supported by CREE.

### 5.2.4 Multilateral Analysis

This part of the first case study involved the use of two sets of requirements: the Infoway requirements described in the previous section and requirements for a Canadian provincial e-health initiative to facilitate adoption of electronic health records for physician offices. The provincial initiative establishes a set of requirements, including the article on privacy, security and confidentiality for service providers.

The multilateral analysis uses CREE's stratified goal relationships to identify interactions between the two sets of requirements. The Infoway requirements address cross-jurisdictional concerns, therefore, analyzing these requirements with the provincial requirements provides insights into the consistency of the two set of requirements.

There are 25 requirements in the article from the provincial initiative. Twenty three of these requirements were annotated using CREE-tool; 2 requirements were not annotated due to CREE's limitations as observed with the Infoway requirements.

Table 5.4 shows the number of instances of the concepts identified during annotation of the provincial requirements:

| CREE concept | Instances |
|---|---|
| Role | 20 |
| Instance | 5 |
| Information | 11 |
| Goal (excluding nominal goal) | 76 |
| Nominal goal | 23 |

**Table 5.4.** *CREE concepts from provincial requirements*

Role instances identified in this study include "patient", "physician", "staff", "person" and 'service provider". Instance concepts include "province", "privacy commissioner" and "public". Information instances include: "notice", "terms and circumstances", "particulars" and "phi".

#### 5.2.4.1 Findings

The following were observed during this part of the case study.

i) During analysis of the provincial requirements, a defeating exception was identified. This is similar to defeating goals described in the previous section. The following text span (*P1*) was modelled as shown in Figure 5.2.

"Access by the Province to Patient Personal Information (P1)

(a) The Province will not access, collect, copy or retrieve Patient Personal Information that is in the possession of the Service Provider, including Patient Personal Information that is stored in the Data Centre, except as expressly provided in (b).

(b) The Province may only receive, access and use Patient Personal Information that is in the possession of the Service Provider, including Patient Personal Information that is stored in the Data Centre: (i) if the Patient Personal Information is provided to the Province by or at the instructions of the Eligible Physicians .... "

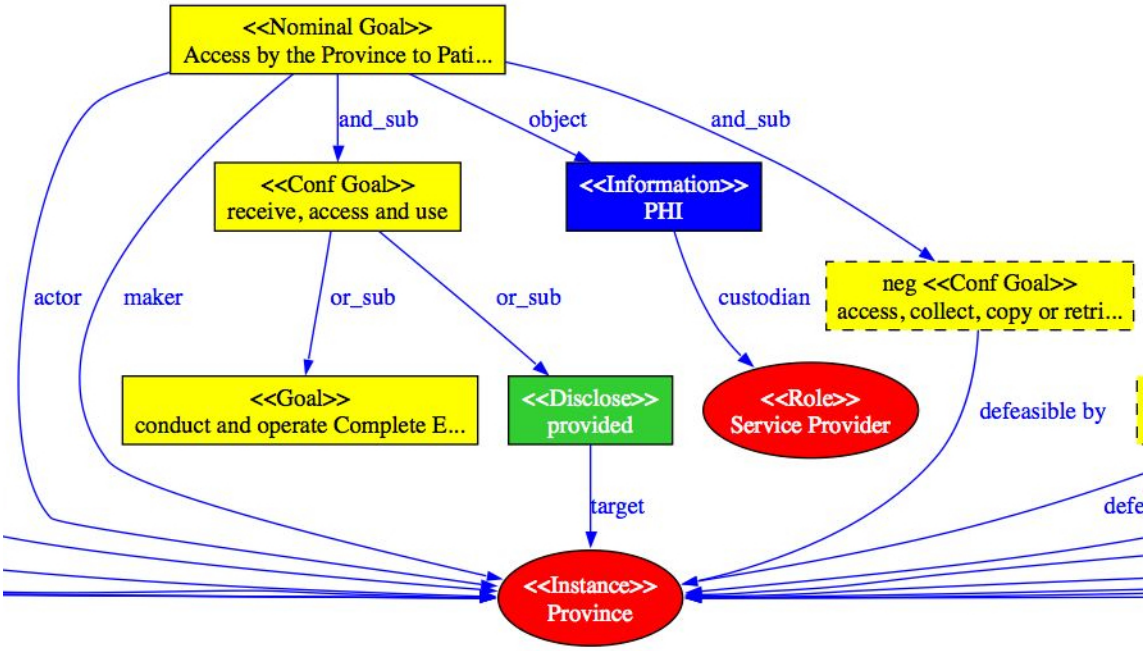


**Figure 5.2.** *Model for P1*

Part (a) (represented as negative Conf Goal, "access, collect, copy or ..." in Figure 5.2) is defeated by part (b) (represented as Conf Goal, "receive, access and use" in Figure 5.2).

ii) In order to study the interaction of the provincial requirements with the Infoway requirements, the following assumptions were made: the *service provider* role identified in the provincial requirements is considered a type of the roles *Organisation*

*hosting components of the EHR* and *Organisation connecting to the EHRi* from the Infoway requirements. Therefore, "sub role" associations were added from service provider to the two Infoway roles. The negative disclose goal, shown in Figure 5.3, was modelled from the following excerpt from the provincial requirements:

"Foreign Disclosures (P2)

The Service Provider will immediately inform the Province if the Service Provider receives any order, directive, ruling, requirement, judgment, injunction, award or decree, decision, or other requirement issued pursuant to any Foreign Disclosure Laws, or any directions or requests from any Affiliate of the Service Provider in respect of the same, and in each case, related to any Patient Personal Information (each a Disclosure Order) that is in the possession of the Service Provider, including while stored at the Data Centre. Upon receipt of a Disclosure Order, the Service Provider will not disclose any Patient Personal Information in response thereto ..."

The target property for the negative Disclose goal is not explicit for the disclosure order from a Foreign Disclosure Law. Interferences were identified between the negative Disclose goal and some goals modelled as Conf Goal from the Infoway requirements. On further review, it was realized that the interferences occur between the negative Disclose goal and Conf Goals from the Infoway models with:

- "PHI" as the object attribute,
- "Organisation hosting components of the EHR" or "Organisation connecting to the EHRi" as the actor attribute, e.g., Conf Goal in Figure C.5,
- undefined target attribute.

The interferences, while not conflicts, can be resolved if the target attributes are defined and the values do not have hierarchical relationships.

The following can be concluded from the observed interference:

(a) Non-assignment of values to goal attributes can cause interferences when the requirements are combined with other sets of requirements. Independently, the

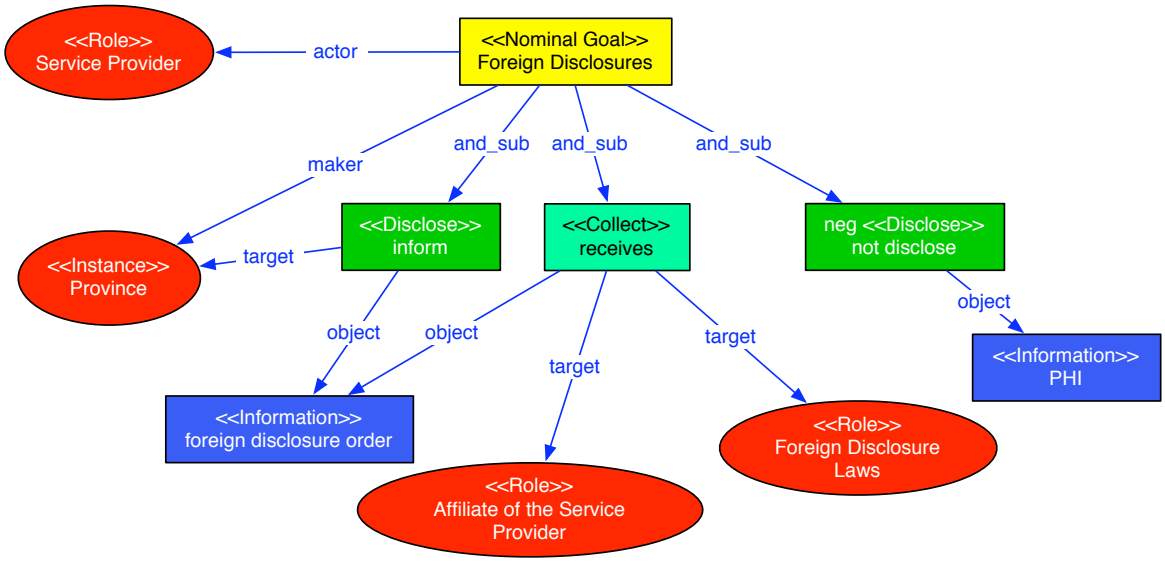


**Figure 5.3.** *Model for P2*

two sets of requirements didn't have interferences from these goals. This re-emphasizes the potential for interactions between sets of requirements and the capability of the CREE method to assist in identifying these interactions.

(b) Ideally all the goal attributes should be assigned. However, this is not often possible immediately because they are not explicit or they are ambiguous. An indication of interference or incomplete goals should be followed by review of the goals and the original natural text.

In addition, the defeating goal identified from the provincial requirements exemplifies the exception handling capability of CREE. This is further discussed in the second case study (cf. Section 5.2.5.3).

### 5.2.5 Comparative Case Study

This study compares CREE's analysis with the RNLS-based and Tropos/Secure Tropos methods. In addition to conflict analysis, the exception handling in CREE using defaults is

evaluated and compared with the RNLS method.

The study used Section 164.520 of the US Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule [15]. The HIPAA section addresses the disclosure of notices of privacy practices for protected health information. Section 5.2.5.1 summarizes the RNLS study which was conducted by Breux et. al [44]. Section 5.2.5.2 provides details of the CREE study and Section 5.2.5.4 describes the use of the Secure Tropos method.

#### 5.2.5.1 RNLS Extraction and Analysis

The RNLS=basd study by Breaux and Antón extracted rule statements, which include obligations or rights of stakeholders, as well as constraints [44]. A right is defined as: "an action that a stakeholder is conditionally permitted to perform", and an obligation is defined as: "an action that a stakeholder is conditionally required to perform" [44].

The following obligation statements were extracted from the section (the specific subsections are specified in parenthesis):

- O2: The GHP must provide notice to any person (a)(2)(ii)(B).
- O4: The GHP is not required to provide notice to any person (a)(2)(iii).
- O7: The CE must provide the notice to any person or individual (c)
- O8: The HP must provide the notice (c)(1)(i) to any person or individual (c).
- O10: The HCP must (c)(2) provide notice (c)(2)(i) to the individual (c).
- O13: The CE must provide electronic notice to the individual (c)(3)(i).
- O14: The CE must provide a paper copy of the notice to the individual (c)(3)(ii).
- O15: The HCP must automatically provide electronic notice to the individual (c)(3)(iii).

Stakeholders involved in the obligations are:

i) HP - Health plan: an individual or group plan that provides, or pays the cost of, medical care. This could be a group health plan (GHP), a health insurance issuer or a health maintenance organization.

ii) HCP - Health care provider: a provider of medical or health services

iii) CE - Covered entity could be a health plan, health care provider

The stakeholders were organized into a class hierarchy, and Figure 5.4 shows part of the hierarchy with the stakeholders from the obligations above.

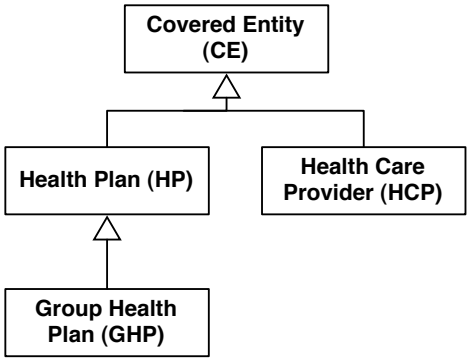


**Figure 5.4.** *Stakeholder Class Hierarchy [44]*

Queries were used to compare semantic models of obligations and right. The results of the queries were used to establish a partial ordering of obligations or rights that affect stakeholders at different levels of abstraction. For the obligations above, the query algorithm was used to generate the hierarchy shown in Figure 5.5. Obligation O7 is the most abstract in the hierarchy because it is made up of the most minimal model attributes of the obligations.

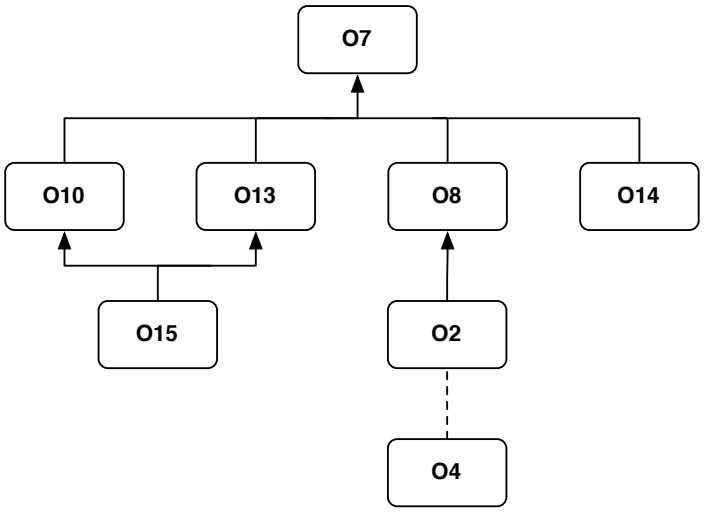


**Figure 5.5.** *Obligation Hierarchy [44]*

Obligation O4 was identified as conflicting (represented by the dotted line) with obligation O2, because O4 represents situations in which the GHP is not required to provide notice to individuals [44].

#### 5.2.5.2 CREE Annotation and Analysis

Similar to goals, obligations, can be operationalized into requirements while rights can be reformulated into implied rights and obligations of counter parties [44]. Therefore, the obligations can be annotated as CREE goals.

The stakeholders from the obligations were modelled as CREE roles and the subtype relationship between the stakeholders was represented by CREE "sub role" association. Figure 5.6 depicts the stakeholders as modelled with CREE.

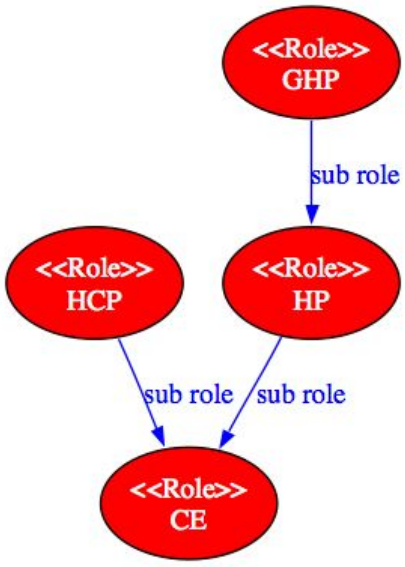


**Figure 5.6.** *Stakeholder Hierarchy with CREE*

Figure 5.7 shows the representation of the obligations in CREE. The role "Individual" represents the individual or person stakeholder. Each obligation is represented by a *Disclose* goal and modelled as a subgoal of the nominal goal "notice of privacy practices". Information entities "paper copy of notice" and "electronic notice" are modelled as part of a "notice". "HIPAA" stakeholder was modelled as the *maker* of the goals.

Using CREE's analysis the following results were obtained:

i) interferences were identified between: O4 and 02; O4 and O8; and O4 and O7.

   The analysis identified interference between O4 and O2, similar to the result from the RNLS-based study. CREE's analysis infers the goal hierarchy using the goal properties (Definition 9). The hierarchy is subsequently used to identify interferences. Obligation O4 interferes with O2, and with all the goals which subsume O2. Identifying the interferences with the subsuming goals of O2 (i.e., O8 and O7) is also in line with an observation in the RNLS study: "*Also, a potential conflict exists since a GHP may not need to satisfy either obligation O520.7 or O520.8.*" [44].

ii) conflicts were identified between: O4 and 02; O4 and O8; and O4 and O7. O4

conflicts with these obligations because it is in an "AND"-decomposed sub-tree and it is not specified as an exception.

#### 5.2.5.3 Exception Handling

Subsection a(3) of Section 164.520 indicates an exception for inmate stakeholders. An inmate is a person who is incarcerated in a correctional institution. The exception states: "*Exception for inmates. An inmate does not have a right to notice under this section ...* " [15].

The RNLS-based method handles exceptions by assigning priorities to rights or obligations and increasing the priorities to exceptions so that they overrule the rules with lower priority [44].

CREE handles exceptions by annotating goals which can be defeated with the *defeasible by* association to actor(s) who can specify overriding requirements. The exception for the inmate was modelled as a negative *Disclose* goal, denoting that inmate will not receive notice. In addition, previously modelled notifications do not apply to all sub roles of "Individual" and can be overridden. These notifications were modelled in CREE as default goals with the *defeasible by* association. Figure 5.8 shows the updated model. Nodes for default goals are represented with dashed borders, each with a *defeasible by* association to "HIPAA", indicating that HIPAA can specify goals which override these default goals. The new goal representing the exception for the "inmate" stakeholder is shown in the figure (with a black rectangle for emphasis).

With CREE's analysis the following results were obtained:

i) interferences were identified between: O4 and 02; O4 and O8; and O4 and O7 similar to the earlier model. In addition, interference between O7 and the new exception goal was also identified.

ii) no conflicts were identified. This is because the interferences which could cause conflicts are identified as exceptions.

In multi-jurisdictional contexts, CREE exception handling approach provides an advantage over the priority-based method. For example, a requirement indicating "*Unless required by X ...*" implies that exceptions are permitted from stakeholder (X) and this can be represented with the defeasible by association to this stakeholder. In complex multi-jurisdictional scenarios, assigning goal priorities is more difficult because correct priority assignments requires a global and complete overview of all relevant multi-jurisdictional goals. In addition, priorities may have to be re-assigned when new exceptions are identified.

Reviewing the subsections for obligations O2 and O4, the actors for the two goals should be different sub roles of GHP, because each specifies distinct constraints for the affected GHP. For example, for O2: "*A group health plan that provides health benefits solely through an insurance contract with a health insurance ....*". Modelling the two goals with distinct sub roles of GHP does not produce interference/conflict between O2 and O4. However, interferences between O4 and O8, and O4 and O7 still occur because the actor for O4, $\text{GHP}_{O4}$ say, is a sub role of HP and CE.

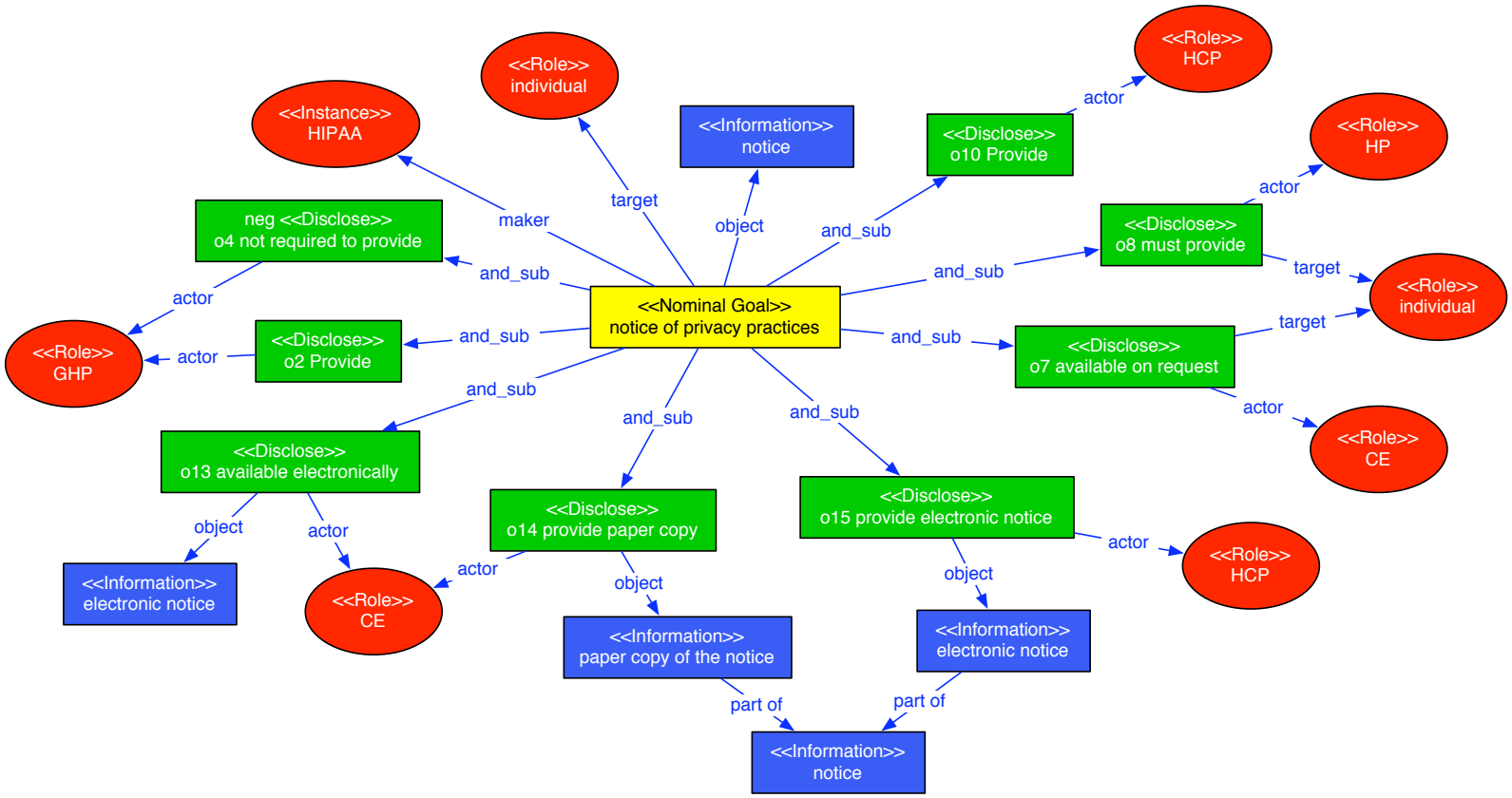


**Figure 5.7.** *CREE model for obligations*

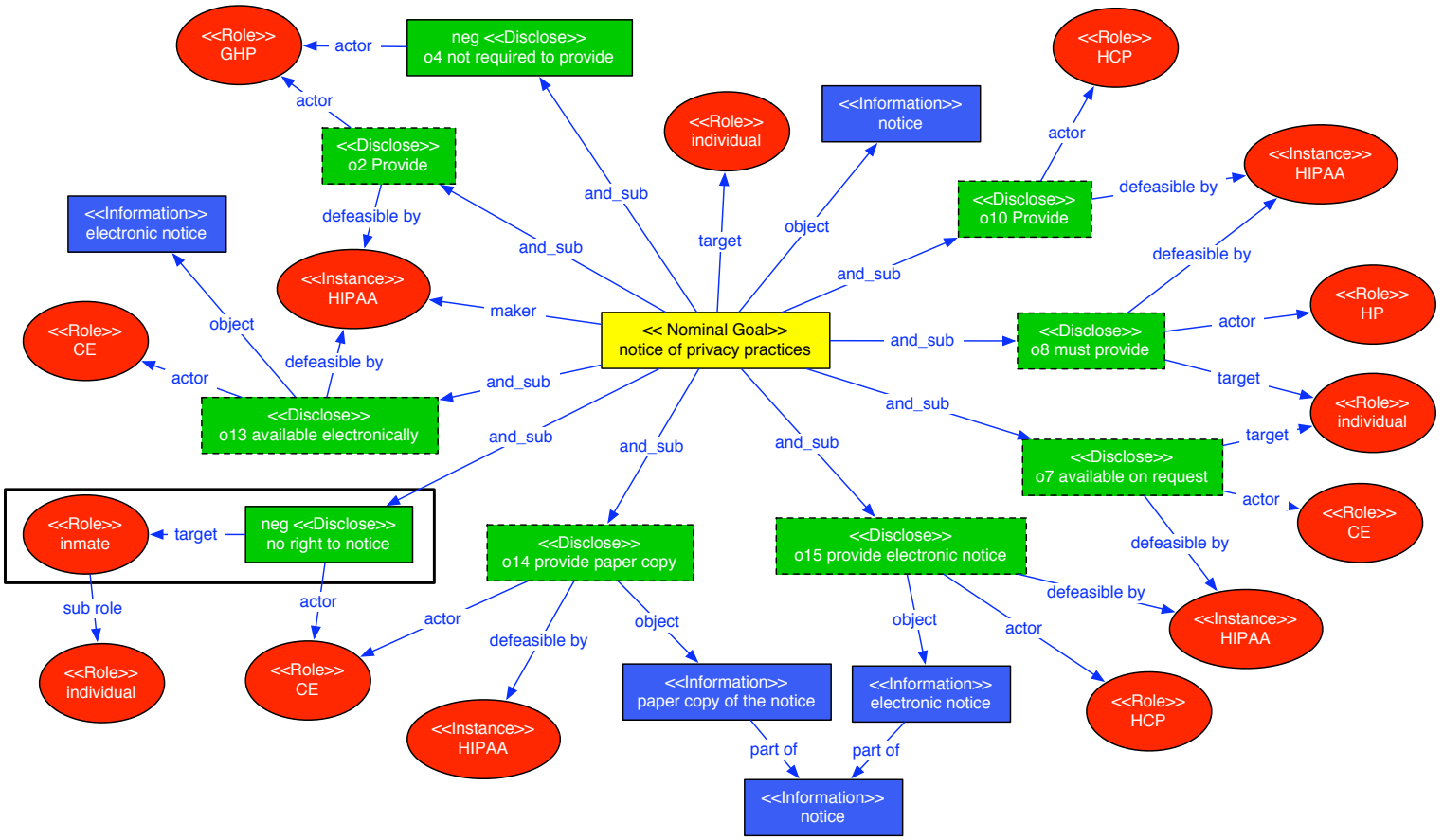


**Figure 5.8.** *CREE model for obligations with exception*

#### 5.2.5.4 Modelling and Analysis with Secure Tropos

Secure Tropos method was used to model the obligations as shown in Figure 5.9. Role hierarchies are represented by *Is A* association between roles. The provision of notice to the individual is modelled as *delegation of permission* - $D_p$ (cf. Section 3.2.4.1) of notice to the individual.

There is no explicit support for specifying negation of goals in the Secure Tropos model. Hence, O4, is not modelled. Similarly, default goals and exceptions between goals are not supported in the Secure Tropos models. Therefore, the conflicts/exceptions previously described were not detected.

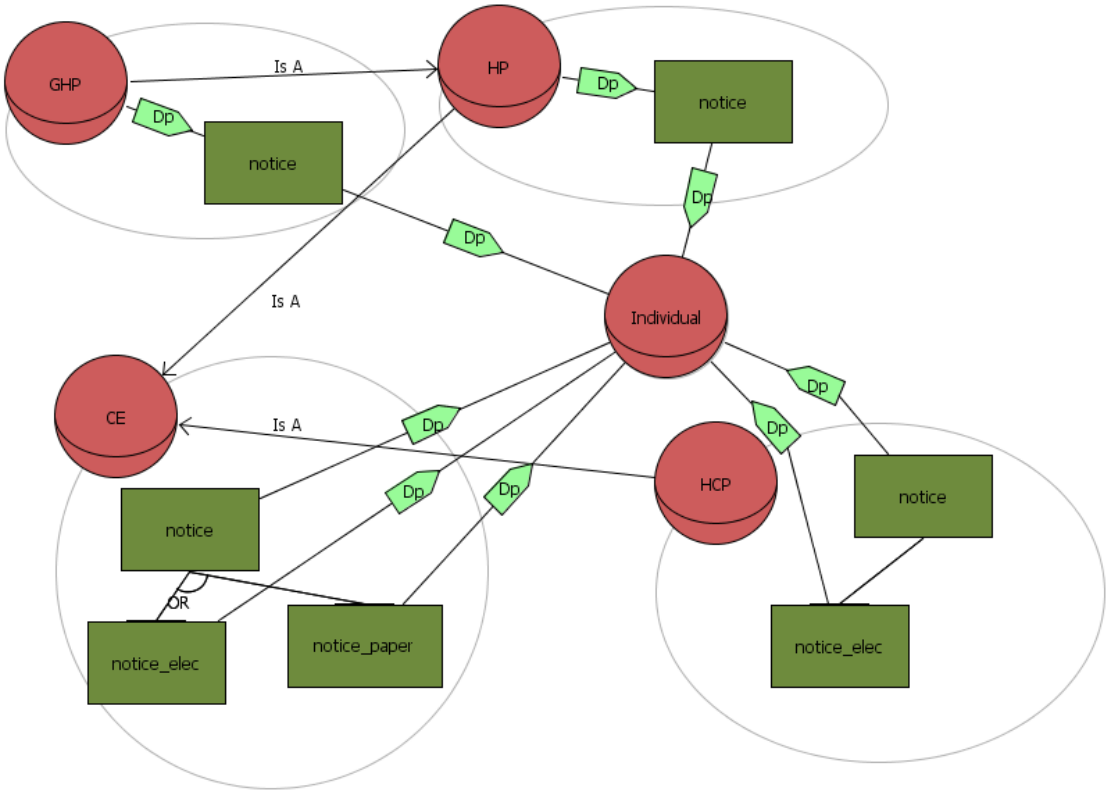


**Figure 5.9.** *Modelling with Secure Tropos*

However, Secure Tropos analysis returns possible violations of security properties, including authorization violations. For the model shown in Figure 5.9, authorization viola-

tions based on Secure Tropos trust dependencies were identified. These violations are due to a lack of trust dependencies, i.e., trust permissions, from the role delegating permission of notice to the individual. For example, there is no trust dependency from “CE” to “Individual”. The CREE method does not support specification of trust dependencies, therefore these violations cannot be identified with CREE analysis.

#### 5.2.5.5 Summary

The following summarizes the findings for the second case study.

i) CREE’s analysis identifies conflicts comparable to the RNLS-based analysis approach.

ii) CREE’s analysis with default goals distinguishes between exceptions and conflicts. The analysis infers exceptions which defeat default goals, thus the exceptions are not reported as conflicts.

iii) Stakeholders and information entities with additional constraints can be represented as specialized types in CREE models. This enables the representation of goals which affect these constrained stakeholders or information entities.

iv) Other security properties, which are not provided by CREE may be relevant for particular scenarios, e.g., trust dependency violations detected with Secure Tropos.

## 5.3 User Study

This section presents the user study we conducted to evaluate the annotation approach for extracting goal models from natural language sources provided by CREE-tool. The emphasis of the user study was to evaluate CREE-tool’s navigation and traceability features, and the ease of using CREE-tool’s annotation. The study was also designed to identify current limitations in supporting the user during annotation, i.e, user experience.

We compared extraction using CREE-tool with the RNLS approach. The RNLS sentence extraction approach was chosen for comparison because the learning curve for a new user is minimal. Although there is no specific tool for RNLS-based approach, participants in the user study used a common text editor for extracting statements.

We developed the following hypotheses, which were evaluated with data from the user study:

**H1**: Using CREE-tool, users will find it easier to navigate between their extraction and the source compared to the sentence extraction method.

**H2**: Using CREE-tool, users will find traceability from extracted elements to source better compared to the sentence extraction method.

**H3**: Using CREE-tool, users will find traceability from extracted elements to source better compared to the sentence extraction method on a large scale.

**H4**: Using CREE-tool, users will find it easier to extract elements from the source compared to the sentence extraction method.

### 5.3.1 Participants

Ten participants volunteered to participate in the user study. Participants were recruited through an email (Appendix E) sent to a mailing list. All participants were graduate students with computer science/software engineering background. There was no requirement for participants to have prior experience with semantic annotations of natural language documents. All the participants had a background in using different tools for their research, and their experiences would be valuable in providing insightful data for the study.

### 5.3.2 Procedure

We used requirements from Canada Infoway's Privacy and Security requirements [2] for the user study. The study was conducted as a lab experiment in which participants performed tasks with the RNLS method and CREE-tool.

Each participant applied the RNLS and the annotation methods to the same requirement. We used the same requirement for all the participants in order to have comparable results based on the same requirement complexity. The RNLS method involved extracting labelled sentences from the selected requirement which was provided in a non-editable format. The extracted sentences were produced in a text editor.

The steps for the study are listed in Appendix G. Each participant was required to sign a consent form (Appendix F) before beginning the study. The study consisted of two sections: the first section involved using the RNLS sentence extraction method and the second section involved annotation with CREE-tool. Each section started with a tutorial on the method and the applicable tool. The participant then performs the assigned task for the section. Participants were encouraged to verbally express their thought process (i.e., thinking aloud) while performing the tasks.

Participants were given a questionnaire (Appendix H) to fill at the end of the tasks. The questionnaire allowed the participants to rank the two methods and thus provide data to evaluate the hypotheses. The questionnaire also included questions for open-ended responses from the participants. These questions allowed the participants to give feedback on the methods.

### 5.3.3 Data Collection

The user study sessions were video and audio recorded, and data from the sentence extraction and annotation tasks were saved for later review. We used the data to score the participants for accuracy. The maximum score for each task is given as a percentage. A participant had two scores for the annotation task; a score (maximum of 100%) for annotating the concepts in the requirement and a score (maximum of 100%) for creating the associations between the identified concepts.

Data from the completed questionnaire was used for quantitative and qualitative analysis to evaluate the hypotheses.

### 5.3.4 Analysis and Results

Each of our hypotheses (**H1 - H4**) is considered an alternative hypothesis to a corresponding null hypothesis for which CREE tool is not easier/better than the sentence extraction for the corresponding criterion. We compared the sentence extraction method and CREE-tool based on these hypotheses by using one-tailed paired *t*-Test for each criterion from the questionnaire (Appendix H). The *t*-Test is used to evaluate the differences in means between two samples.

The significance level ($\alpha$) is the probability of incorrectly rejecting a hypothesis which is correct [83]. A common significance level of 0.05 was used for the null hypotheses of our tests. The p-value of a test statistic is the probability of obtaining a result at least as extreme as the one which was actually observed if the the null hypothesis is true. A null hypothesis is rejected if the p-value $\leq \alpha$.

In addition, the mean accuracy of the tasks were also compared. Responses from the open-ended questions in the questionnaire and the video recordings were analyzed to provide some insight on the results from the quantitative results.

#### 5.3.4.1 Quantitative results

Table 5.5 shows the result of the one-tailed paired *t*-Test calculated for each of the hypothesis. A criterion from the questionnaire was used for each hypothesis. For example, to test hypothesis **H1**: Using CREE-tool, users will find it easier to navigate between their extraction and the source compared to the sentence extraction method, the *t*-Test was calculated using criterion “a”: Ease of navigation from extracted sentence/model to document fragment, from the questionnaire.

| Hypothesis | Measuring criterion from questionnaire (Appendix H) | t-stat | p-value |
|---|---|---|---|
| H1 | a | -3.77 | <0.01 |
| H2 | b | -1.92 | 0.04 |
| H3 | c | -2.87 | 0.01 |
| H4 | d | 0.00 | 0.50 |

**Table 5.5.** *Hypotheses Test Results*

Hypotheses **H1, H2,** and **H3** are accepted based on the test results given in Table 5.5. This provides evidence that CREE-tool offers better navigation and traceability while extracting the structured elements from the source.

However, hypothesis **H4** is not accepted. Thus the participants did not find the extraction with CREE-tool easy. Reasons for the difficulty were revealed during analysis of the responses to the questionnaire and are discussed in Section 5.3.4.2 below.

We also calculated the mean accuracy for the tasks as shown in Table 5.6.

| *Method* | *Mean accuracy score (%)* | |
|---|---|---|
| Sentence Extraction | 95.0 | |
| Annotation with CREE-tool | Annotations | Associations |
| | 84.4 | 60.0 |

**Table 5.6.** *Mean Accuracy*

Participants performed better with the sentence extraction method. The mean accuracy score for annotations and associating annotations for all participants were 84.4% and 60% respectively. However, for participants who identified more than 2/3 of the annotations, the mean score for associating annotations was 69%. The higher value is because associations can only be created between identified annotations.

#### 5.3.4.2 Findings

i) Navigation and Traceability with CREE-tool

The study confirms the benefits offered by CREE-tool's navigation and traceability features. Reviews of the video recordings show participants used the visual representations and the annotation color codings in CREE-tool to navigate and trace elements when creating the goal representations for the semantic annotation task. These features were also useful when participants wanted to check previous annotations created during the tutorial.

Furthermore, some participants' responses to the question "*What did you find difficult with the linguistic transformation method?*" highlight the importance of providing navigation and traceability. The responses included:

- *Reordering/renumbering labels. ... Organizing the result.*
- *Very long sentence. Bit difficult to navigate.*
- *.... This will become very difficult with multiple nested statements.*
- *Pulling the atomic elements out, then tracing those elements back to the original document to derive their relationships.*

ii) CREE-tool: ease of usage

The difficulty with CREE-tool (as reflected with the test result for **H4**) can be attributed to a couple of factors.

a) Using the right concepts to annotate fragments in the source document

This can be caused by difficulty in understanding CREE meta model concepts, e.g., a participant stated: "*Determining what words are in what category*" as a difficulty with the annotation task. This might be addressed by providing more examples to illustrate the use of the concepts to the user.

The problem can also be attributed to difficulty with understanding the meaning of words used in the source, e.g., a participant stated: "*Clearly labeling a sentence fragment (some terms in the sentence are ambiguous)*" as a difficulty

with the semantic annotation task.

b) Deciding on the direction of associations

An example of this is the creation of goal associations for a *sub* relationship. Some responses to the question "*What did you find difficult with the annotation task?*" as listed below indicate this:

- *.... Understanding the relationship between goals. ....*
- *The order in which you connect the words together matters. Determining this before hand is difficult.*
- *Properly labeling connections between objects. ....*

User experience with CREE-tool can be improved with additional cognitive support. For example, the current restriction to create associations in one direction could be removed to improve flexibility. In order to associate a goal to one of its properties, e.g., object property, the goal is selected and connected to an information annotation. Enabling the association to be specified in the opposite direction, i.e., allowing the user to specify an information annotation as an object of a goal will provide additional option and make the process more flexible. Participants' also suggested CREE-tool should have additional support to create associations from the visual notations and a zooming feature for the visual representation.

Integrating better cognitive support is essential for adoption of CREE-tool. As a participant indicated, "*Could be a little more user friendly*".

iii) Accuracy with CREE-tool

Usage difficulties experienced with CREE-tool identified above contributed to the low accuracy scores for the annotation task. This is particularly noticeable for creating associations. Providing better cognitive support to improve ease of use could improve the accuracy score.

Unlike the sentence extraction task, the semantic annotation task requires decomposition of document excerpts into smaller elements (i.e., goals, actors and objects) and

associating these elements. Therefore, understanding the associations between these elements within the CREE annotation structure is critical for a high level of accuracy. Extended use of CREE-tool would help to improve the understanding and usage of CREE meta model concepts for annotating document fragments.

### 5.3.5 Threats to Validity

a) Lack of tool-support for sentence extraction

The sentence extraction task was performed with a familiar text editor because of a lack of dedicated tool for this approach. This potentially led to the better navigation and traceability results in CREE-tool. The choice of a text editor which participants were familiar with provided some mitigation for the lack of tool support.

b) Order of tasks

All the participants performed the sentence extraction task before CREE-tool's annotation task. This could cause a learning bias in favour of CREE-tool because participants become familiar with the requirement during the first task.

However, the order allowed participants to focus on using CREE-tool's features (e.g., annotation and navigation) during the second task, with a reasonable understanding of the selected requirement. Moreover, the mean accuracy of the tasks suggest the learning bias had minimal effect.

### 5.3.6 Limitations

A limitation of the study is the number of participants. More participants would improve the statistical significance of the test results. Furthermore, a larger sample size could also help in discovering other issues which the participants could raise. Another limitation with the study was that none of the participants had expertise with the domain of the requirements used. Such expertise could have mitigated the issue of misunderstanding parts of the requirement.

Participants used a text editor for the sentence extraction task. Therefore, for this task the participants had the benefit of interacting with a familiar environment and its cognitive support [129]. All the participants used CREE-tool for the first time during the study, and this contributed to the difficulty experienced with the tool and the lower accuracy of the annotation task. Extensive use of CREE-tool would improve the level of accuracy as highlighted by comments from some participants: "*it's not something which can be done with a glance.*" and "*... This would get easier with practice though*".

## 5.4 Summary

Two case studies and a user study were conducted to evaluate CREE. The case studies illustrated the feasibility of using CREE to analyze confidentiality requirements from natural language sources using goal structures extracted through annotation.

Analysis with CREE-tool is shown to be useful in detecting requirement interferences and conflicts. It was also shown to be able to distinguish between conflicts and exceptions to default requirements. Although the CREE approach does not guarantee detection of all conflicts, the comparative study (Section 5.2.5) indicates that CREE method performs at least as well as the RNLS-based conflict analysis technique.

Navigation and traceability from the visual model representation to the natural language source in CREE-tool assisted in reviewing model representations, and was useful when reviewing goals with unspecified properties as well as interfering or conflicting goals.

The user study was used to evaluate benefits of using CREE-tool's annotation feature. The study provided a means to get unbiased feedback of the tool's annotation technique in comparison to a sentence extraction method. The user study confirmed the navigation and traceability from annotated elements/structures to the source in CREE-tool enhance the extraction process. The findings from the user study indicate that better cognitive support in CREE-tool would improve the user experience and the accuracy of the annotated structure.

# Chapter 6

# Conclusion

This chapter describes contributions made by the research presented in this dissertation and the current limitations which could form the basis of future work.

## 6.1 Contributions

As summarized in Table 3.3, the different methods discussed in Chapter 3 do not support all of: defaults, exceptions and extraction of representations from natural language sources. The CREE method was developed to support analysis of confidentiality requirements with explicit representation of defaults with exception handling approach.

Unlike security modelling methods which use models from the system design perspective (Section 3.1), CREE uses goal models to analyze confidentiality requirements in order to identify stakeholder objectives and to analyze the requirements for incompleteness and conflicts during the early phase of the development process. Early detection and resolution of these defects help to avoid issues during the later phases of the development process, e.g., time and cost associated with system redesign.

The goal concepts are adopted from goal-oriented requirement engineering methods (Section 3.2). However, in contrast to other goal-oriented methods, CREE, using CREE-tool, supports goal model creation from annotations of document sources. The creation of goal models directly from document sources provides navigational links between the created goal models and the corresponding document text fragments. In addition, CREE supports default goals and goal exceptions.

The contributions from the work presented in this dissertation are discussed below.

1) **CREE meta model**

The meta model was created through a qualitative method to identify key concepts for describing confidentiality and privacy data concerns, and adoption of concepts from goal-oriented approaches. The model is used to create models to express multilateral requirements as goal models from the perspective of stakeholders.

2) **Modelling through annotation**

Direct annotation of identified concepts in natural language documents to create CREE models reduces the number of steps required to derive model elements from natural language sources in comparison to a method using initial linguistic translations and semantic parameterization of the restricted sentences. The annotation method also provides traceability and navigation between the annotated models and the text sources, a missing feature in many existing methods.

3) **Analysis to aid annotation and identify incompleteness and conflicts**

Analysis which assists in the annotation process and detecting defects was defined. Structural analysis identifies omissions during the annotation process while semantic analysis identifies potential goal interferences and conflicts. The semantic analysis also provides goal inference, using defeasible reasoning, to identify goals which are definitely and/or defeasibly inferred, as well as defeated goals.

4) **Defaults and exception representation**

The CREE meta model supports explicit representation of default goals, which can be overruled by exceptions. The defaults are modelled as goals which allow overriding goals (exceptions) from a restricted set of stakeholders.

The default goal representation enables CREE's analysis to distinguish exceptions from conflicts and to support exceptions from multi-jurisdictional stakeholders. CREE's exception handling approach is more flexible in comparison to methods which use priority-assignments for exceptions. In CREE, exceptions are specified for defaults

by indicating that the exceptions are from stakeholders who can override the defaults. Priority-assignment based methods require a global and complete set of requirements for priority (re-)assignments. The flexibility of the CREE approach is useful for supporting evolving requirements.

5) **Tool support**

A prototype tool, CREE-tool, for the CREE approach was implemented. It provides: (i) model creation through annotation (ii) visual representation of the model (iii) navigation from text fragments to the corresponding visual representation of the model (iv) analysis of the created model.

## 6.2 Limitations and Future work

Attribute inheritance by subgoals in CREE models significantly improved scalability of model representation by reducing the verbosity of goal model representations. However, a drawback of the attribute inheritance is that it can sometime produce models with unintended scenarios. As described in Section 5.2.3 these scenarios are detectable by reviewing the results of interference and conflict analysis. Furthermore, the Information concept in the current CREE meta model does not currently support the data property "entangled" and data association "inferableFrom" (Section 4.1.1.3). Integrating these into the meta model will improve the expressiveness of the goal representations and the analysis provided. Other limitations, which could be addressed by future work include:

i) CREE goal annotation step is currently performed manually with CREE-tool. Improving this step by having (semi) automatic annotation, which could at least produce initial suggestions of identified concept instances could assist the user, especially for large documents. CREE-tool is not expected to replace a competent analyst with domain expertise and legal advice where required.

ii) CREE analysis currently abstracts from temporal factors. Temporal concepts e.g., "before", "after", "during", "always" are not currently expressed in CREE models or

used in the analysis. An extension of the current CREE approach to support temporal phenomena will improve representation of the requirements. Given the current use of defeasible logic for CREE analysis, this extension would be further enabled by advancement and support for temporal defeasible logic [73, 74].

iii) Similar to temporal concepts, the CREE approach could potentially be extended to support spatial or location concepts. Confidentiality requirements may stipulate data usage based on location of access, e.g., usage of patients' medical records could be restricted to access from specific locations. Extension of CREE with location concepts would enable representation of such a location-based restriction with CREE goal models.

iv) CREE does not adequately support goals as preconditions, i.e, goals which are dependent on by other goals. Although some of these could be represented as subgoals, e.g., in R10 (Figure C.7) "*Where POS systems connected to the EHRi record a patient/persons consent directives*" is represented as a subgoal. However, the preconditions are often different from subgoals, which represent goal compositions. An explicit support of precondition goals would improve the approach.

v) CREE goals can model stakeholder obligations, however, stakeholder rights are not adequately represented. Support for concepts of obligations and rights will further improve CREE's expressiveness and analysis of confidentiality and privacy requirements.

vi) CREE-tool needs additional cognitive support to improve its usage. The findings from the controlled user study (Section 5.3.4.2) indicate that better cognitive support will provide a better user experience and improve the semantic annotation process.

## 6.3 Summary

Confidentiality requirements are usually identified from regulations, guidelines or organization policies which are specified in natural language. In addition, defaults are commonly

used to express requirements which are typically applicable, except for specific scenarios.

Although there exist methods for extracting requirements from natural language sources, many requirements engineering methods do not provide support for representing and analyzing defaults. CREE was developed to provide a method which supports default requirement representation and analysis, as well as extraction of the requirement representation through annotation of natural language sources. CREE-tool was implemented to support the method.

The case study evaluations presented illustrate the feasibility of the CREE approach. Adoption of CREE in industrial scenarios requires addressing its current limitations and improving CREE-tool. Although CREE does not support all the analysis a requirements analyst might desire, e.g., delegation and trust dependency analysis which are provided by Secure Tropos (Sections 3.2.4.1 and 5.2.5.4), CREE's goal extraction technique and analysis with defaults are features useful for early analysis of confidentiality requirements from natural language sources.

# Bibliography


[1] "Alberta Health Information Act, 2001," http://www.qp.alberta.ca/574.cfm?page=H05.cfm&leg_type=Acts&isbncln=9780779739493, accessed 26 April, 2009.

[2] "An Overview of the Electronic Health Record Privacy and Security Conceptual Architecture," http://knowledge.infoway-inforoute.ca/EHRSRA/doc/EHR-Privacy-Security-Overview.pdf, accessed 26 April, 2009.

[3] "ATLAS.ti - The Knowledge Workbench," http://www.atlasti.com/, accessed 2 March, 2009.

[4] "British Columbia Personal Information Protection Act (PIPA)," http://www.oipc.bc.ca/legislation/PIPA/PIPA(2006).pdf, accessed 26 April, 2009.

[5] "Canada Health Infoway," http://www.infoway-inforoute.ca/, accessed 4 February, 2009.

[6] "Canada Personal Information Protection and Electronic Documents Act (PIPEDA)," http://www.privcom.gc.ca/legislation/02_06_01_e.asp, accessed 26 April, 2009.

[7] "Canadian Standard Association Model Code for the Protection of Personal Information," http://www.csa.ca/standards/privacy/code/Default.asp?articleID=5286&language=English, accessed 4 February, 2009.

[8] "European Standards on Confidentiality and Privacy in Healthcare," http://www.eurosocap.org/, accessed 4 February, 2009.

[9] "Graphviz - Graph visualization software," http://www.graphviz.org/, accessed 4 February, 2009.

[10] "Health Insurance Portability and Accountability Act of 1996," http://aspe.hhs.gov/admnsimp/pl104191.htm, accessed 4 February, 2009.

[11] "ISO/IEC 27002:2005 Information technology - Security techniques - Code of practice for information security management," http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50297, accessed 30 May, 2009.

[12] "Knowtator," http://knowtator.sourceforge.net/, accessed 4 February, 2009.

[13] "Ontario Personal Health Information Protection Act, 2004," http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_04p03_e.htm, accessed 26 April, 2009.

[14] "Ontoviz," http://protegewiki.stanford.edu/index.php/OntoViz, accessed 26 April, 2009.

[15] "Privacy of Individually Identifiable Health Information. Code of Federal Regulations, 45 Part 164, Subpart E," http://frwebgate.access.gpo.gov/cgi-bin/get-cfr.cgi?YEAR=current&TITLE=45&PART=164&SECTION=520&SUBPART=&TYPE=TEXT, accessed 4 Feburary, 2009.

[16] "The Protege Ontology Editor and Knowledge Acquisition System," http://protege.stanford.edu/, accessed 4 February, 2009.

[17] "UK Data Protection Act," http://www.opsi.gov.uk/acts/acts1998/19980029.htm, accessed 4 February, 2009.

[18] "UML Profils," http://www.uml.org/#UMLProfiles, accessed 4 February, 2009.

[19] "What is Protégé-Frames?" http://protege.stanford.edu/overview/protege-frames.html, accessed 4 February, 2009.

[20] "What is Protégé-OWL?" http://protege.stanford.edu/overview/protege-owl.html, accessed 4 February, 2009.

[21] "XSB Logic Programming System," http://xsb.sourceforge.net/, accessed 5 February, 2009.

[22] "What is Bayesian Analysis?" *International Society for Bayesian Analysis*, 2006, http://www.bayesian.org/bayesexp/bayesexp.html, accessed 26 April, 2009.

[23] I. Alexander, "Initial Industrial Experience of Misuse Cases in Trade-Off Analysis," in *Proceedings IEEE Joint International Conference on Requirements Engineering*, Essen, Germany, 9-13 September 2002, pp. 61–68.

[24] A. I. Antón, "Goal-Based Requirements Analysis," in *Proceedings 2nd International Conference on Requirements Engineering*, Colorado Springs, Colorado, USA, 15 - 18 April 1996, pp. 136–144.

[25] A. I. Antón and C. Potts, "The Use of Goals to Surface Requirements for Evolving Systems," in *Proceedings 20th International Conference on Software Engineering (ICSE 1998)*, Kyoto, Japan, 19-25 April 1998, pp. 157–166.

[26] G. Antoniou and A. Bikakis, "DR-Prolog: A System for Defeasible Reasoning with Rules and Ontologies on the Semantic Web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 2, pp. 233–245, 2007.

[27] G. Antoniou, D. Billington, G. Governatori, M. J. Maher, and A. Rock, "A Family of Defeasible Reasoning Logics and its Implementation," in *Proceedings 14th European Conference on Artificial Intelligence*, W. Horn, Ed., Berlin, Germany, 20-25 August 2000, pp. 459–463.

[28] G. Antoniou, D. Billington, and M. J. Maher, "On the Analysis of Regulations using Defeasible Rules," in *Proceedings 32nd Hawaii International Conference on System Sciences*, Maui, Hawaii, USA, 5 - 8 January 1999.

[29] G. Antoniou and A. Ghose, "What is Default Reasoning Good For? Applications Revisited," in *Proceedings 32nd Hawaii International Conference on System Sciences*, Maui, Hawaii, USA, 5 - 8 January 1999.

[30] N. Y. Ashry and W. A. Taylor, "Requirements Analysis as Innovation Diffusion: A Proposed Requirements Analysis Strategy for the Development of an Integrated Hospital Information Support System," in *Proceedings 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii, USA, 4-7 January 2000, 10 pages.

[31] R. C. Barrows and P. D. Clayton, "Privacy, Confidentiality, and Electronic Medical Records," *Journal of the American Medical Informatics Association*, vol. 3, no. 2, pp. 139–148, Mar/ Apr 1996.

[32] D. Basin, J. Doser, and T. Lodderstedt, "Model Driven Security for Process-Oriented Systems," in *Proceedings 8th ACM Symposium on Access Control Models and Technologies (SACMAT '03)*. Como, Italy: ACM Press, 2003, pp. 100–109.

[33] D. Basin, J. Doser, and T. Lodderstedt, "Model Driven Security: From UML Models to Access Control Infrastructures," *ACM Transactions on Software Engineering and Methodology*, vol. 15, no. 1, pp. 39–91, 2006.

[34] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations," MITRE Corporation, Tech. Rep. M74-244, 1973.

[35] T. E. Bell and T. A. Thayer, "Software Requirements: Are They Really a Problem?" in *Proceedings 2nd International Conference on Software Engineering*, San Francisco, CA, USA, 13-15 October 1976, pp. 61–68.

[36] E. Bertino, "RBAC Models - Concepts and Trends," *Computers & Security*, vol. 22, no. 6, pp. 511–514, 2003.

[37] M. Bishop, *Computer Security: Art and Science.* Addison-Wesley, 2002.

[38] B. Boehm, P. Bose, E. Horowitz, and M. J. Lee, "Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach," in *Proceedings 17th International Conference on Software Engineering*, Seattle, WA, USA, 23-30 April 1995, pp. 243–254.

[39] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, "Using the WinWin Spiral Model: A Case Study," *IEEE Computer*, vol. 31, no. 7, pp. 33–44, July 1998.

[40] B. Boehm and H. In, "Identifying Quality Requirement Conflicts," *IEEE Software*,

vol. 13, no. 2, pp. 25–35, March 1996.

[41] B. W. Boehm, *Software Engineering Economics*. Prentice Hall, 1981.

[42] T. Borland, "Omniscience and Divine Foreknowledge," *The Internet Encyclopedia of Philosophy*, 2006, http://www.iep.utm.edu/o/omnisci.htm, accessed 26 April, 2009.

[43] T. D. Breaux and A. I. Antón, "Analyzing Goal Semantics for Rights, Permissions and Obligations," in *Proceedings 13th International Requirement Engineering Conference (RE'05)*, Paris, France, 29 August - 2 September 2005, pp. 177–186.

[44] T. D. Breaux, M. W. Vail, and A. I. Antón, "Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations," in *Proceedings 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis/St. Paul MN, USA, 11 - 15 September 2006, pp. 46–55.

[45] G. Brewka, *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, 1991.

[46] F. P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer*, vol. 20, no. 4, pp. 10–19, April 1987.

[47] B. Bryant and B. . Lee, "Two-Level Grammar as an Object-Oriented Requirements Specification Language," in *Proceedings 35th Hawaii International Conference on System Sciences (HICSS '02)*. IEEE Computer Society, 7-10 January 2002, p. 280.

[48] B. R. Bryant, "Object-Oriented Natural Language Requirement Specification," in *Proceedings 23rd Australasian Computer Science Conference (ACSC 2000)*, Canberra, Australia, 31 January - 3 February 2000, pp. 24–30.

[49] S. A. Buckovich, H. E. Rippen, and M. J. Rozen, "Driving Toward Guiding Principles: A Goal for Privacy, Confidentiality, and Security of Health Information," *Journal of American Medical Association*, vol. 6, no. 2, pp. 122–133, 1999.

[50] F. Cerbah and J. Euzenat, "Integrating Textual Knowledge and Formal Knowledge for Improving Traceability," in *Proceedings 12th International Conference Knowledge Acquisition, Modeling and Management (EKAW 2000)*, Juan-les-Pins, France, 2-6 October 2000, pp. 296–303.

[51] F. Cerbah and J. Euzenat, "Traceability between Models and Texts through Terminology," *Data and Knowledge Engineering*, vol. 38, no. 1, pp. 31–43, 7 2001.

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

[53] E. Coiera and R. Clarke, "e-Consent: The Design and Implementation of Consumer

Consent Mechanisms in an Electronic Environment," *Journal of American Medical Informatics Association*, vol. 11, no. 2, pp. 129–140, Mar/Apr 2004.

[54] K. Cooper., "Stimulus Response Requirements Specification Notation: An Empirically Evaluated Requirements Specification Notation," Ph.D. dissertation, University of British Columbia, 2001.

[55] K. Cooper and M. Ito, "Formalizing a Structured Natural Language Requirements Specification Notation," in *Proceedings 12th International Symposium of International Council on Systems Engineering (INCOSE 2002)*, Las Vegas, Nevada, USA, 28 July- 1 August 2002.

[56] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde, "GRAIL/KAOS: An Environment for Goal-driven Requirements Engineering," in *Proceedings 19th International Conference on Software Engineering*. Boston, MA, USA: ACM, 17-23 May 1997, pp. 612–613.

[57] R. De Landtsheer and A. van Lamsweerde, "Reasoning about Confidentiality at Requirements Engineering Time," in *Proceedings 10th European Software Engineering Conference/13th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/SIGSOFT FSE)*. Lisbon, Portugal: ACM, 5-9 September 2005, pp. 41–49.

[58] P. T. Devanbu and S. Stubblebine, "Software Engineering for Security: A Roadmap," in *The Future of Software Engineering, Proceedings 22nd International Conference on Software Engineering (ICSE 2000)*, A. C. W. Finkelstein, Ed. Limerick, Ireland: ACM Press, 2000, pp. 227–239.

[59] K. S. Dewitz, Y. Ryu, and M. R. Lee, "Defeasible Reasoning in Law," *Decision Support Systems*, vol. 11, no. 2, pp. 133–155, Feb 1994.

[60] B. Dick., "Grounded Theory: A Thumbnail Sketch," http://www.scu.edu.au/schools/gcm/ar/arp/grounded.html, 2005, accessed 26 April, 2009.

[61] D. Dubois, J. Lang, and H. Prade, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford Univeristy Press, 1994, vol. 3, ch. Possiblistic Logic, pp. 439–513.

[62] M. D. Fraser, K. Kumar, and V. K. Vaishnavi, "Informal and Formal Requirements Specification Languages; Bridging the Gap," *IEEE Transactions on Software Engineering*, vol. 17, no. 5, pp. 454–466, May 1991.

[63] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso, "Specifying and Analyzing Early Requirements in Tropos," *Journal of Requirement Engineering*, vol. 9, no. 2, pp. 132–150, May 2004.

[64] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso, "Model Checking Early Requirements Specifications in Tropos," in *Proceedings 9th IEEE International Requirements Engineering Conference*, Toronto, Canada, 27-31 August 2001, pp. 174–181.

[65] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu, "The Evolution of Protégé: An Environment for Knowledge-based Systems Development," *International Journal of Human-Computer Studies*, vol. 58, no. 1, pp. 89–123, 2003.

[66] L. Ginsberg, Matthew, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford Univeristy Press, 1994, vol. 3, ch. AI and Nonmonotonic Reasoning, pp. 1–33.

[67] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Formal Reasoning Techniques for Goal Models," *Journal on Data Semantics*, vol. 2800, pp. 1–20, 2003.

[68] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, "Requirements Engineering Meets Trust Management: Model, Methodology, and Reasoning," in *Proceedings 2nd International Conference on Trust Management (iTrust 2004)*, Oxford, UK, 2004, pp. 176–190.

[69] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, "Modeling Security Requirements Through Ownership, Permission and Delegation," in *Proceedings 13th IEEE International Conference on Requirements Engineering*, Paris, France, 29 August - 2 September 2005, pp. 167–176.

[70] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, "ST-Tool: A CASE Tool for Security Requirements Engineering," in *Proceedings 13th IEEE International Conference on Requirements Engineering (RE 2005)*. Paris, France: IEEE Computer Society, 29 August - 2 September 2005, pp. 451–452.

[71] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with Goal Models," in *Proceedings 21st International Conference on Conceptual Modeling (ER2002)*. Tampere, Finland: Springer Verlag, 2002.

[72] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory*. Chicago: Aldine, 1967.

[73] G. Governatori, A. Rotolo, R. Riveret, M. Palmirani, and G. Sartor, "Variants of Temporal Defeasible Logics for Modelling Norm Modifications," in *Proceedings 11th International Conference on Artificial Intelligence and Law (ICAIL '07)*, Stanford, CA, USA, 4 - 8 June 2007, pp. 155–159.

[74] G. Governatori and P. Terenziani, "Temporal Extensions to Defeasible Logic," in

*Proceedings 20th Australian Joint Conference on Artificial Intelligence*. Gold Coast, Australia: Springer Berlin / Heidelberg, 2 - 6 December 2007, pp. 476–485.

[75] S. Gürses, J. H. Jahnke, C. Obry, A. Onabajo, T. Santen, and M. Price, "Eliciting Confidentiality Requirements in Practice," in *Proceedings 15th Centers for Advanced Studies Conference (CASCON '05)*, Richmond Hill, ON, Canada, 18-20 October 2005, pp. 207–222.

[76] B. D. Haig, "Grounded Theory as Scientific Method," *Philosophy of Education*, 1995, http://www.ed.uiuc.edu/EPS/PES-Yearbook/95_docs/haig.html accessed 26 April, 2009.

[77] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," *IEEE Transactions on Software Engineering*, vol. 34, no. 1, pp. 133–153, 2008.

[78] J. H. Hammer and G. Schneider, "On the Definition and Policies of Confidentiality," in *Proceedings 3rd International Symposium on Information Assurance and Security (IAS 2007)*, Manchester, UK, 29-31 August 2007, pp. 337–342.

[79] W. Heaven and A. Finkelstein, "UML Profile to Support Requirements Engineering with KAOS," *IEE Proceedings Software*, vol. 151, no. 1, pp. 10–27, 9 February 2004.

[80] A. M. Hickey and A. M. Davis, "Elicitation Technique Selection: How Do Experts Do It?" in *Proceedings 11th International Requirements Engineering Conference*, Monterey Bay, CA, USA, 8-12 September 2003, pp. 169–178.

[81] M. Hussein and M. Zulkernine, "UMLIntr: A UML Profile for Specifying Intrusions," in *Proceedings 13th IEEE International Symposium and Workshop on Engineering of Computer Bases Systems (ECBS'06)*, Potsdam, Germany, 27-30 March 2006, pp. 279–286.

[82] B. Johnston and G. Governatori, "Induction of Defeasible Logic Theories in the Legal Domain," in *Proceedings 9th International Conference on Artificial Intelligence and Law (ICAIL '03)*. Edinburgh, Scotland, UK: ACM, 24-28 June 2003, pp. 204–213.

[83] C. M. Judd and G. H. McClelland, *Data Analysis: A Model-Comparison Approach*. Harcourt Brace College Publishers, 1989.

[84] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *Proceedings 5th International Conference of The Unified Modeling Language, UML 2002*, Dresden, Germany, 2002, pp. 412–425.

[85] J. Jürjens, "Using UMLsec and Goal Trees for Secure Systems Development," in

*Proceedings 2002 ACM Symposium on Applied Computing (SAC '02)*. Madrid, Spain: ACM Press, 2002, pp. 1026–1030.

[86] N. K. Kasabov, *Foundations of Neural Networks, Fuzzy Systems, And Knowledge Engineering*, 2nd ed. MIT Press, 1998.

[87] S. E. Keller, L. G. Kahn, and R. B. Panara, "Specifying Software Quality Requirements with Metrics," in *Tutorial: Systems and Software Requirements Engineering*, R. H. Thayer and M. Dorfman, Eds. IEEE Computer Society Press, 1990, pp. 145–163.

[88] G. Kotonya and I. Sommerville, "Viewpoints for Requirements Definition," *Software Engineering Journal*, vol. 7, no. 6, pp. 375–387, November 1992.

[89] C. Landwehr, "Formal Methods for Computer Security," *ACM Computing Surveys (CSUR)*, vol. 13, no. 3, pp. 247–278, 1981.

[90] L. Liu, E. Yu, and J. Mylopoulos, "Security and Privacy Requirements Analysis within a Social Setting," in *Proceedings 11th IEEE Requirements Engineering Conference*. Monterey Bay, CA, USA: IEEE Press, 8-12 September 2003, pp. 151–161.

[91] T. Lodderstedt, D. A. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," in *Proceedings 5th International Conference of The Unified Modeling Language, UML 2002*, Dresden, Germany, 2002, pp. 426–441.

[92] Luqi and J. Goguen, "Formal Methods: Promises and Problems," *IEEE Software*, vol. 14, no. 1, pp. 73–85, January 1997.

[93] R. R. Lutz, "Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems," in *Proceedings 1st International Symposium on Requirements Engineering*. San Diego, USA: IEEE Press, 4-6 Jan 1993, pp. 126–133.

[94] M. J. Maher, "Propositional Defeasible Logic has Linear Complexity," *Theory and Practice of Logic Programming*, vol. 1, no. 6, pp. 691–711, 2001.

[95] M.-F. Moens, "Combining Structured and Unstructured Information in a Retrieval Model for Accessing Legislation," in *Proceedings 10th International Conference on Artificial Intelligence and Law*. Bologna, Italy: ACM, 6 - 11 June 2005, pp. 141–145.

[96] H. Mouratidis, P. Giorgini, and G. A. Manson, "Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems," in *Proceedings 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)*, Klagenfurt, Austria, 16-20 June 2003, pp. 63–78.

[97] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and Using Non-Functional

Requirements: A Process-Oriented Approach," *IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Development*, vol. 18, no. 6, pp. 483–497, June 1992.

[98] B. Navarro, P. Martnez-Barco, and M. Palomar, *Natural Language Processing and Information Systems*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2005, ch. Semantic Annotation of a Natural Language Corpus for Knowledge Extraction, pp. 365–368.

[99] N. F. Noy, R. W. Fergerson, and M. A. Musen, "The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility," in *Proceedings 12th International Conference Knowledge Acquisition, Modeling and Management (EKAW 2000)*, Juan-les-Pins, France, 2 - 6 October 2000, pp. 17–32.

[100] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A RoadMap," in *The Future of Software Engineering, Proceedings 22nd International Conference on Software Engineering (ICSE 2000)*, A. C. W. Finkelstein, Ed. Limerick, Ireland: ACM Press, NY, USA, 4-11 June 2000, pp. 35–46.

[101] D. Nute, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford Univeristy Press, 1994, vol. 3, ch. Defeasible Logic, pp. 353–395.

[102] P. V. Ogren, "Knowtator: A Protégé Plug-in for Annotated Corpus Construction," in *Proceedings 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. New York, New York: Association for Computational Linguistics, Morristown, NJ, USA, 5-7 June 2006, pp. 273–275.

[103] P. O. Otto and A. I. Antón, "Addressing Legal Requirements in Requirement Engineering," in *Proceedings 15th IEEE Requirements Engineering Conference (RE'07)*, New Delhi, India, 15 - 19 October 2007, pp. 5–14.

[104] N. R. Pandit, "The Creation of Theory: A Recent Application of the Grounded Theory Method," *The Qualitative Report*, vol. 2, no. 4, December 1996, http://www.nova.edu/ssss/QR/QR2-4/pandit.html accessed 26 April, 2009.

[105] K. Pohl, "The Three Dimensions of Requirements Engineering," in *Proceedings Conference on Advanced Information Systems Engineering (CAiSE'93)*, ser. Lecture Notes in Computer Science, vol. 685. Paris, France: Springer, 8-11 June 1993, pp. 275–292.

[106] J. L. Pollock, *Reasoning: Studies of Human Inference and its Foundations*. Cambridge University Press, 2008, ch. Defeasible Reasoning, pp. 451–471.

[107] D. Poole, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 1994, vol. 3, ch. Default Logic, pp. 189–215.

[108] K. Rannenberg, "Multilateral Security A Concept and Examples for Balanced Security," in *Proceedings 2000 Workshop on New Security Paradigms*. Ballycotton, County Cork, Ireland: ACM Press New York, NY, USA, 2001, pp. 151–162.

[109] A. L. Rector, "Clinical Terminology: Why is it so Hard?" *Methods of Information in Medicine*, vol. 38, no. 4-5, pp. 239–252, Dec 1999.

[110] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements Interaction Management," *ACM Computing Surveys*, vol. 35, no. 2, pp. 132–190, 2003.

[111] W. N. Robinson and V. Volkov, "Supporting the Negotiation Life Cycle," *Communications of the ACM*, vol. 41, no. 5, pp. 95–102, 1998.

[112] G.-C. Roman, "A Taxonomy of Current Issues in Requirements Engineering," *IEEE Computer*, vol. 18, no. 4, pp. 14–23, April 1985.

[113] B. Sadan, "Patient Data Confidentiality and Patients Rights," *International Journal of Medical Informatics*, vol. 62, no. 1, pp. 41–49, 2001.

[114] A. Schürr, "Adding Graph Transformation Concepts to UML's Constraint Language OCL," *Electronic Notes in Theoretical Computer Science*, vol. 44, no. 4, pp. 93–106, July 2001.

[115] G. Sindre and A. L. Opdahl, "Eliciting Security Requirements with Misuse Cases," *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, 2005.

[116] M. T. Siponen and H. Oinas-Kukkonen, "A Review of Information Security Issues and Respective Research Contributions," *SIGMIS Database*, vol. 38, no. 1, pp. 60–80, February 2007.

[117] I. Sommerville and P. Sawyer, "Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering," *Annals of Software Engineering*, vol. 3, pp. 101–130, January 1997.

[118] R. Stegers., "From Natural Language to Formal Proof Goal," Master's thesis, Vrije Universiteit, Amsterdam, Netherlands, 2006.

[119] R. Stegers, A. ten Teije, and F. van Harmelen, "From Natural Language to Formal Proof Goal," in *Proceedings 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006)*. Podebrady, Czech Republic: Springer, 2-6 October 2006, pp. 51–58.

[120] A. Strauss and J. Corbin, *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage, 1990.

[121] A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini, "The Tropos Metamodel and its Use," *Informatica (Slovenia)*, vol. 29, no. 4, pp. 401–408, 2005.

[122] T. Tryfonas, E. Kiountouzis, and A. Poulymenakou, "Embedding Security Prac-

tices in Contemporary Information Systems Development Approaches," *Information Management and Computer Security*, vol. 9, pp. 183–197, 2001.

[123] A. van Lamsweerde, "Requirements Engineering in the year 00: A Research Perspective," in *Proceedings 22nd International Conference on Software Engineering, ICSE 00*. Limerick, Ireland: ACM Press, NY USA, 2000, pp. 5–19.

[124] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," in *Proceedings 5th IEEE International Symposium on Requirements Engineering*, Toronto, Canada, 27-31 August 2001, pp. 249–262.

[125] A. van Lamsweerde, "Elaborating Security Requirements by Construction of Intentional Anti-Models," in *Proceedings 26th International Conference on Software Engineering (ICSE 2004)*, Edinburgh, UK, 23-28 May 2004, pp. 148–157.

[126] A. van Lamsweerde, R. Darimont, and E. Letier, "Managing Conflicts in Goal-driven Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 24, no. 11, pp. 908–926, 1998.

[127] A. van Lamsweerde, "Formal Specification: A Roadmap," in *The Future of Software Engineering, Proceedings 22nd International Conference on Software Engineering (ICSE 2000)*, A. C. W. Finkelstein, Ed. Limerick, Ireland: ACM Press NY, USA, 2000, pp. 147–159.

[128] A. van Lamsweerde and E. Letier, "Handling Obstacles in Goal-Oriented Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 26, no. 10, pp. 978–1005, 2000.

[129] A. Walenstein, "Improving Adoptability by Preserving, Leveraging, and Adding Cognitive Support to Existing Tools and Environments," in *Proceedings 3rd International Workshop on Adoption-Centric Software Engineering (ACSE 2003)*, Portland, Oregon, 9 May 2003, pp. 36–41.

[130] W. M. Wilson, "Writing Effective Natural Language Requirement Specifications," *CROSSTALK - The Journal of Defense Software Engineering*, pp. 16–19, February 1999.

[131] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," in *Proceedings 3rd IEEE International Symposium on Requirements Engineering*, Annapolis, MD, USA, 6-10 Jan 1997, pp. 226–235.

[132] E. Yu and J. Mylopoulos, "Why Goal-Oriented Requirements Engineering," in *Proceedings 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, Pisa, Italy, 8-9 June 1998, pp. 15–22.

[133] P. Zave, "Classification of Research Efforts in Requirements Engineering," *ACM*

*Computing Surveys*, vol. 29, no. 4, pp. 315–321, 1997.

# Appendix A

# Canada Infoway Privacy Requirements

The following are privacy requirements of the Canadian Health Infoway's [5] Electronic Health Record Infostructure (EHRi) [2] used in the case study.

Privacy Requirement 3 Privacy Policy

Organisations connecting to the EHRi and organisations hosting components of the EHRi
must implement policies and practices, including:
a) Implementing procedures to protect PHI
b) Establishing procedures to receive and respond to privacy related complaints and inquiries;
c) Training users and communicating to users information about the organisations privacy policies and practices; and
d) Developing communications materials to explain to the general public the organisations privacy policies and practices

Privacy Requirement 4 Privacy Impact Assessments

Organisations hosting components of the EHRi, should assess, by means of a Privacy Impact Assessment, the risks to personal privacy associated with implementation of the hosted components and should implement appropriate privacy controls to mitigate identified risks. Privacy Impact Assessments should be made available

to the public upon request.

Privacy Requirement 5 Identifying Purposes for Collection, Use and Disclosure

Organisations connected to the EHRi and organisations hosting components of the EHRi must:
a) identify all the purposes for which PHI will be collected, used, and disclosed at or before the time it is collected; and
b) make a reasonable effort to inform patient/persons of these purposes, in a readily understandable manner, prior to collecting their PHI.

Privacy Requirement 7 Limitation of Collection to Identified Purposes

Organisations connecting to the EHRi or organisations hosting components of the EHRi should only collect PHI necessary to fulfill the purposes that they have identified.

Privacy Requirement 8 Obtaining Knowledgeable Consent

Except where inappropriate (e.g. specifically exempted by law or professional code of practice), organisations connecting to the EHRi, and organisations hosting components of the EHRi should obtain the knowledge and consent of each patient/person for the collection, use or disclosure of his or her PHI and where required by law, must obtain the knowledge and consent of each patient/person for the collection, use or disclosure of his or her PHI.

Privacy Requirement 9 Recording Consent in POS Systems

POS systems connected to the EHRi where required by law, must be able to record a patient/persons consent directives, including the withholding, withdrawal or revocation of consent.

Privacy Requirement 10 Associating Consent with PHI in POS Systems

Where POS systems connected to the EHRi record a patient/persons consent directives, including the withholding, withdrawal or revocation of consent, such POS systems must transmit these consent directives to the EHRi, in a consistent form, whenever they transmit the associated PHI to the EHRi.

Privacy Requirement 11 Recording Consent in the EHRi

The EHRi where required by law, must be able to record a patient/persons consent directives, including the withholding, withdrawal or revocation of consent and must be able to do so in a way that allows each jurisdiction to comply with its own legal requirements on consent.

Privacy Requirement 12 Associating Consent Directives with PHI in the EHRi

When consent is required by law, whenever receiving, storing, processing, or transmitting PHI, the EHRi must be able to:
a) maintain the association between this data and the consent

directives under which it may be used or disclosed;
b) process these consent directives before transmitting the associated data and block the transmission where it would violate the directives and where no exception for such a disclosure is outlined in law; and
c) notify the requestor whenever data is blocked as in b) above.

Privacy Requirement 13 Logging the Application of Consent Directives

The EHRi must be able to:
a) log when the processing of consent directives (cf. Privacy Requirement 12, item b) prohibits the transmission of data;
b) log the identity of any user who overrides a patient/person's consent directives, the reason for the consent override, and the date and time when the consent override occurred. and
c) alert the individual accountable for facilitating privacy compliance in the organisation where the accessing user works as well as in the organisation where the information was collected that such a consent override has occurred.

Privacy Requirement 15 Recording Identity of Substitute Decision Makers

Where required to do so by law ,the EHRi and POS systems connected to the EHRi must have the ability to indicate when consent is given on behalf of a patient/person by a substitute decision maker (e.g. consent given by an authorized representative), as well as identify this substitute decision maker and the substitute decision makers relation to the patient/person.

Privacy Requirement 18 Limiting Use and Disclosure of Personal Health Information to Identified Purposes

Organisations connecting to the EHRi and organisations hosting components of the EHRi must only use or disclose PHI for purposes consistent with those for which it was collected, except with the consent of the patient/person or as permitted or required by law.

Privacy Requirement 19 Logging Access, Modification, and Disclosure

The EHRi and POS systems connected to the EHRi must:
a) have a mechanism to record every access, modification or disclosure of PHI, together with the time and identity of the accessing user;
b) have a mechanism to record every access, modification or disclosure of provider and user registration data, together with the time and identity of the accessing user; and
c) where required by law, have mechanisms to alert the organisations individual accountable for privacy when it is suspected that PHI has been accessed, used or disclosed inappropriately.

Privacy Requirement 21 Retaining Records

The EHRi, POS systems connected to the EHRi, organisations connecting to the EHRi, and organisations hosting components of the EHRi:
a) must retain PHI in accordance with record-keeping requirements

outlined in legislation; and
b) should develop guidelines and implement procedures with respect to the retention of PHI, including minimum and maximum retention periods.

Privacy Requirement 22 Accuracy

The EHRi, POS systems connected to the EHRi, organisations connecting to the EHRi and organisations hosting components of the EHRi must take reasonable steps or make a reasonable effort to:
a) ensure that PHI is as accurate, complete, and up-to-date as is necessary for the purposes for which it is to be used, including disclosures of PHI to third parties; and
b) accurately identify a patient/person when accessing or modifying his or her PHI

Privacy Requirement 22a Denoting Patients/Persons At Elevated Risk

The EHRi must provide functions for marking records of selected patients/persons and subsequently making accesses to such data subject to mandatory auditing by the individual accountable for privacy compliance in the organisation.

Privacy Requirement 24 Patient/Person Access

Organisations connecting to the EHRi and organisations hosting components of the EHRi must, upon request:

a) inform a patient/person of the existence, use and disclosure of his or her PHI and shall give the patient/person direct access to that PHI where such access is not prohibited by legislation;
b) respond to requests for access to a patient/persons PHI within a reasonable time and make it available in a form that is generally understandable; and
c) allow a patient/person to challenge the accuracy and completeness of his or her PHI and have it amended as appropriate.

Privacy Requirement 25 Amending Inaccurate or Incomplete Information

Organisations connecting to the EHRi and organisations hosting components of the EHRi should:
a) amend PHI when a patient/person successfully demonstrates the inaccuracy or incompleteness of this information;
b) notify EHRi users that have accessed the information in question that the information has been amended when the amended information can reasonably be expected to have effect on the ongoing treatment of the patient/person;
c) record the substance of the unresolved challenge when the organisation disagrees with the patient/person's assessment of incompleteness or inaccuracy; and
d) transmit the existence of the unresolved challenge to EHRi users accessing the information in question.

Privacy Requirement 27 Complaint Procedures

Organisations connecting to the EHRi and organisations hosting components of the EHRi must
a) put easily accessible and simple-to-use procedures in place

to receive and respond to complaints or inquiries about their policies and practices relating to the handling of PHI;

b) inform complainants and inquirers of the existence of these procedures

c) treat all received complaints as confidential.

# Appendix B

# RNLS Translations

The following are RNLS translations of the privacy requirements of Appendix A. Labels for the RNLS are based on the numeric labels of the privacy requirements, e.g., RNLS_R3_1 is an RNLS for privacy Requirement 3. The following abbreviations are used:

- org.: organisation
- EHRi: Electronic Health Record Infostructure
- PHI: Personal Health Information
- POS: Point of Service

The privacy requirements can be read by starting with RNLS marked with an asterisk (*).

RNLS_R3_1: org. connecting to EHRi protects PHI

RNLS_R3_2: org. connecting to EHRi receives privacy related complaints

RNLS_R3_3: org. connecting to EHRi receives privacy related inquiries

RNLS_R3_4: org. connecting to EHRi responds to privacy related complaints

RNLS_R3_5: org. connecting to EHRi responds to privacy related inquiries

RNLS_R3_6*: org. connecting to EHRi trains users on organisation's privacy policies

RNLS_R3_7*: org. connecting to EHRi trains users on organisation's privacy practices

RNLS_R3_8*: org. connecting to EHRi communicates to users the organisation's privacy policies

RNLS_R3_9*: org. connecting to EHRi communicates to users the organisation's privacy practices

RNLS_R3_10: org. connecting to EHRi explains to the public the organisation's privacy policies

RNLS_R3_11: org. connecting to EHRi explains to the public the organisation's privacy practices

RNLS_R3_12*: org. connecting to EHRi develops communication materials to RNLS_R3_10

RNLS_R3_13*: org. connecting to EHRi develops communication materials to RNLS_R3_11

RNLS_R3_14*: org. connecting to EHRi must implement procedures for RNLS_R3_1

RNLS_R3_15*: org. connecting to EHRi must establish procedures for RNLS_R3_2

RNLS_R3_16*: org. connecting to EHRi must establish procedures for RNLS_R3_3

RNLS_R3_17*: org. connecting to EHRi must establish procedures for RNLS_R3_4

RNLS_R3_18*: org. connecting to EHRi must establish procedures for RNLS_R3_5

RNLS_R3_19: org. connecting to EHRi protects PHI

RNLS_R3_20: org. connecting to EHRi receives privacy related complaints

RNLS_R3_21: org. connecting to EHRi receives privacy related inquiries

RNLS_R3_22: org. connecting to EHRi responds to privacy related complaints

RNLS_R3_23: org. connecting to EHRi responds to privacy related inquiries

RNLS_R3_24*: org. connecting to EHRi trains users on organisation's privacy policies

RNLS_R3_25*: org. connecting to EHRi communicates to users the organisation's privacy policies

RNLS_R3_26: org. connecting to EHRi explains to the public the organisation's privacy policies

RNLS_R3_27: org. connecting to EHRi explains to the public the organisation's privacy practices

RNLS_R3_28*: org. connecting to EHRi develops communication materials to RNLS_R3_26

RNLS_R3_29*: org. connecting to EHRi develops communication materials to RNLS_R3_27

RNLS_R3_30*: org. connecting to EHRi must implement procedures for RNLS_R3_19

RNLS_R3_31*: org. connecting to EHRi must establish procedures for RNLS_R3_20

RNLS_R3_32*: org. connecting to EHRi must establish procedures for RNLS_R3_21

RNLS_R3_33*: org. connecting to EHRi must establish procedures for RNLS_R3_22

RNLS_R3_34*: org. connecting to EHRi must establish procedures for RNLS_R3_23

Same applies to org. hosting components of the EHRi

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R4_1*: org. hosting components of the EHRi should assess the risks to personal privacy by means of a Privacy Impact Assessment

RNLS_R4_2*: org hosting components of the EHRi should implement appropriate privacy controls to mitigate identified risks

RNLS_R4_3*: org. hosting components of the EHRi should make available Privacy Impact Assessments available to the public upon request

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R5_1: org. connecting to EHRi identifies all the purposes

RNLS_R5_2: org. connecting to EHRi informs patient/person of purpose

RNLS_R5_3: org. connecting to EHRi collects PHI

RNLS_R5_4: org. connecting to EHRi uses PHI

RNLS_R5_5: org. connecting to EHRi discloses PHI

RNLS_R5_6*: org. connecting to EHRi must RNLS_R5_1 when RNLS_R5_3

RNLS_R5_7*: org. connecting to EHRi must RNLS_R5_1 when RNLS_R5_4

RNLS_R5_8*: org. connecting to EHRi must RNLS_R5_1 when RNLS_R5_5

RNLS_R5_9*: org. connecting to EHRi must RNLS_R5_1 before RNLS_R5_3

RNLS_R5_10*: org. connecting to EHRi must RNLS_R5_1 before RNLS_R5_4

RNLS_R5_11*: org. connecting to EHRi must RNLS_R5_1 before RNLS_R5_5

RNLS_R5_12*: org. connecting to EHRi must RNLS_R5_2 before RNLS_R5_3

RNLS_R5_13*: org. connecting to EHRi must RNLS_R5_2 before RNLS_R5_4

RNLS_R5_14*: org. connecting to EHRi must RNLS_R5_2 before RNLS_R5_5

RNLS_R5_15: org. hosting components of the EHRi identifies all the purposes

RNLS_R5_16: org. hosting components of the EHRi informs patient/person of purpose

RNLS_R5_17: org. hosting components of the EHRi collects PHI

RNLS_R5_18: org. hosting components of the EHRi uses PHI

RNLS_R5_19: org. hosting components of the EHRi discloses PHI

RNLS_R5_20*: org. hosting components of the EHRi must RNLS_R5_15 when RNLS_R5_17

RNLS_R5_21*: org. hosting components of the EHRi must RNLS_R5_15 when RNLS_R5_18

RNLS_R5_22*: org. hosting components of the EHRi must RNLS_R5_15 when RNLS_R5_19

RNLS_R5_23*: org. hosting components of the EHRi must RNLS_R5_16 before RNLS_R5_17

RNLS_R5_24*: org. hosting components of the EHRi must RNLS_R5_16 before RNLS_R5_18

RNLS_R5_25*: org. hosting components of the EHRi must RNLS_R5_16 before RNLS_R5_19

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R7_1: org. connecting to EHRi fulfills identified purposes

RNLS_R7_2: org. connecting to EHRi collects PHI

RNLS_R7_ 3*: if RNLS_R7_1 then RNLS_R7_2

RNLS_R7_ 4: org. hosting components of the EHRi fulfills identified purposes

RNLS_R7_ 5: org. hosting components of the EHRi collect PHI

RNLS_R7_ 6*: if RNLS_R7_4 then RNLS_R7_5

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R8_1: org. connecting to EHRi collects PHI

RNLS_R8_2: org. connecting to EHRi uses PHI

RNLS_R8_3: org. connecting to EHRi discloses PHI

RNLS_R8_4: org. connecting to EHRi obtains patient/person's knowledge for RNLS_R8_1

RNLS_R8_5: org. connecting to EHRi obtains patient/person's knowledge for RNLS_R8_2

RNLS_R8_6: org. connecting to EHRi obtains patient/person's knowledge for RNLS_R8_3

RNLS_R8_7: org. connecting to EHRi obtains patient/person's consent for RNLS_R8_1

RNLS_R8_8: org. connecting to EHRi obtains patient/person's consent for RNLS_R8_2

RNLS_R8_9: org. connecting to EHRi obtains patient/person's consent for RNLS_R8_3

RNLS_R8_10: RNLS_R8_4 is inappropriate

RNLS_R8_11: RNLS_R8_5 is inappropriate

RNLS_R8_12: RNLS_R8_6 is inappropriate

RNLS_R8_13: RNLS_R8_7 is inappropriate

RNLS_R8_14: RNLS_R8_8 is inappropriate

RNLS_R8_15: RNLS_R8_9 is inappropriate

RNLS_R8_16*: if not RNLS_R8_10 then RNLS_R8_4

RNLS_R8_17*: if not RNLS_R8_11 then RNLS_R8_5

RNLS_R8_18*: if not RNLS_R8_12 then RNLS_R8_6

RNLS_R8_19*: if not RNLS_R8_13 then RNLS_R8_7

RNLS_R8_20*: if not RNLS_R8_14 then RNLS_R8_8

RNLS_R8_21*: if not RNLS_R8_15 then RNLS_R8_9

RNLS_R8_22: law requires RNLS_R8_4

RNLS_R8_23: law requires RNLS_R8_5

RNLS_R8_24: law requires RNLS_R8_6

RNLS_R8_25: law requires RNLS_R8_7

RNLS_R8_26: law requires RNLS_R8_8

RNLS_R8_27: law requires RNLS_R8_9

RNLS_R8_28*: if RNLS_R8_22 then RNLS_R8_4

RNLS_R8_29*: if RNLS_R8_23 then RNLS_R8_5

RNLS_R8_30*: if RNLS_R8_24 then RNLS_R8_6

RNLS_R8_31*: if RNLS_R8_25 then RNLS_R8_7

RNLS_R8_32*: if RNLS_R8_26 then RNLS_R8_8

RNLS_R8_33*: if RNLS_R8_27 then RNLS_R8_9

RNLS_R8_34: org. hosting components of the EHRi collects PHI

RNLS_R8_35: org. hosting components of the EHRi uses PHI

RNLS_R8_36: org. hosting components of the EHRi discloses PHI

RNLS_R8_37: org. hosting components of the EHRi obtains patient/person's knowledge for RNLS_R8_34

RNLS_R8_38: org. hosting components of the EHRi obtains patient/person's knowledge for RNLS_R8_35

RNLS_R8_39: org. hosting components of the EHRi obtains patient/person's knowledge for RNLS_R8_36

RNLS_R8_40: org. hosting components of the EHRi obtains patient/person's consent for RNLS_R8_34

RNLS_R8_41: org. hosting components of the EHRi obtains patient/person's consent for RNLS_R8_35

RNLS_R8_42: org. hosting components of the EHRi obtains patient/person's consent for RNLS_R8_36

RNLS_R8_43: RNLS_R8_37 is inappropriate

RNLS_R8_44: RNLS_R8_38 is inappropriate

RNLS_R8_45: RNLS_R8_39 is inappropriate

RNLS_R8_46: RNLS_R8_40 is inappropriate

RNLS_R8_47: RNLS_R8_41 is inappropriate

RNLS_R8_48: RNLS_R8_42 is inappropriate

RNLS_R8_49*: if not RNLS_R8_43 then RNLS_R8_37

RNLS_R8_50*: if not RNLS_R8_44 then RNLS_R8_38

RNLS_R8_51*: if not RNLS_R8_45 then RNLS_R8_39

RNLS_R8_52*: if not RNLS_R8_46 then RNLS_R8_40

RNLS_R8_53*: if not RNLS_R8_47 then RNLS_R8_41

RNLS_R8_54*: if not RNLS_R8_48 then RNLS_R8_42

RNLS_R8_55: law requires RNLS_R8_37

RNLS_R8_56: law requires RNLS_R8_38

RNLS_R8_57: law requires RNLS_R8_39

RNLS_R8_58: law requires RNLS_R8_40

RNLS_R8_59: law requires RNLS_R8_41

RNLS_R8_60: law requires RNLS_R8_42

RNLS_R8_61*: if RNLS_R8_55 then RNLS_R8_37

RNLS_R8_62*: if RNLS_R8_56 then RNLS_R8_38

RNLS_R8_63*: if RNLS_R8_57 then RNLS_R8_39

RNLS_R8_64*: if RNLS_R8_58 then RNLS_R8_40

RNLS_R8_65*: if RNLS_R8_59 then RNLS_R8_41

RNLS_R8_66*: if RNLS_R8_60 then RNLS_R8_42

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R9_1: POS systems connected to the EHRi record a patient's consent directives

RNLS_R9_2: withholding of consent is a consent directive

RNLS_R9_3: withdrawal of a consent is a consent directive

RNLS_R9_4: revocation of a consent is a consent directive

RNLS_R9_5: law requires RNLS_R9_1

RNLS_R9_6*: if RNLS_R9_5 then RNLS_R9_1

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R10_1: POS systems connected to the EHRi record a patient/person's consent directive

RNLS_R10_2: withholding a consent is a consent directive

RNLS_R10_3: withdrawal of consent is a consent directive

RNLS_R10_4: revocation of a consent is a consent directive

RNLS_R10_5: POS systems connected to the EHRi transmit PHI to the EHRi

RNLS_R10_6: POS systems connected to the EHRi must transmit consent directives to the EHRi

RNLS_R10_7*: if RNLS_R10_1 and RNLS_R10_5 then RNLS_R10_6

++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R11_1: EHRi must be able to record a patient/person's consent directives

RNLS_R11_2: withholding a consent is a consent directive

RNLS_R11_3: withdrawal of consent is a consent directive

RNLS_R11_4: revocation of a consent is a consent directive

RNLS_R11_5: law requires RNLS_R11_1

RNLS_R11_6: Jurisdiction complies with the legal requirements on consent

RNLS_R11_7*: If RNLS_R11_5 then RNLS_R11_1

RNLS_R11_8*: If RNLS_R11_5 then RNLS_R11_6

++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R12_1: law requires consent

RNLS_R12_2: EHRi receives PHI

RNLS_R12_3: EHRi stores PHI

RNLS_R12_4: EHRi processes PHI

RNLS_R12_5: EHRi transmits PHI

RNLS_R12_6: RNLS_R12_1 for RNLS_R12_2

RNLS_R12_7: RNLS_R12_1 for RNLS_R12_3

RNLS_R12_8: RNLS_R12_1 for RNLS_R12_4

RNLS_R12_9: RNLS_R12_1 for RNLS_R12_5

RNLS_R12_10: EHRi must be able to maintain association between PHI and usage consent directives

RNLS_R12_11: EHRi must be able to maintain association between PHI and disclosure consent directives

RNLS_R12_12*: if RNLS_R12_6 then RNLS_R12_10
RNLS_R12_13*: if RNLS_R12_7 then RNLS_R12_10
RNLS_R12_14*: if RNLS_R12_8 then RNLS_R12_10
RNLS_R12_15*: if RNLS_R12_9 then RNLS_R12_10
RNLS_R12_16*: if RNLS_R12_6 then RNLS_R12_11
RNLS_R12_17*: if RNLS_R12_7 then RNLS_R12_11
RNLS_R12_18*: if RNLS_R12_8 then RNLS_R12_11
RNLS_R12_19*: if RNLS_R12_9 then RNLS_R12_11

RNLS_R12_20: EHRi must be able to process consent directives before RNLS_R12_5

RNLS_R12_21*: if RNLS_R12_6 then RNLS_R12_20
RNLS_R12_22*: if RNLS_R12_7 then RNLS_R12_20
RNLS_R12_23*: if RNLS_R12_8 then RNLS_R12_20
RNLS_R12_24*: if RNLS_R12_9 then RNLS_R12_20

RNLS_R12_25: RNLS_R12_5 violates consent directives
RNLS_R12_26: EHRi discloses PHI
RNLS_R12_27: law permits RNLS_R12_26

RNLS_R12_28*: if RNLS_R12_6 and RNLS_R12_25 and not RNLS_R12_27 then prohibit RNLS_R12_5
RNLS_R12_29*: if RNLS_R12_7 and RNLS_R12_25 and not RNLS_R12_27 then prohibit RNLS_R12_5
RNLS_R12_30*: if RNLS_R12_8 and RNLS_R12_25 and not RNLS_R12_27 then prohibit RNLS_R12_5
RNLS_R12_31*: if RNLS_R12_9 and RNLS_R12_25 and not RNLS_R12_27 then prohibit RNLS_R12_5

RNLS_R12_32: EHRi notifies requestor

RNLS_R12_33*: if RNLS_R12_28 then RNLS_R12_32

RNLS_R12_34*: if RNLS_R12_29 then RNLS_R12_32

RNLS_R12_35*: if RNLS_R12_30 then RNLS_R12_32

RNLS_R12_36*: if RNLS_R12_31 then RNLS_R12_32

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R13_1: EHRi processes consent directives

RNLS_R13_2: EHRi transmits PHI

RNLS_R13_3: RNLS_R13_1 prohibits RNLS_R13_2

RNLS_R13_4: EHRi must be able to log RNLS_R13_3

RNLS_R13_5*: if RNLS_R13_3 then RNLS_R13_4

RNLS_R13_6: user overrides a patient/person's consent directives

RNLS_R13_7: EHRi must be able to log user's identity

RNLS_R13_8: EHRi must be able to log reason for RNLS_R13_6

RNLS_R13_9: EHRi must be able to log date for RNLS_R13_6

RNLS_R13_10: EHRi must be able to log time for RNLS_R13_6

RNLS_R13_11*: if RNLS_R13_6 then RNLS_R13_7

RNLS_R13_12*: if RNLS_R13_6 then RNLS_R13_8

RNLS_R13_13*: if RNLS_R13_6 then RNLS_R13_9

RNLS_R13_14*: if RNLS_R13_6 then RNLS_R13_10

RNLS_R13_15: EHRi must be able to alert privacy compliance officer of org. collecting data

RNLS_R13_16: EHRi must be able to alert privacy compliance officer of org. employing user

RNLS_R13_17*: if RNLS_R13_6 then RNLS_R13_15

RNLS_R13_18*: if RNLS_R13_6 then RNLS_R13_16

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R15_1: Substitute decision maker provides consent

RNLS_R15_2: Authorized representative isa substitute decision maker

RNLS_R15_3: EHRi must have the ability to identify RNLS_R15_1

RNLS_R15_4: POS connected to the EHRi must have the ability to identity RNLS_R15_1

RNLS_R15_5: EHRi must have the ability to identify substitute decision maker

RNLS_R15_6: POS connected to the EHRi must have the ability to identity substitute decision maker

RNLS_R15_7: EHRi must have the ability to identify relationship between patient/person and substitute decision maker

RNLS_R15_8: POS connected to the EHRi must have the ability to identity relationship between patient/person and substitute decision maker

RNLS_R15_9: law requires RNLS_R15_3

RNLS_R15_10: law requires RNLS_R15_4

RNLS_R15_11: law requires RNLS_R15_5

RNLS_R15_12: law requires RNLS_R15_6

RNLS_R15_13: law requires RNLS_R15_7

RNLS_R15_14: law requires RNLS_R15_8

RNLS_R15_15*: if RNLS_R15_9 then RNLS_R15_3

RNLS_R15_16*: if RNLS_R15_10 then RNLS_R15_4

RNLS_R15_17*: if RNLS_R15_11 then RNLS_R15_5

RNLS_R15_18*: if RNLS_R15_12 then RNLS_R15_6

RNLS_R15_19*: if RNLS_R15_13 then RNLS_R15_7

RNLS_R15_20*: if RNLS_R15_14 then RNLS_R15_8

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R18_1: org. connecting to the EHRi collect PHI for identified purpose

RNLS_R18_2: org. hosting components of the EHRi collect PHI for identified purpose

RNLS_R18_3: org. connecting to the EHRi use PHI for identified purposes

RNLS_R18_4: org. hosting components of the EHRi use PHI for identified purposes

RNLS_R18_5: org. connecting to the EHRi disclose PHI for identified purposes

RNLS_R18_6: org. hosting components of the EHRi disclose PHI for identified purposes

RNLS_R18_7: org. connecting to the EHRi use PHI for unidentified purposes

RNLS_R18_8: org. hosting components of the EHRi use PHI for unidentified purposes

RNLS_R18_9: org. connecting to the EHRi disclose PHI for unidentified purposes

RNLS_R18_10: org. hosting components of the EHRi disclose PHI for unidentified purposes

RNLS_R18_11: patient/person permits RNLS_R18_7

RNLS_R18_12: patient/person permits RNLS_R18_8

RNLS_R18_13: patient/person permits RNLS_R18_9

RNLS_R18_14: patient/person permits RNLS_R18_10

RNLS_R18_15: law permits RNLS_R18_7

RNLS_R18_16: law permits RNLS_R18_8

RNLS_R18_17: law permits RNLS_R18_9

RNLS_R18_18: law permits RNLS_R18_10

RNLS_R18_19: law requires RNLS_R18_7

RNLS_R18_20: law requires RNLS_R18_8

RNLS_R18_21: law requires RNLS_R18_9

RNLS_R18_22: law requires RNLS_R18_10

RNLS_R18_23*: if not RNLS_R18_11 then/do RNLS_R18_3

RNLS_R18_24*: if not RNLS_R18_12 then/do RNLS_R18_4

RNLS_R18_25*: if not RNLS_R18_13 then/do RNLS_R18_5

RNLS_R18_26*: if not RNLS_R18_14 then/do RNLS_R18_6

RNLS_R18_27*: if not RNLS_R18_15 then/do RNLS_R18_3

RNLS_R18_28*: if not RNLS_R18_16 then/do RNLS_R18_4

RNLS_R18_29*: if not RNLS_R18_17 then/do RNLS_R18_5

RNLS_R18_30*: if not RNLS_R18_18 then/do RNLS_R18_6

RNLS_R18_31*: if not RNLS_R18_19 then/do RNLS_R18_3

RNLS_R18_32*: if not RNLS_R18_20 then/do RNLS_R18_4

RNLS_R18_33*: if not RNLS_R18_21 then/do RNLS_R18_5

RNLS_R18_34*: if not RNLS_R18_22 then/do RNLS_R18_6

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R19_1: User accesses PHI

RNLS_R19_2: User modifies PHI

RNLS_R19_3: User discloses PHI

RNLS_R19_4: User accesses provider registration data

RNLS_R19_5: User modifies provider registration data

RNLS_R19_6: User discloses provider registration data

RNLS_R19_7: User accesses user registration data

RNLS_R19_8: User modifies user registration data

RNLS_R19_9: User discloses user registration data

RNLS_R19_10: EHRi must have mechanism to record user identity

RNLS_R19_11: EHRi must have mechanism to record time of access

RNLS_R19_12*: EHRi must have mechanism to record RNLS_R19_1, RNLS_R19_10 and RNLS_R19_11

RNLS_R19_13*: EHRi must have mechanism to record RNLS_R19_2 ,RNLS_R19_10 and RNLS_R19_11

RNLS_R19_14*: EHRi must have mechanism to record RNLS_R19_3, RNLS_R19_10 and RNLS_R19_11

RNLS_R19_15*: EHRi must have mechanism to record RNLS_R19_4, RNLS_R19_10 and RNLS_R19_11

RNLS_R19_16*: EHRi must have mechanism to record RNLS_R19_5, RNLS_R19_10 and RNLS_R19_11

RNLS_R19_17*: EHRi must have mechanism to record RNLS_R19_6, RNLS_R19_10 and RNLS_R19_11

RNLS_R19_18*: EHRi must have mechanism to record RNLS_R19_7 , RNLS_R19_10 and RNLS_R19_11

RNLS_R19_19*: EHRi must have mechanism to record RNLS_R19_8, RNLS_R19_10 and RNLS_R19_11

RNLS_R19_20*: EHRi must have mechanism to record RNLS_R19_9 , RNLS_R19_10 and RNLS_R19_11

RNLS_R19_21*: POS systems connected to EHR must have mechanism to record user identity

RNLS_R19_22*: POS systems connected to EHR must have mechanism to record time of access

RNLS_R19_23*: POS systems connected to EHRi must have mechanism to record RNLS_R19_1, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_24*: POS systems connected to EHRi must have mechanism to record RNLS_R19_2, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_25*: POS systems connected to EHRi must have mechanism to record RNLS_R19_3, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_26*: POS systems connected to EHRi must have mechanism to record RNLS_R19_4, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_27*: POS systems connected to EHRi must have mechanism to record RNLS_R19_5, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_28*: POS systems connected to EHRi must have mechanism to record RNLS_R19_6, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_29*: POS systems connected to EHRi must have mechanism to record RNLS_R19_7, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_30*: POS systems connected to EHRi must have mechanism to record RNLS_R19_8, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_31*: POS systems connected to EHRi must have mechanism to record RNLS_R19_9, RNLS_R19_21 and RNLS_R19_22

RNLS_R19_32: User accesses PHI inappropriately

RNLS_R19_33: User modifies PHI inappropriately

RNLS_R19_34: User discloses PHI inappropriately

RNLS_R19_35: EHRi alerts privacy compliance officer of org.

RNLS_R19_36: if RNLS_R19_32 then RNLS_R19_35

RNLS_R19_37: if RNLS_R19_33 then RNLS_R19_35

RNLS_R19_38: if RNLS_R19_34 then RNLS_R19_35

RNLS_R19_39: Law requires RNLS_R19_36

RNLS_R19_40: Law requires RNLS_R19_37

RNLS_R19_41: Law requires RNLS_R19_38

RNLS_R19_42: EHRi has mechanism for RNLS_R19_35

RNLS_R19_43*: if RNLS_R19_39 then RNLS_R19_42

RNLS_R19_44*: if RNLS_R19_40 then RNLS_R19_42

RNLS_R19_45*: if RNLS_R19_41 then RNLS_R19_42

RNLS_R19_46: POS systems connected to EHRi alerts privacy compliance officer of org.

RNLS_R19_47: if RNLS_R19_32 then RNLS_R19_46

RNLS_R19_48: if RNLS_R19_33 then RNLS_R19_46

RNLS_R19_49: if RNLS_R19_34 then RNLS_R19_46

RNLS_R19_50: Law requires RNLS_R19_47

RNLS_R19_51: Law requires RNLS_R19_48

RNLS_R19_52: Law requires RNLS_R19_49

RNLS_R19_53: POS systems connected to EHRi has mechanism for RNLS_R19_46

RNLS_R19_54*: if RNLS_R19_50 then RNLS_R19_53

RNLS_R19_55*: if RNLS_R19_51 then RNLS_R19_53

RNLS_R19_56*: if RNLS_R19_52 then RNLS_R19_53

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R21_1: EHRi retains PHI

RNLS_R21_2: POS systems connected to the EHRi retains PHI

RNLS_R21_3: org. connecting the EHRi retains PHI

RNLS_R21_4: org. hosting components of EHRi retains PHI

RNLS_R21_5: legislation requires RNLS_R21_1

RNLS_R21_6: legislation requires RNLS_R21_2

RNLS_R21_7: legislation requires RNLS_R21_3

RNLS_R21_8: legislation requires RNLS_R21_4

RNLS_R21_9*: if RNLS_R21_5 then RNLS_R21_1

RNLS_R21_10*: if RNLS_R21_6 then RNLS_R21_2

RNLS_R21_11*: if RNLS_R21_7 then RNLS_R21_3

RNLS_R21_12*: if RNLS_R21_8 then RNLS_R21_4

RNLS_R21_13: min retention period is a guideline

RNLS_R21_14: max retention period is a guideline

RNLS_R21_15*: EHRi develops guidelines for PHI retention

RNLS_R21_16*: POS systems connected to the EHRi develops guidelines for PHI retention

RNLS_R21_17*: org. connecting the EHRi develops guidelines for PHI retention

RNLS_R21_18*: org. hosting components of EHRi develops guidelines for PHI retention

RNLS_R21_19*: EHRi implements procedures for PHI retention

RNLS_R21_20*: POS systems connected to the EHRi implements procedures for PHI retention

RNLS_R21_21*: org. connecting the EHRi implements procedures for PHI retention

RNLS_R21_22*: org. hosting components of EHRi implements procedures for PHI retention

++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R22_1: disclosure to third parties is a purpose

RNLS_R22_2*: EHRi take reasonable steps to ensure PHI is accurate

RNLS_R22_3*: POS systems connected to the EHRi take reasonable steps to ensure PHI is accurate

RNLS_R22_4*: org. connecting the EHRi take reasonable steps to ensure PHI is accurate

RNLS_R22_5*: org. hosting components of EHRi take reasonable steps to ensure PHI is accurate

RNLS_R22_6*: EHRi make reasonable effort to ensure PHI is complete

RNLS_R22_7*: POS systems connected to the EHRi make reasonable effort to ensure PHI is complete

RNLS_R22_8*: org. connecting the EHRi make reasonable effort to ensure PHI is complete

RNLS_R22_9*: org. hosting components of EHRi make reasonable effort to ensure PHI is complete

RNLS_R22_10*: EHRi make reasonable effort to ensure PHI is up-to-date

RNLS_R22_11*: POS systems connected to the EHRi make reasonable effort to ensure PHI is up-to-date

RNLS_R22_12*: org. connecting the EHRi make reasonable effort to ensure PHI is up-to-date

RNLS_R22_13*: org. hosting components of EHRi make reasonable effort to ensure PHI is up-to-date

RNLS_R22_14: EHRi access patient/person's PHI

RNLS_R22_15: POS systems connected to the EHRi access patient/person's PHI

RNLS_R22_16: org. connecting the EHRi access patient/person's PHI

RNLS_R22_17: org. hosting components of EHRi access patient/person's PHI

RNLS_R22_18: EHRi modify patient/person's PHI

RNLS_R22_19: POS systems connected to the EHRi modify patient/person's PHI

RNLS_R22_20: org. connecting the EHRi modify patient/person's PHI

RNLS_R22_21: org. hosting components of EHRi modify patient/person's PHI

RNLS_R22_22*: EHRi take reasonable steps to identify patient/person for RNLS_R22_14

RNLS_R22_23*: POS systems connected to the EHRi take reasonable steps to identify patient/person for RNLS_R22_15

RNLS_R22_24*: org. connecting the EHRi take reasonable steps to identify patient/person for RNLS_R22_16

RNLS_R22_25*: org. hosting components of EHRi take reasonable steps to identify patient/person for RNLS_R22_17

RNLS_R22_26*: EHRi take reasonable steps to identify patient/person for RNLS_R22_18

RNLS_R22_27*: POS systems connected to the EHRi take reasonable steps to identify patient/person for RNLS_R22_19

RNLS_R22_28*: org. connecting the EHRi take reasonable steps to identify patient/person for RNLS_R22_20

RNLS_R22_29*: org. hosting components of EHRi take reasonable steps to identify patient/person for RNLS_R22_21

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R22a_1*: EHRi provides functions to mark selected patients/persons records PHIs

RNLS_R22a_2: User accesses selected patients/persons records PHIs

RNLS_R22a_3*: Privacy compliance person audits RNLS_22a_2

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R24_1: patient/person requests information on existence of PHI

RNLS_R24_2: patient/person requests information on use of PHI

RNLS_R24_3: patient/person requests information on disclosure of PHI

RNLS_R24_4: org. connecting to the EHRi inform patient/person of existence of PHI

RNLS_R24_5: org. connecting to the EHRi inform patient/person of use of PHI

RNLS_R24_6: org. connecting to the EHRi inform patient/person of disclosure of PHI

RNLS_R24_7: org. hosting components of the EHRi inform patient/person of existence of PHI

RNLS_R24_8: org. hosting components of the EHRi inform patient/person of use of PHI

RNLS_R24_9: org. hosting components of the EHRi inform patient/person of disclosure of PHI

RNLS_R24_10: patient/person accesses PHI

RNLS_R24_11: org. connecting to the EHRi provides RNLS_24_10

RNLS_R24_12: org. hosting components of the EHRi provides RNLS_24_10

RNLS_R24_13: Legislation prohibits RNLS_24_10

RNLS_R24_14: if not RNLS_24_13 then RNLS_24_11

RNLS_R24_15: if not RNLS_24_13 then RNLS_24_12

RNLS_R24_16*: if RNLS_24_1 then RNLS_24_4 and RNLS_24_14

RNLS_R24_17*: if RNLS_24_2 then RNLS_24_5 and RNLS_24_14

RNLS_R24_18*: if RNLS_24_3 then RNLS_24_6 and RNLS_24_14

RNLS_R24_19*: if RNLS_24_1 then RNLS_24_7 and RNLS_24_15

RNLS_R24_20*: if RNLS_24_2 then RNLS_24_8 and RNLS_24_15

RNLS_R24_21*: if RNLS_24_3 then RNLS_24_9 and RNLS_24_15

RNLS_R24_22: patient/person requests RNLS_24_10

RNLS_R24_23: org. connecting to the EHRi responds to RNLS_24_22

RNLS_R24_24: org. hosting components of the EHRi responds to RNLS_24_22

RNLS_R24_25: org. connecting to the EHRi provides understandable PHI

RNLS_R24_26: org. hosting components of the EHRi provides understandable PHI

RNLS_R24_27*: if RNLS_24_22 then RNLS_24_23 and RNLS_24_25

RNLS_R24_28*: if RNLS_24_22 then RNLS_24_24 and RNLS_24_26

RNLS_R24_29: patient/person challenges PHI accuracy

RNLS_R24_30: patient/person challenges PHI completeness

RNLS_R24_31: patient/person requests RNLS_24_29

RNLS_R24_32: patient/person requests RNLS_24_30

RNLS_R24_33: org. connecting to the EHRi allows RNLS_24_29

RNLS_R24_34: org. hosting components of the EHRi allows RNLS_24_29

RNLS_R24_35: org. connecting to the EHRi allows RNLS_24_30

RNLS_R24_36: org. hosting components of the EHRi allows RNLS_24_30

RNLS_R24_37: org. connecting to the EHRi amends PHI

RNLS_R24_38: org. hosting components of the EHRi amends PHI

RNLS_R24_39*: if RNLS_24_31 then RNLS_24_33 and RNLS_24_37

RNLS_R24_40*: if RNLS_24_31 then RNLS_24_34 and RNLS_24_38

RNLS_R24_41*: if RNLS_24_32 then RNLS_24_35 and RNLS_24_37

RNLS_R24_42*: if RNLS_24_32 then RNLS_24_36 and RNLS_24_38

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R25_1: patient/person demonstrates PHI inaccuracy

RNLS_R25_2: patient/person demonstrates PHI incompleteness

RNLS_R25_3: org. connecting to the EHRi amends PHI

RNLS_R25_4: org. hosting components of the EHRi amends PHI

RNLS_R25_5*: if RNLS_R25_1 then RNLS_R25_3

RNLS_R25_6*: if RNLS_R25_1 then RNLS_R25_4

RNLS_R25_7*: if RNLS_R25_2 then RNLS_R25_3

RNLS_R25_8*: if RNLS_R25_2 then RNLS_R25_4

RNLS_R25_9: User accesses PHI

RNLS_R25_10: org. connecting to the EHRi notifies user of the amendment of PHI

RNLS_R25_11: org. hosting components of the EHRi notifies user of the amendment of PHI

RNLS_R25_12*: if RNLS_R25_5 and RNLS_R25_9 then RNLS_R25_10

RNLS_R25_13*: if RNLS_R25_6 and RNLS_R25_9 then RNLS_R25_11

RNLS_R25_14*: if RNLS_R25_7 and RNLS_R25_9 then RNLS_R25_10

RNLS_R25_15*: if RNLS_R25_8 and RNLS_R25_9 then RNLS_R25_11

RNLS_R25_16: org. connecting to the EHRi disagrees with RNLS_R25_1

RNLS_R25_17: org. hosting components of the EHRi disagrees with RNLS_R25_1

RNLS_R25_18: org. connecting to the EHRi disagrees with RNLS_R25_2

RNLS_R25_19: org. hosting components of the EHRi disagrees with RNLS_R25_2

RNLS_R25_20: org. connecting to the EHRi records unresolved challenge

RNLS_R25_21: org. hosting components of the EHRi records unresolved challenge

RNLS_R25_22*: if RNLS_R25_16 then RNLS_R25_20

RNLS_R25_23*: if RNLS_R25_17 then RNLS_R25_21

RNLS_R25_24*: if RNLS_R25_18 then RNLS_R25_20

RNLS_R25_25*: if RNLS_R25_19 then RNLS_R25_21

RNLS_R25_26: org. connecting to the EHRi transmit unresolved challenge to user

RNLS_R25_27: org. hosting components of the EHRi transmit unresolved challenge to user

RNLS_R25_28*: if RNLS_R25_16 and RNLS_R25_9 then RNLS_R25_26

RNLS_R25_29*: if RNLS_R25_17 and RNLS_R25_9 then RNLS_R25_27

RNLS_R25_30*: if RNLS_R25_18 and RNLS_R25_9 then RNLS_R25_26

RNLS_R25_31*: if RNLS_R25_19 and RNLS_R25_9 then RNLS_R25_27

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RNLS_R27_1: org. connecting to the EHRi receives complaints about policies and practices on handling PHI

RNLS_R27_2: org. hosting components of the EHRi receives complaints about policies and practices on handling PHI

RNLS_R27_3: org. connecting to the EHRi receives inquiries about policies and practices on handling PHI

RNLS_R27_4: org. hosting components of the EHRi receives inquiries about policies and practices on handling PHI

RNLS_R27_5: org. connecting to the EHRi responds to complaints about policies and practices on handling PHI

RNLS_R27_6: org. hosting components of the EHRi responds to complaints about policies and practices on handling PHI

RNLS_R27_7: org. connecting to the EHRi responds to inquiries about policies and practices on handling PHI

RNLS_R27_8: org. hosting components of the EHRi responds to inquiries about policies and practices on handling PHI

RNLS_R27_9*: org. connecting to the EHRi develops procedures to RNLS_R27_1

RNLS_R27_10*: org. hosting components of the EHRi develops procedures to RNLS_R27_2

RNLS_R27_11*: org. connecting to the EHRi develops procedures to RNLS_R27_3

RNLS_R27_12*: org. hosting components of the EHRi develops procedures to RNLS_R27_4

RNLS_R27_13*: org. connecting to the EHRi develops procedures to RNLS_R27_5

RNLS_R27_14*: org. hosting components of the EHRi develops procedures to RNLS_R27_6

RNLS_R27_15*: org. connecting to the EHRi develops procedures to RNLS_R27_7

RNLS_R27_16*: org. hosting components of the EHRi develops procedures to RNLS_R27_8

RNLS_R27_17*: org. connecting to the EHRi informs complainants of procedures to RNLS_R27_1

RNLS_R27_18*: org. hosting components of the EHRi informs complainants of procedures to RNLS_R27_2

RNLS_R27_19*: org. connecting to the EHRi informs inquirers of procedures to RNLS_R27_3

RNLS_R27_20*: org. hosting components of the EHRi informs inquirers of procedures to RNLS_R27_4

RNLS_R27_21*: org. connecting to the EHRi informs complainants of procedures to RNLS_R27_5

RNLS_R27_22*: org. hosting components of the EHRi informs complainants of procedures to RNLS_R27_6

RNLS_R27_23*: org. connecting to the EHRi informs inquirers of procedures to RNLS_R27_7

RNLS_R27_24*: org. hosting components of the EHRi informs inquirers of procedures to RNLS_R27_8

RNLS_R27_25*: org. connecting to the EHRi treats complaints as confidential

RNLS_R27_26*: org. hosting components of the EHRi treats complaints as confidential

# Appendix C

# CREE Models for Case Study

The following are CREE models of the privacy requirements of Appendix A. The nominal goals in the models correspond to the privacy requirements and each nominal goal (<<Nominal Goal>>) has a “maker” attribute, “Infoway”, not shown in order to reduce the size.

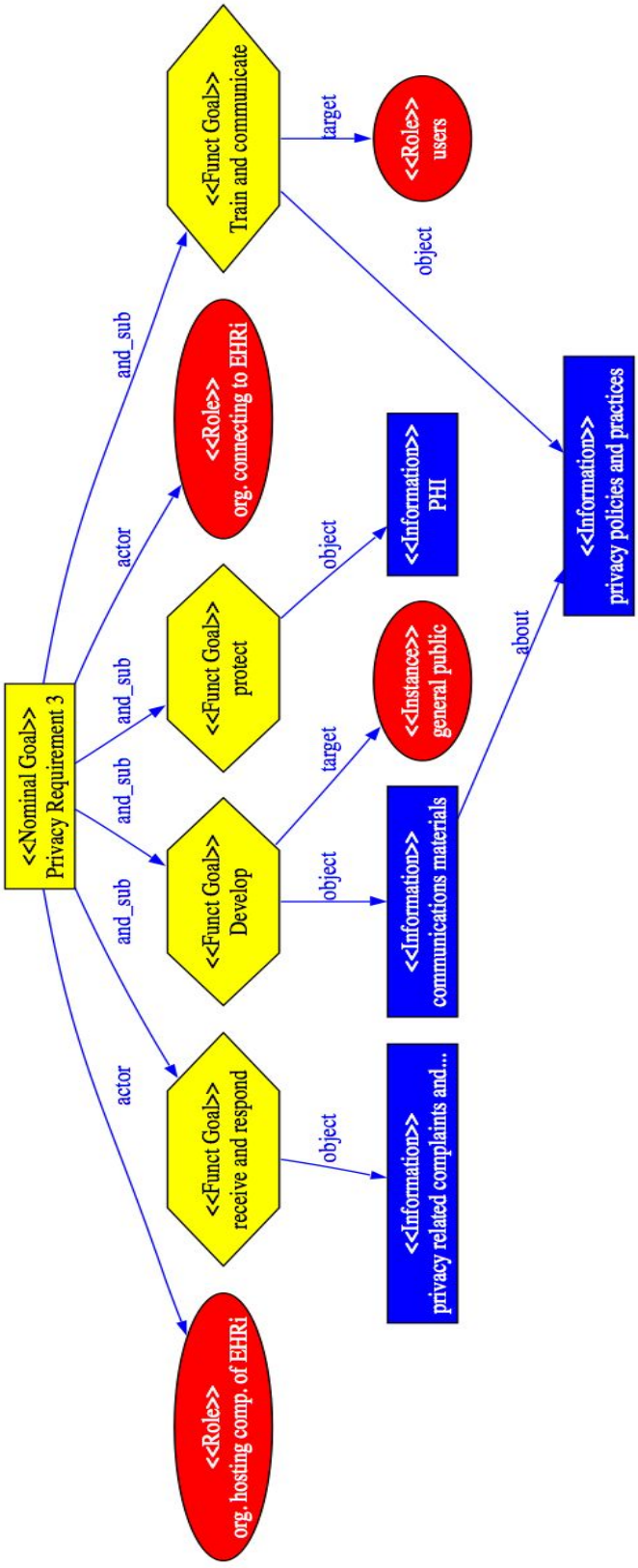


**Figure C.1.** *CREE model for Privacy Requirement 3 from Appendix A*

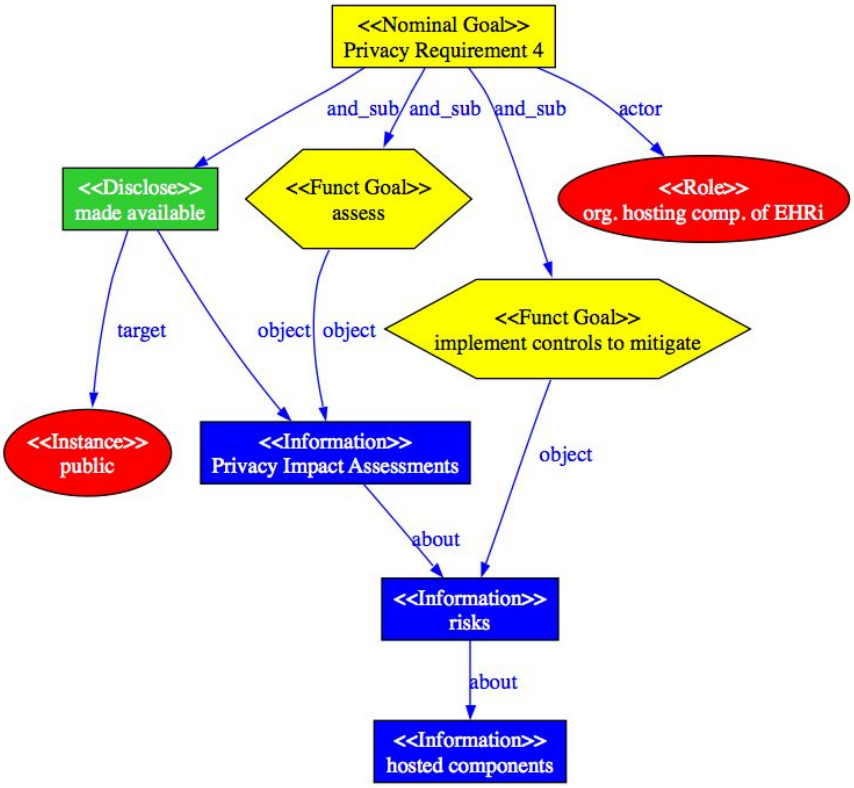


**Figure C.2.** *CREE model for Privacy Requirement 4 from Appendix A*

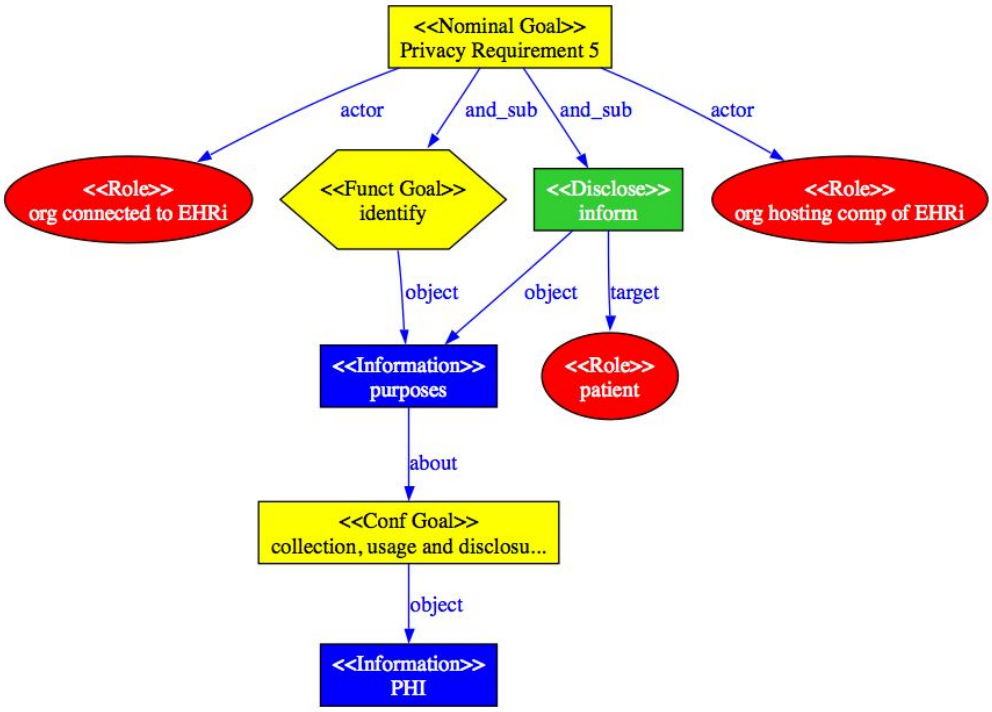


**Figure C.3.** *CREE model for Privacy Requirement 5 from Appendix A*

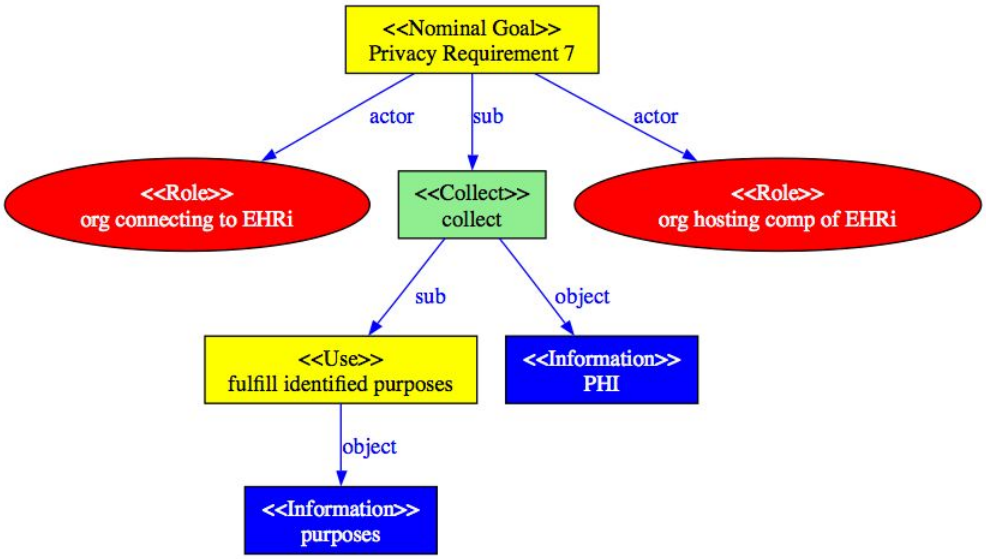


**Figure C.4.** *CREE model for Privacy Requirement 7 from Appendix A*

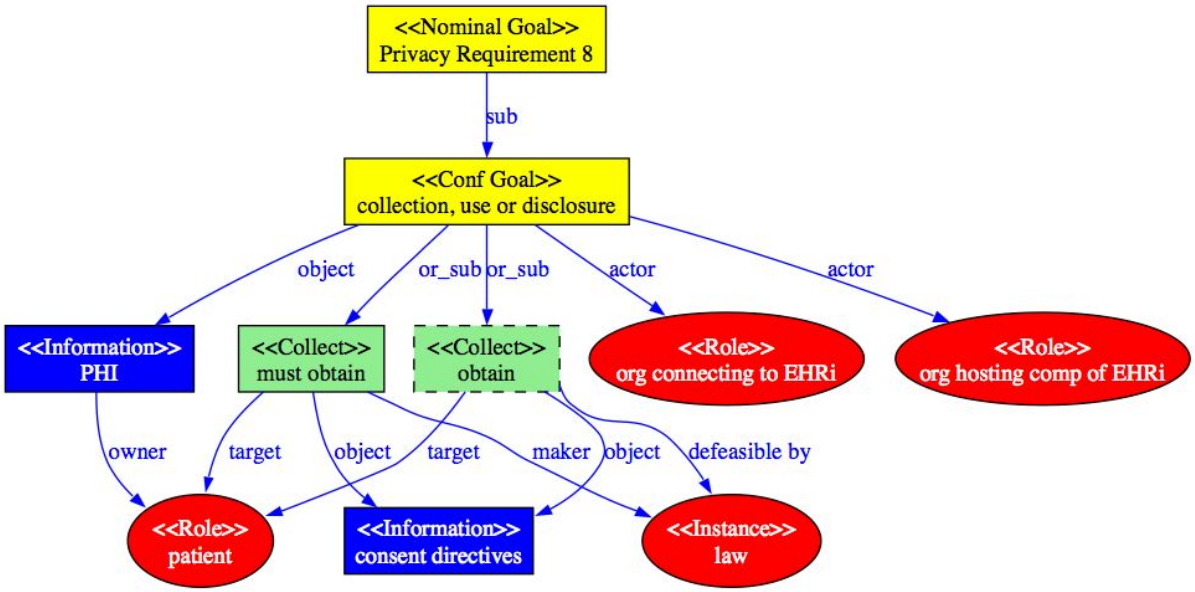


**Figure C.5.** *CREE model for Privacy Requirement 8 from Appendix A*

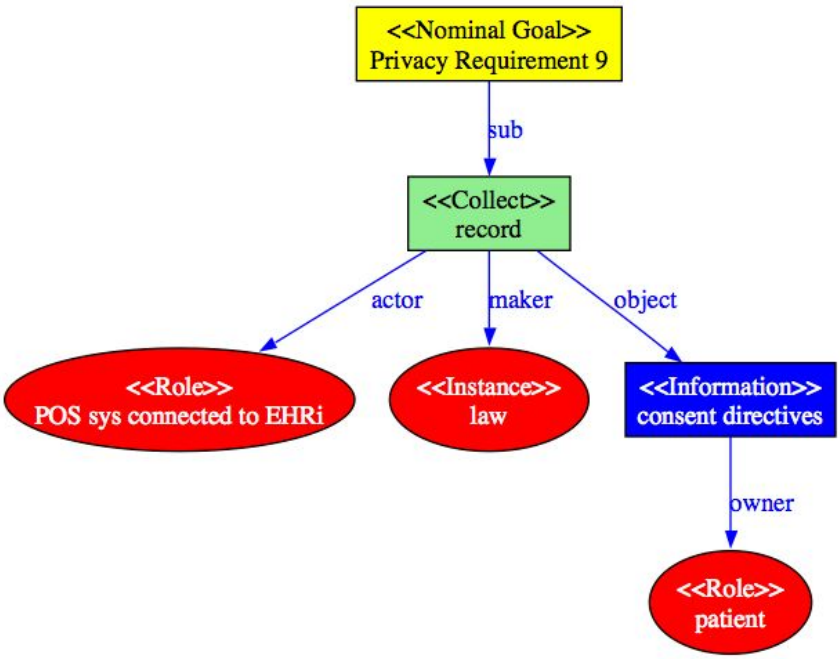


**Figure C.6.** *CREE model for Privacy Requirement 9 from Appendix A*

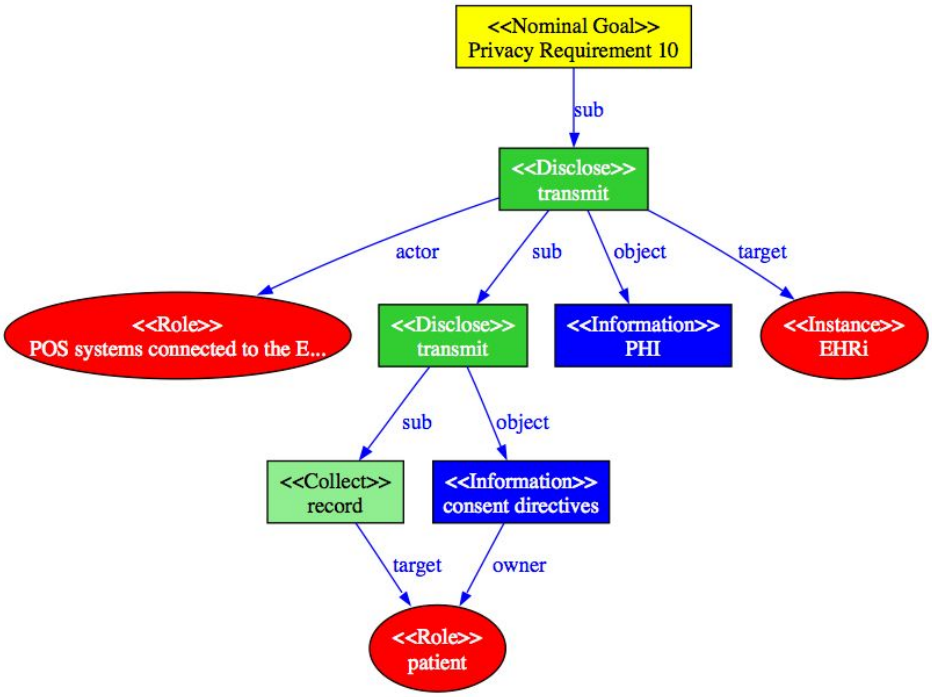


**Figure C.7.** *CREE model for Privacy Requirement 10 from Appendix A*

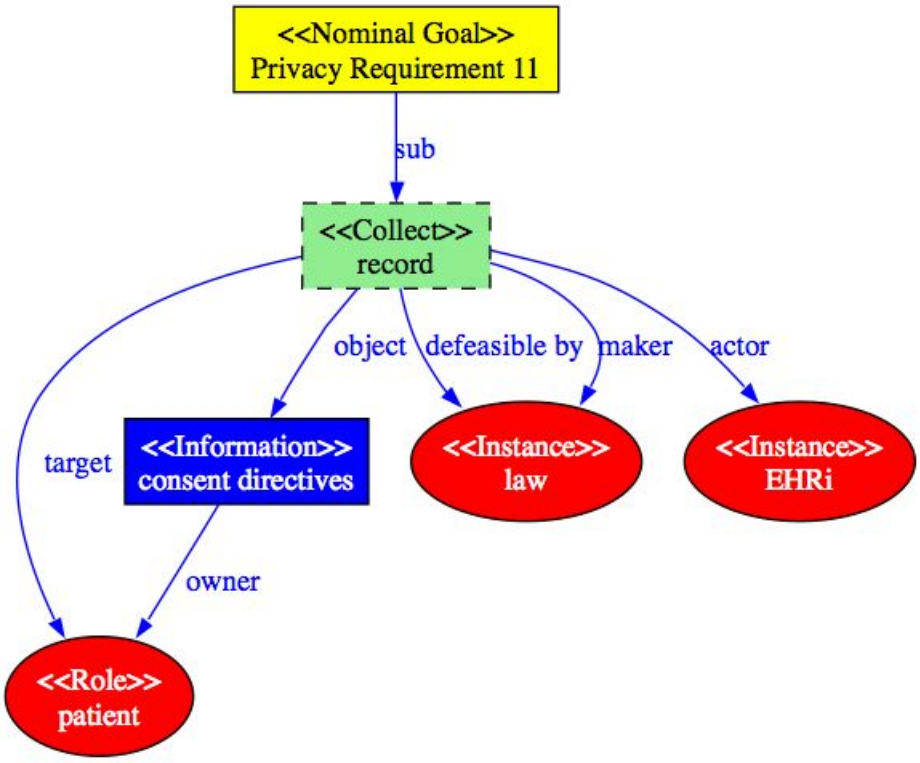


**Figure C.8.** *CREE model for Privacy Requirement 11 from Appendix A*

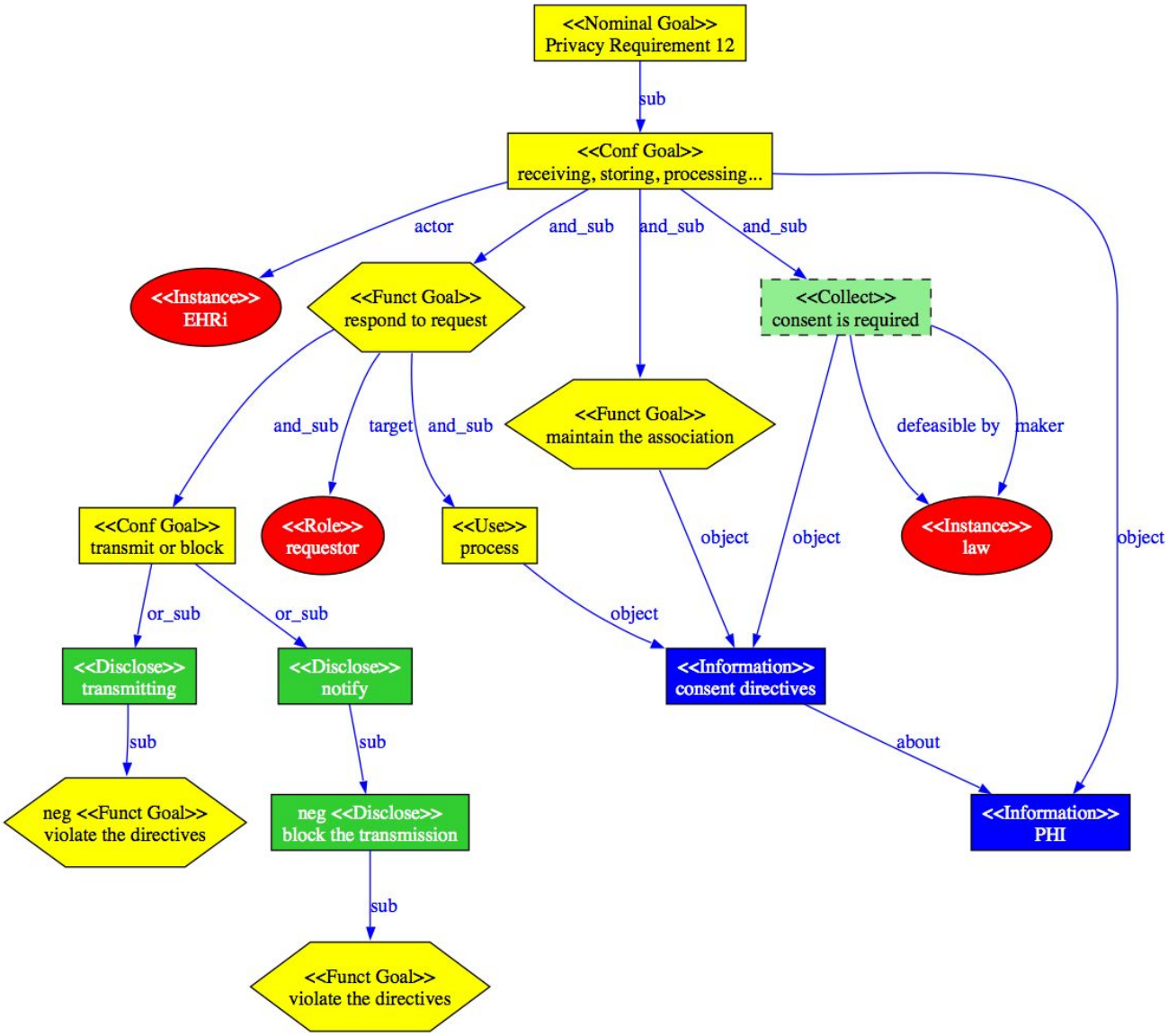


**Figure C.9.** *CREE model for Privacy Requirement 12 from Appendix A*

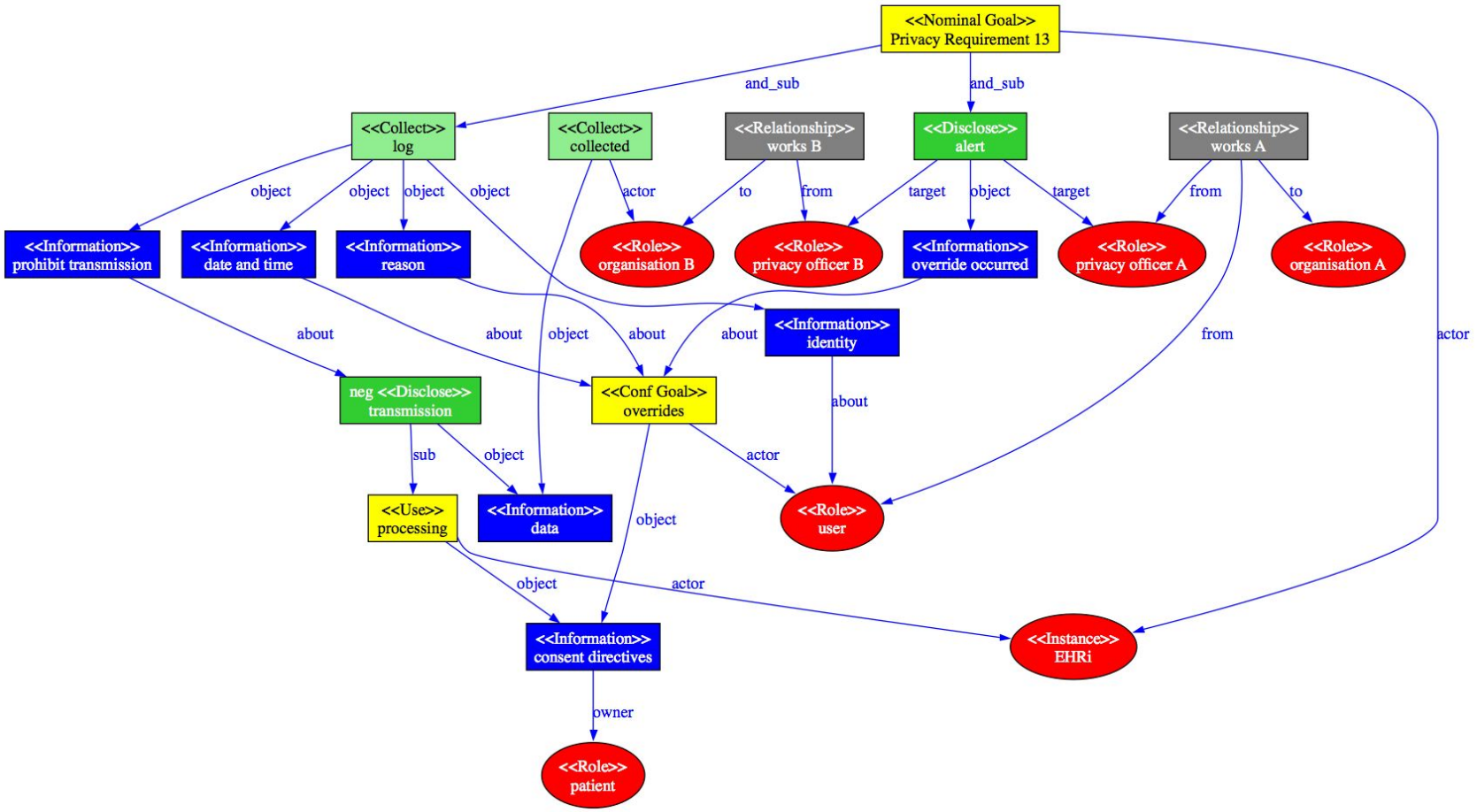


**Figure C.10.** *CREE model for Privacy Requirement 13 from Appendix A*

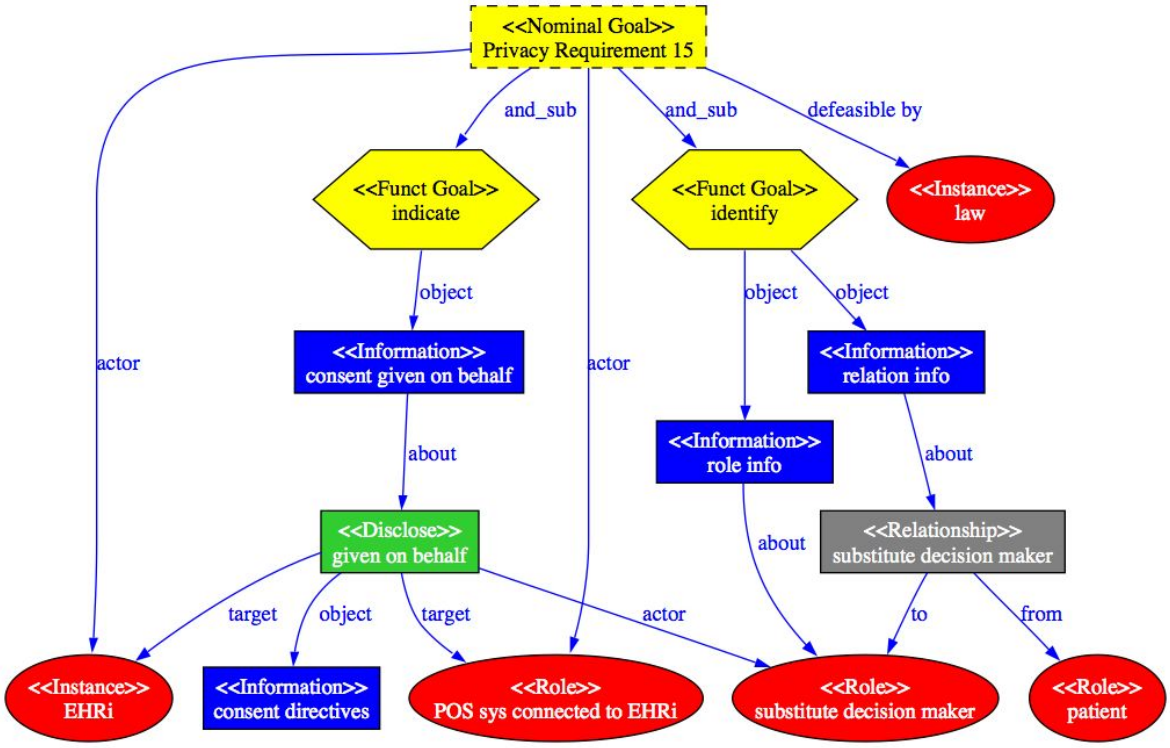


**Figure C.11.** *CREE model for Privacy Requirement 15 from Appendix A*

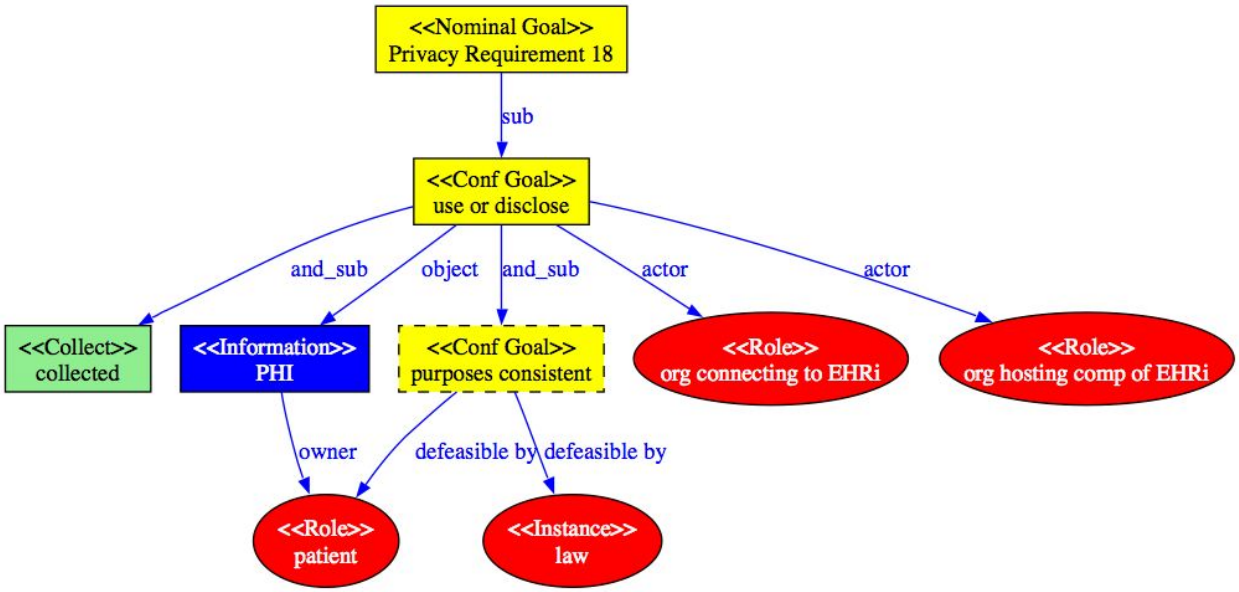


**Figure C.12.** *CREE model for Privacy Requirement 18 from Appendix A*

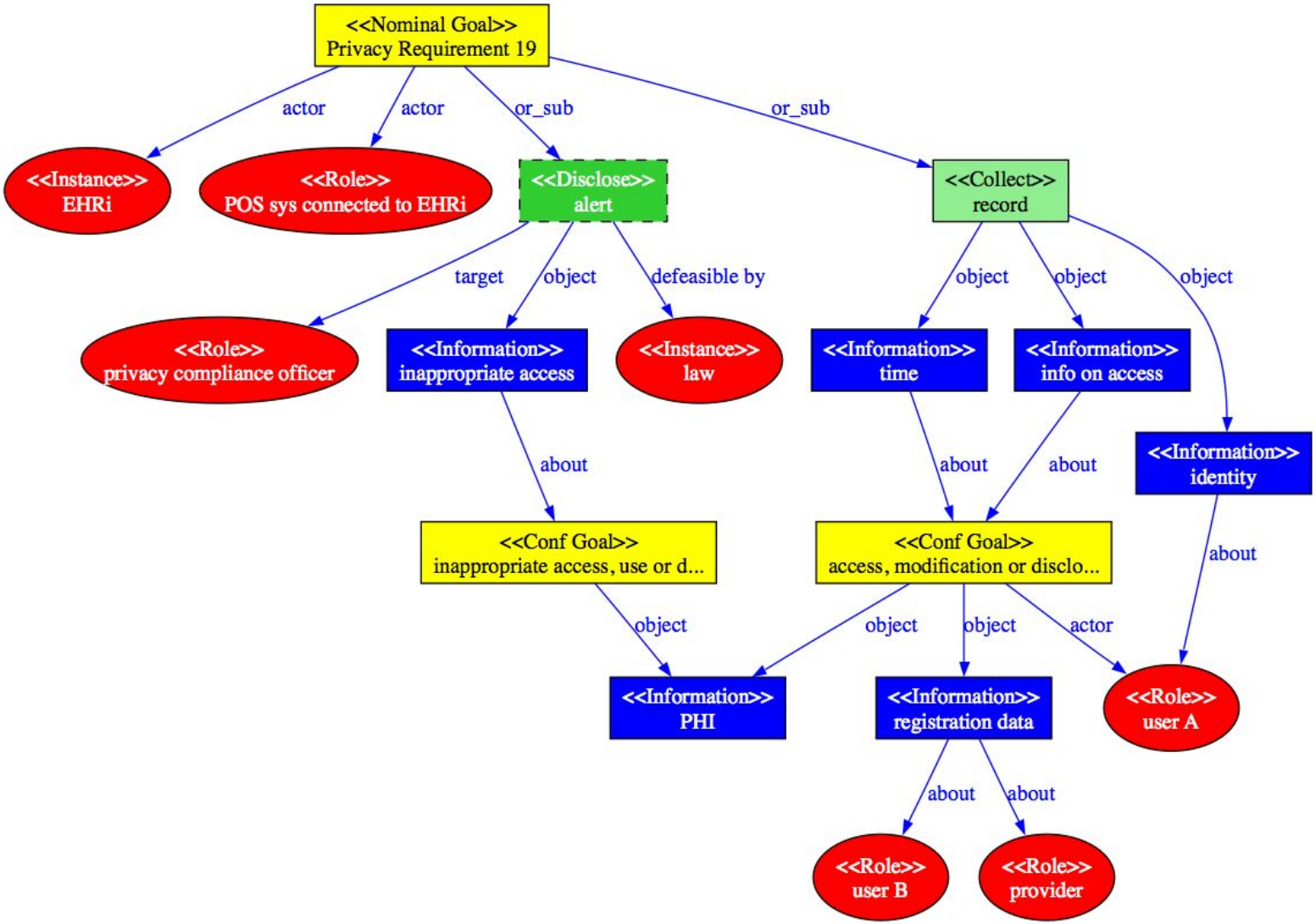


**Figure C.13.** *CREE model for Privacy Requirement 19 from Appendix A*

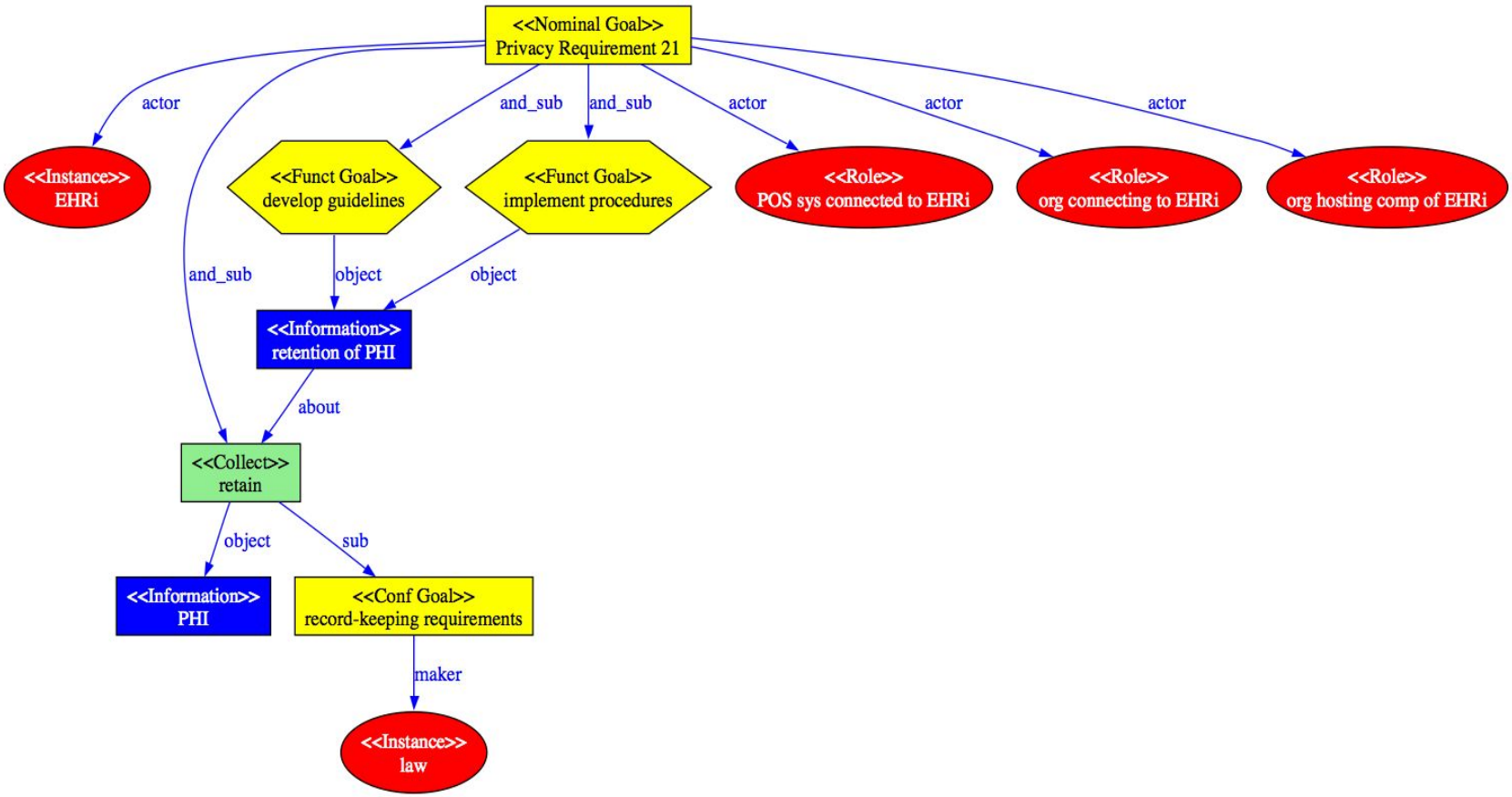


**Figure C.14.** *CREE model for Privacy Requirement 21 from Appendix A*

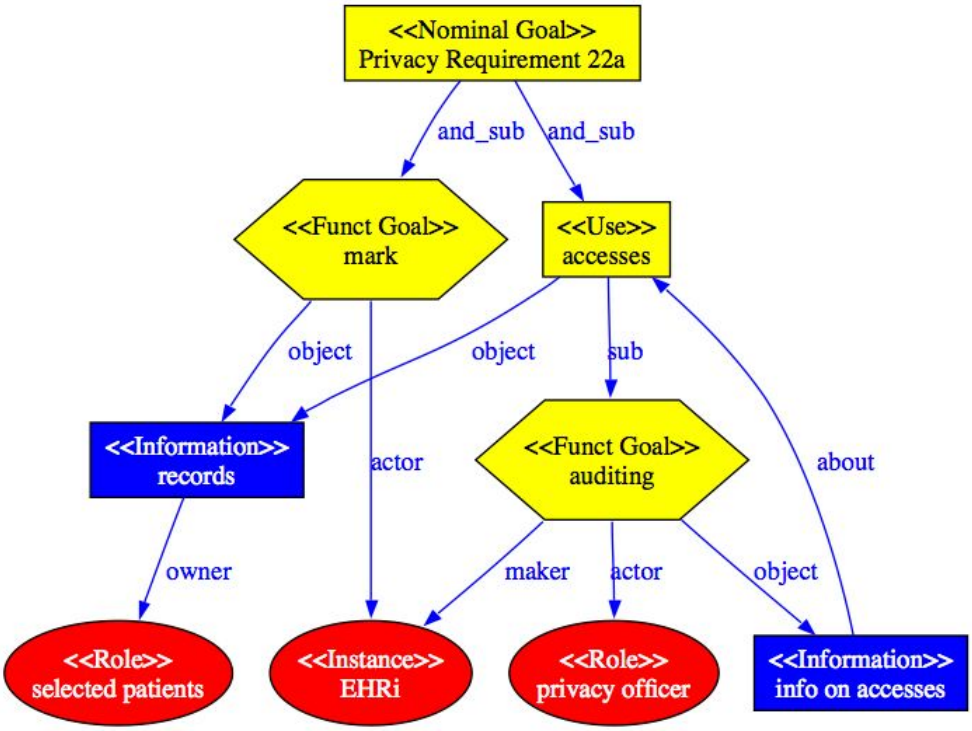


**Figure C.15.** *CREE model for Privacy Requirement 22a from Appendix A*

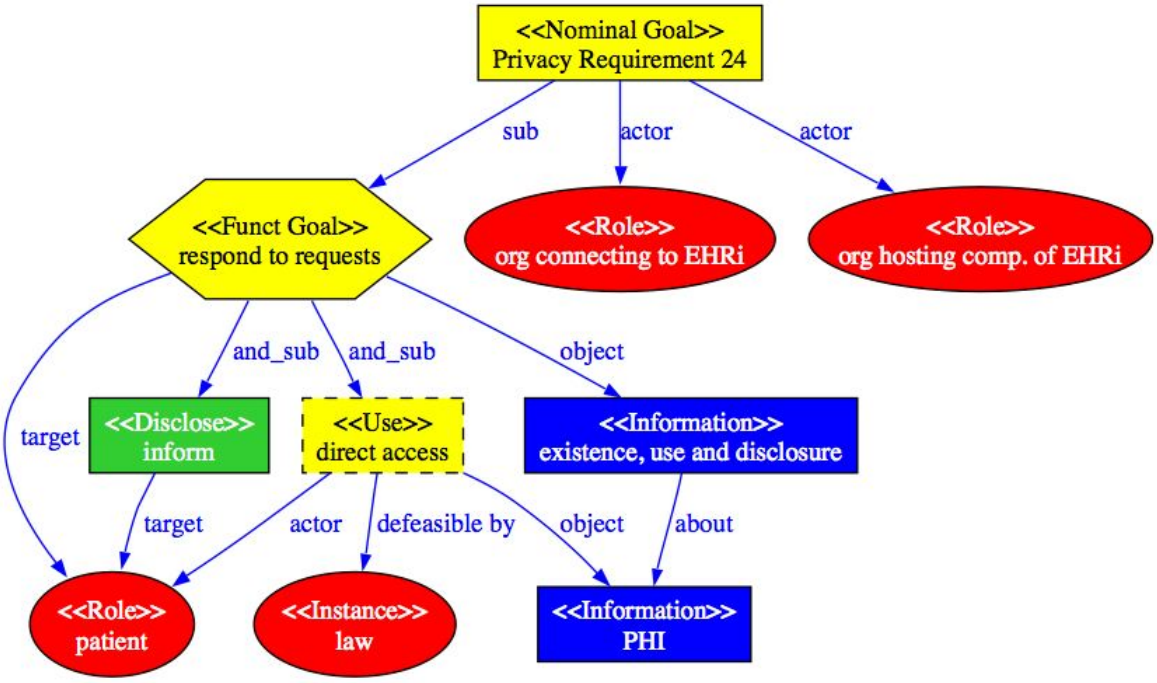


**Figure C.16.** *CREE model for Privacy Requirement 24 from Appendix A*

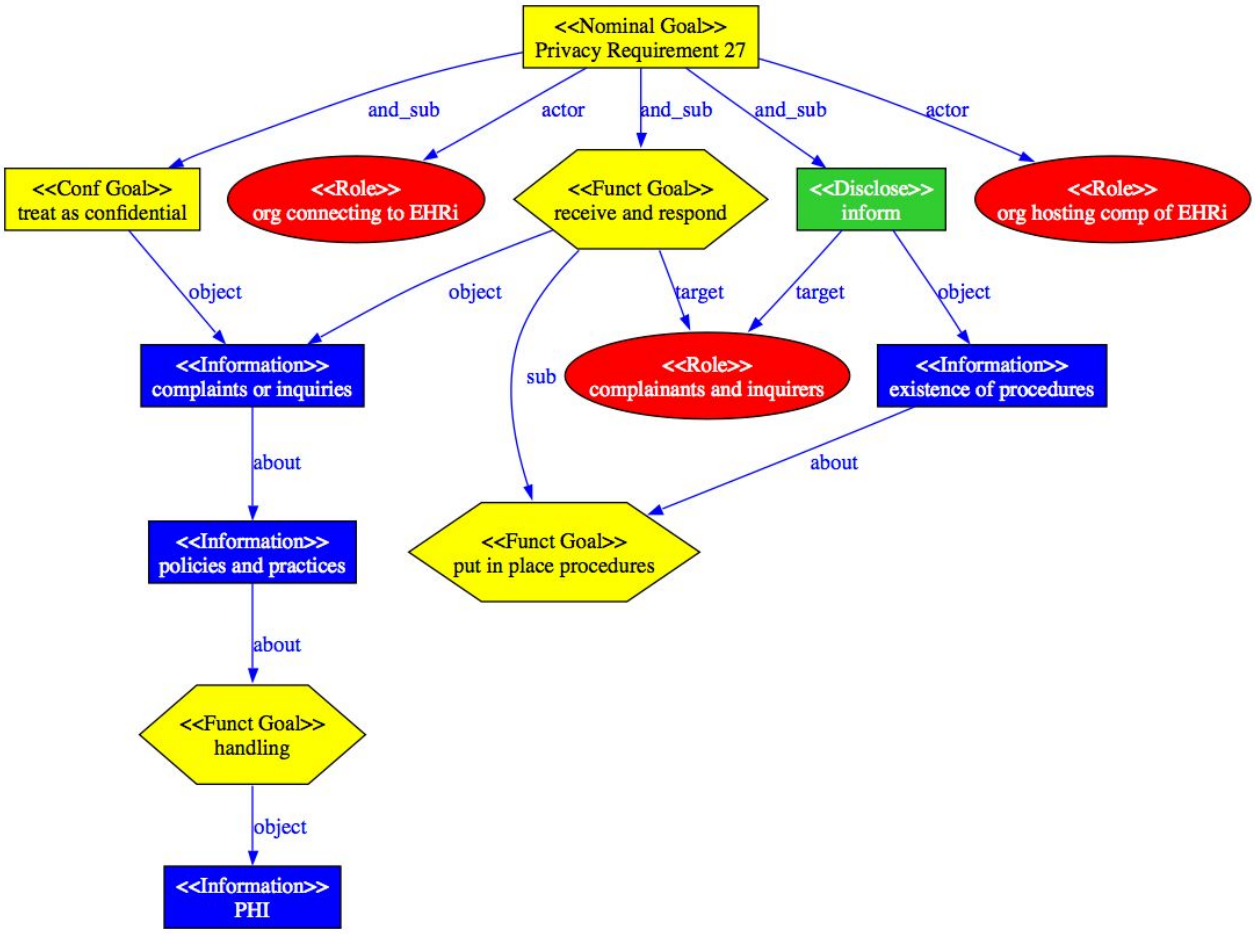


**Figure C.17.** *CREE model for Privacy Requirement 27 from Appendix A*

# Appendix D

# Logic Translation for CREE Interference and Conflict

The following are Prolog clauses used for formal analysis of CREE goal models to identify goal interference and conflicts.

Clause *opp_subsumedBy* is the translation of Definition 15 (i) to a Prolog representation. The following clauses are used in the definition of opp_subsumedBy.

- is_goalType(X,Y) - goal X is a subtype of goal Y
- stratified_stakeholder(A1,A2) - role A1 is a sub role of A2 or instance A1 is a member of role A2
- is_part_of(O1,O2) - information O1 is part of information O2

opp_subsumedBy(H1,H2) indicates that goal H1 is subsumed by goal H2, and H1 and H2 have opposite negative values. Negative goals are specified by using DR-Prolog's "~" operator.

```
opp_subsumedBy(H1,H2):- H1= X(L,M,A,T,O),H2=~(Y(L,M,A,T,O)),
                is_goalType(X,Y).
opp_subsumedBy(H1,H2):- H1=X(L1,M,A,T,O),H2=~(Y(L2,M,A,T,O)),
                is_goalType(X,Y).
opp_subsumedBy(H1,H2):- H1= X(L1,M1,A,T,O),H2=~(Y(L2,M2,A,T,O)),
                is_goalType(X,Y).
opp_subsumedBy(H1,H2):- H1=X(L1,M,A1,T,O),H2=~(Y(L2,M,A2,T,O)),
              is_goalType(X,Y),stratified_stakeholder(A1,A2).
opp_subsumedBy(H1,H2):- H1= X(L1,M,A,T1,O), H2=~(Y(L2,M,A,T2,O)),
              is_goalType(X,Y),stratified_stakeholder(T1,T2).
```

```
opp_subsumedBy(H1,H2):- H1=X(L1,M,A,T,O1),H2=~(Y(L2,M,A,T,O2)),
            is_goalType(X,Y),is_part_of(O1,O2).
opp_subsumedBy(H1,H2):- H1=X(L1,M,A1,T1,O),H2=~(Y(L2,M,A2,T2,O)),
            is_goalType(X,Y),stratified_stakeholder(A1,A2),
            stratified_stakeholder(T1,T2).
opp_subsumedBy(H1,H2):- H1=X(L1,M,A1,T,O1),H2=~(Y(L2,M,A2,T,O2)),
            is_goalType(X,Y),stratified_stakeholder(A1,A2),
            is_part_of(O1,O2).
opp_subsumedBy(H1,H2):- H1=X(L1,M,A,T1,O1),H2=~(Y(L2,M,A,T2,O2)),
            is_goalType(X,Y),stratified_stakeholder(T1,T2),
            is_part_of(O1,O2).
opp_subsumedBy(H1,H2):- H1=X(L1,M,A1,T1,O1),H2=~(Y(L2,M,A2,T2,O2)),
            is_goalType(X,Y),stratified_stakeholder(A1,A2),
            stratified_stakeholder(T1,T2),is_part_of(O1,O2).
opp_subsumedBy(H1,H2):- H1=~(X(L1,M,A1,T,O)),H2= Y(L2,M,A2,T,O),
            is_goalType(X,Y),stratified_stakeholder(A1,A2).
opp_subsumedBy(H1,H2):- H1=~(X(L1,M,A,T1,O)),H2= Y(L2,M,A,T2,O),
            is_goalType(X,Y),stratified_stakeholder(T1,T2).
opp_subsumedBy(H1,H2):- H1= ~(X(L1,M,A,T,O1)),H2= Y(L2,M,A,T,O2),
            is_goalType(X,Y),is_part_of(O1,O2).
opp_subsumedBy(H1,H2):- H1=~(X(L1,M,A1,T1,O)),H2= Y(L2,M,A2,T2,O),
            is_goalType(X,Y),stratified_stakeholder(A1,A2),
            stratified_stakeholder(T1,T2).
opp_subsumedBy(H1,H2):- H1=~(X(L1,M,A1,T,O1)),H2= Y(L2,M,A2,T,O2),
            is_goalType(X,Y),stratified_stakeholder(A1,A2),
            is_part_of(O1,O2).
opp_subsumedBy(H1,H2):- H1=~(X(L1,M,A,T1,O1)),H2= Y(L2,M,A,T2,O2),
            is_goalType(X,Y),stratified_stakeholder(T1,T2),
            is_part_of(O1,O2).
opp_subsumedBy(H1,H2):- H1=~(X(L1,M,A1,T1,O1)),H2=Y(L2,M,A2,T2,O2),
```

```
        is_goalType(X,Y),stratified_stakeholder(A1,A2),
        stratified_stakeholder(T1,T2),is_part_of(O1,O2).
```

Goal interference is defined in the Prolog translation with clause *interfere*. Clauses is_confGoalType and subgoal are used to define interfere.

- is_confGoalType(H) - the goal type of H, is a subtype of "Conf Goal"
- subgoal(H,G) - goal H is a subgoal of goal G

```
interfere(G1,G2,H1,H2):- opp_subsumedBy(H1,H2),is_confGoalType(H1),
        is_confGoalType(H2),subgoal(H1,G1),subgoal(H2,G2).
```

The subgoal clauses in the interfere clause represent the translation of Definition 15 (ii), and the is_confGoalType clauses are the translation of 15 (iii).

Goal conflict is defined in the Prolog translation with clause *conflict*. Clauses impliedFrom, hasMaker, and is_defeasible_by are used in the conflict clause.

- impliedFrom(G1,G2) - G1 is implied from G2 if it is in a conjunctive path of G2
- hasMaker(G,A) - A is the goal attribute *maker*, for goal G
- is_defeasible_by(G,A) - goal G has the *defeasible By* association with A

```
conflict(G1,G2,H1,H2):- interfere(G1,G2,H1,H2),impliedFrom(H1,G1),
        impliedFrom(H2,G2),impliedFrom(G1,T),impliedFrom(G2,T),
        hasMaker(G1,M1), sk_not(is_defeasible_by(G1,M2)),
        hasMaker(G2,M2), sk_not(is_defeasible_by(G2,M1)).
```

The impliedFrom clauses in the conflict clause represent the translation of Definition 16 (ii) and (iii), and the hasMaker and sk_not(is_defeasible_by) clauses are the translation of Definition 16 (iv).

# Appendix E

# User Study - Recruitment Email

Hello,

My name is Adeniyi Onabajo and I am a graduate student in the Department of Computer Science. I would like to invite you to participate in a study I am conducting to evaluate a tool developed in my research, for extracting requirements models from natural language sources such as regulations.

Your participation and feedback will be highly valued and would help me in evaluating and improving the approach. Coupons for Nibble and Bytes Cafe at ELW will be provided to each participant.

If you are interested in participating or need additional information on the study and research please contact me (onabajo@uvic.ca) or Dr. Jens Weber (jens@uvic.ca).

Thank you,
Adeniyi Onabajo

# Appendix F

# User Study - Consent Form

## User Study: Semantic Annotations of Natural Language Confidentiality Requirements

### Consent Form

You are invited to participate in a study entitled Semantic Annotations of Natural Language Confidentiality Requirements. The study is being conducted by Adeniyi Onabajo.

Adeniyi Onabajo is a graduate student in the department of Computer Science at the University of Victoria and if you have further questions you may contact him at onabajo@uvic.ca. The study is being conducted under the supervision of Dr. Jens Weber (jens@uvic.ca) and Dr. Margaret-Anne Storey (mstorey@uvic.ca).

Privacy and confidentiality requirements are usually specified in documents in natural language formats, e.g., regulations. The requirements can be represented as formal models, which are used for automated analysis. Linguistic transformations are used to extract restricted simple sentences, which can be translated into models. In this research we use semantic annotations as an alternative method of deriving models from natural language requirements. Semantic annotation involves the use of concepts in a particular domain (in this case confidentiality requirements concepts), and their relationships as annotations of regulation documents. The derived annotated structure is used for analysis. We are developing a tool to support the method.

The objectives of this study are: (a) compare the semantic annotation approach to a linguistic transformation method (b) get insights into ways of improving the semantic annotation method and the tool. Our study contributes to the research area by examining the benefits and drawbacks of using semantic annotation approach. We plan to improve the method and tool based on the results of the study. The potential benefits of your participation in this research are in contributing to the state of knowledge. Extracting requirements from natural language regulations is a challenging task

required for development of software systems that comply with the regulations. This research is aimed at developing a method that assists in analyzing the requirements by using annotations of regulations. We believe our study will contribute to knowledge by providing insights for improving the semantic annotation process.

You are being asked to participate in this study because you have experience in computer science or software engineering.

Participation involves: (i) extracting simple sentences from section of regulation document, (ii) introducing the semantic concepts, the tool and performing an annotation task, (iii) answering a set of interview questions to share your opinion and experience. The sessions will be audio and video recorded, and require a maximum of 90 minutes. The study will be held at ECS 448/555.

Participation in this study may cause some inconvenience to you since you must give up the required time to participate. There are no known or anticipated risks to you by participating in this study.

To compensate you for any inconvenience related to your participation, a coupon for Nibbles and Bytes Caf (at ELW) will be provided. It is important for you to know that it is unethical to provide undue compensation or inducements to research participants and if you agree to be a participant in this study, this form of compensation to you must not be coercive. If you would not otherwise choose to participate if the compensation was not offered, then you should decline.

Participation in this study is completely voluntary. If you do decide to participate, you may withdraw at any time without any consequences or any explanation. If you do withdraw from the study your data will be destroyed and will not be used in any research related to this study. Furthermore, if you withdraw, the coupon will still be available to you.

The researcher may have a relationship to potential participants, as the researcher is a student in the same university as potential participants. To help prevent this relationship from influencing your decision to participate, the following step(s) to prevent coercion have been taken: invitation was sent to general e-mail lists for recruitment of participants and no direct contact with potential participants was conducted during recruitment.

It is anticipated that the results of this study will be shared with others through presentations at scholarly meetings, academic dissertations, and published articles.

In terms of protecting your anonymity, you will not be directly identified in any dissemination

of the research. Data collected and responses will only be presented using a Participant ID.

All information disclosed to the researchers in this study is confidential. Electronic data, such as the video and audio recordings, will be securely stored on a computer in a locked research lab at the University of Victoria. All hard-copy documents containing participant information will be stored in a locked filing cabinet, which is only accessible to the principal investigator. Study data will be stored for no longer than three years. All electronic data will be deleted and paper-copies shredded at the end of this period.

For questions or concerns, you may contact the researcher and researchers supervisor through the contact information provided above.

In addition, you may verify the ethical approval of this study, or raise any concerns you might have, by contacting the Human Research Ethics Office, University of Victoria (250-472-4545 or ethics@uvic.ca).

Your signature below indicates that you understand the above conditions of participation in this study and that you have had the opportunity to have your questions answered by the researchers.

**Visually Recorded Image/Data**: Participant to provide initials:

Videos may be taken of me for: Analysis _____ Dissemination* _____

*Even if no names are used, you may be recognizable if visual images are shown as part of the results.

| _____ | _____ | _____ |
|---|---|---|
| Name of Participant | Signature | Date |

**A copy of this consent will be left with you, and a copy will be taken by the researcher.**

# Appendix G

# User Study - Procedure

1) Obtain Participant's consent
    a) Participant reads and signs the consent form.
2) Overview of the study
    a) Give a brief description of the objective of the research.
    b) Explain the extraction of structured elements from natural language sources.
3) Tutorial on sentence extraction
    a) Explain the use of patterns for extracting labelled simple statements.
    b) Use illustrative examples
        - Use a simple example with labelled sentences.
        - Use an example with nested/connected labelled sentences.
4) Sentence extraction task
    a) Inform participant that the study task will begin now: the requirement for the task is in the same document used for the tutorial.
    b) Inform participant to think aloud while performing the task.
    c) Inform the participant that he/she can ask questions while performing the task.
5) Tutorial on semantic annotation
    a) Explain the semantic concepts used for annotation.
    b) Show how concepts in the document are annotated in CREE-tool.
    c) Show how identified concepts are connected to create the goal structure in CREE-tool.
    d) Use illustrative examples
        - Annotate the same examples used in the first tutorial. The source document is

loaded into CREE-tool.

6) Semantic annotation task

   a) Inform participant that the study task will begin now: the requirement for the task is in the same document used for the semantic annotation examples.

   b) Inform participant to think aloud while performing the task.

   c) Inform the participant that he/she can ask questions while performing the task.

7) Study questionnaire

   a) Participant fills the study questionnaire.

# Appendix H

# User Study - Questionnaire

## User Study: Semantic Annotations of Natural Language Confidentiality Requirements

### Questionnaire

Participant ____

1) Compare the two approaches used in the study using a scale of 1- 5, where 1 is lowest and 5 is the highest for the following criteria

| | Criteria | Linguistic Transformation | Semantic Annotation |
|---|---|---|---|
| a | Ease of navigation from extracted sentence/model to document fragment | 1 2 3 4 5 | 1 2 3 4 5 |
| b | Traceability (linking from extracted sentence/model to document fragment) | 1 2 3 4 5 | 1 2 3 4 5 |
| c | Traceability on a large scale (say, 300 page document) | 1 2 3 4 5 | 1 2 3 4 5 |
| d | Extracting the structured elements (understanding and ease) | 1 2 3 4 5 | 1 2 3 4 5 |

2) What did you find difficult with the linguistic transformation method?

3) What did you find difficult with the annotation task?

4) Are the semantic concepts adequate in annotating the document fragment? If not, what changes or additional concepts would you suggest?

5) What additional functionalities should the semantic annotation tool support?