Analyzing Security Requirements As Relationships among Strategic Actors

Lin Liu[1], Eric Yu[2], John Mylopoulos[1]

[1] Computer Science Department, University of Toronto, Toronto, Canada M5S 1A4
{liu, jm}@cs.toronto.edu
[2] Faculty of Information Studies, University of Toronto, Toronto, Canada M5S 3G6
yu@fis.utoronto.ca

Abstract. Security issues for software systems ultimately concern relationships among social actors – stakeholders, users, potential attackers, etc. -- and software acting on their behalf. In assessing vulnerabilities and mitigation measures, actors make strategic decisions to achieve desired levels of security while trading off competing requirements such as costs, performance, usability and so on. This paper explores the explicit modeling of relationships among strategic actors in order to elicit, identify and analyze security requirements. In particular, actor dependency analysis helps in the identification of attackers and their potential threats, while actor goal analysis helps to elicit the dynamic decision making process of system players for security issues. Patterns of relationships at various levels of abstraction (e.g. intentional dependencies among abstract roles) can be studied separately. These patterns can be selectively applied and combined for analyzing specific system configurations. The approach is particularly suitable for new Internet applications where layers of software entities and human roles interact to create complex security challenges. Examples from Peer-to-Peer computing are used to illustrate the proposed framework.

1. Introduction

Security is an increasingly prominent and critical concern for all software systems. Compared to traditional software, new Internet applications face new security issues which are too complex to deal with without effective models and decision support tools. When relevant actors for a given software system belong to a closed community, security is comparatively easy to enforce and control. For flexible sharing among members of dynamic and open groups such as those on the Internet, different parts of networks and systems may be operated by parties with conflicting interests or even malicious intent. Threats of varying technical difficulty targeting different levels of a network infrastructure exist everywhere. Furthermore, many technologies, as well as system models, are new and their viability remains unproven [Shneider99b].

This paper explores how strategic actor models can be used to elicit, identify and analyze security requirements. Security issues are ultimately about relationships among social actors – stakeholders, users, potential attackers, etc., and software acting on their behalf. In assessing vulnerabilities and mitigation measures, actors make strategic decisions to achieve desired levels of security while trading off competing requirements such as costs, performance, usability and the like.

Consider the case of peer-to-peer (P2P) computing. Peers acting as endpoints on the Internet, often without knowing each other, exchange information and form communities, thereby creating new vulnerabilities. P2P offers an interesting illustration of networked applications, where layers of software entities and human roles interact to present complex security challenges. By analyzing the dependency relationships between peers and other supporting players of the system, questions such as the following can be answered: Who are the potential adversaries of the system? How do they threaten the security of the system? Why do they want to attack? How much trust should a peer place on another peer or a peer community she is acquainted with? How much can she trust the information circulated in this community? How much private information should be exposed to another peer? What counter-measures can be sought to the system designers and participants? Which one of them is more effective?

In section 2, a modeling framework is briefly introduced. Section 3 shows an initial model of security issues in file sharing P2P systems, while section 4 discusses the strategic modeling techniques using examples in the P2P domain. Section 5 discusses related work. Finally, section 6 summarizes the approach and concludes the paper.

2. Modeling Strategic Actor Relationships with the i* Framework

*i**[Yu97] is a framework that supports goal- and agent-oriented strategic modeling and analysis of requirements, with emphasis on non-functional requirements (NFRs) [Chung00]. *i** offers three basic types of concepts for modeling applications: *actors*, *intentional elements*, and *links*.

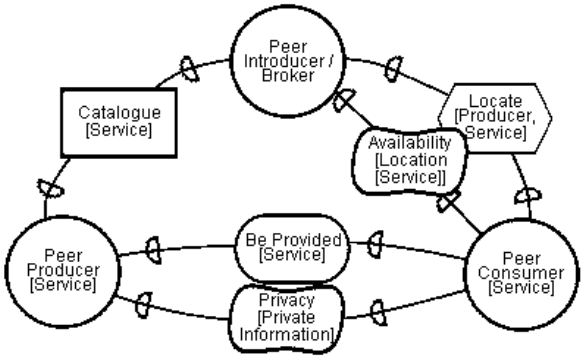

Figure 1: A simple P2P cooperation model.

We consider a simple P2P computing example for illustration. P2P computing [Oram01] consists of an open-ended network of distributed computational peers, where each peer can exchange data and services with a varying set of other peers (its *acquaintances*). Figure 1 shows an abstract P2P cooperation model. For each cooperation, there is a peer acting as producer of a service (or resource), while another peer acts as the consumer of the service. The Consumer depends on the Producer not only for Service to Be Provided (the functional requirement), but also for any number of non-functional requirements such as Privacy. The two peers establish an acquaintance through the introduction of a Broker. The three players of the system are represented as actors. They depend on each other for goals to be achieved, tasks to be performed or resources to be furnished. A dependency relationship allows an actor to take advantage of capabilities offered by another actor. At the same time, a dependency creates vulnerabilities since the "depender" may fail to achieve its goals if the dependency fails. In *i**, the term actor (represented with a circle) is an active entity that carries out actions to achieve its goals by exercising strategic know-how. An actor may be thought of as an encapsulation of intentionality, rationality and autonomy [Yu01c]. Graphically, an actor may optionally have a boundary containing intentional elements. To model complex relationships among actors, we further define the concepts of *agents* and *roles*, which are special cases of actors:

- An agent (circle with a line at the top) is an actor with concrete, physical manifestations, such as a human individual or a software/hardware system. An agent has dependencies that apply regardless of what role he/she/it happens to be playing.
- A role (circles with a line at the bottom) is an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor. Dependencies are associated with a role when these dependencies apply regardless of who plays the role.

*i** models are intentional in that they describe intended states of affairs. Unlike behavioral models, intentional models express answers to "why" questions: Why are particular behavioral or informational structures chosen? What alternatives have been considered? What criteria are used to deliberate among alternatives? What are the reasons for choosing one alternative over others?

An *i** model can either be composed of a global goal model, or a series of goal models distributed amongst several actors. If a goal model includes more than one actor, then the intentional dependency relationships between actors can also be represented and reasoned about.

The intentional elements in *i** are goals, tasks, softgoals, and resources.

- A goal (rounded rectangle) is a condition/state of affairs that a stakeholder wants to achieve.
- A task (hexagon) specifies a particular way of doing something, a particular course of action.
- A softgoal represents quality requirements for a system, such as, performance, security, accuracy, reusability, interoperability, time to market and cost. A softgoal, is shown as an irregular curvilinear shape ("cloud").
- A resource is an (physical or informational) entity, about which the main concern is whether it is available. Resources are represented as rectangles.

Intentional links include means-ends, decomposition, contribution, correlation and dependency links. *Means-ends* links are used to describe how goals can be achieved. Each task connected to a goal by a means-ends link is an alternative way to achieve the goal. *Decomposition* links define the sub-components of a task. A *Contribution* link describe the impact that one element has on another. A contribution can be negative or positive. The extent of the contribution can be partial (*Help* or *Hurt*) or sufficient (*Make* or *Break*) based on the concept of satisficing [Simon81]. *Correlation* links describe the side effects of the existence of one element to others. *Scenario path* links describe the temporal order of the execution of tasks in a scenario. *Dependency* links describe the inter-actor dependency relationships. There can be contributions to a link too. For example, a *Break* contribution to a dependency link means that the contribution is negative enough to render the dependency unviable. Following are the graphical representations for links.

By doing intentional level modeling, we can avoid the operational details and focus on the analysis of the high-level strategic relationships between actors. These strategic relationships are the fundamental driving force of the system, and they constitute the basis for stakeholders' decisions, which affect their strategic interests.

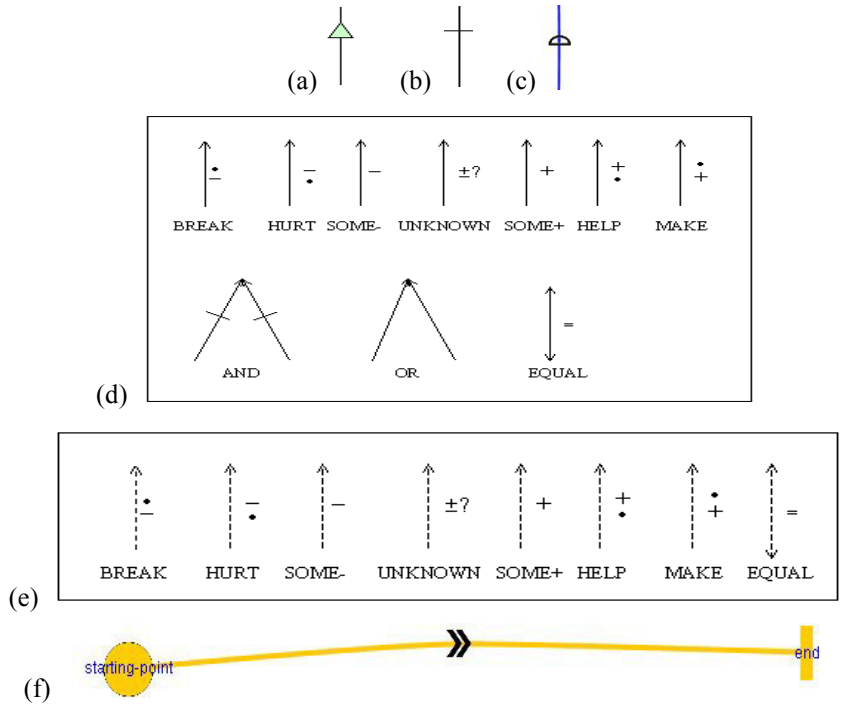

Figure 2: (a) Means-Ends; (b) Decomposition; (c) Dependency;

(d) Contribution; (e) Correlation; (f) Scenario path.

Note that security issues are represented and reasoned about without special notations and constructs. They are treated in the same way as other non-functional requirements, such as usability, maintainability, and time-to-market. This facilitates the simultaneous treatment of these interacting requirements during early stage analysis and design. Furthermore, security is not treated as a local bounded phenomenon. It is modeled and analyzed in terms of a network of interdependent, but ultimately autonomous actors. Weaknesses and faults can propagate across the dependency network with far-reaching effects. Analyzing strategic relationships helps understand the impact and extent of threats, and the effectiveness of mitigating measures.

3. Starting from An Initial Model – A First Glimpse of the Security Issues in File Sharing P2P

Figure 3 is an initial model of the Napster system showing some security issues. Major players of the system are represented as agents, including Napster.com – the central broker and controller, and Napster Peer sharing MP3 files. A supporting player ISP Server, also modeled as an agent, provides Internet Service to the peers. Each Napster Peer may play two roles– Peer Producer and Peer Consumer.

Agents with square shadows in the model are potential "attackers" of the system, e.g., an internal attacker Hacker Jack may submit a fake list of songs to Napster.com, or he may send an MP3 file with "back doors" to another peer. An external intruder Malicious Jim may do harm to the system by flooding emails to an ISP with a consequence of "denial of service" failure to the system. The model was arrived at in an ad hoc way and relies on experience with individual relationships and specific cases. A more systematic approach is needed.

Note that the model includes issues that are not usually addressed in technical treatments, e.g., legal attacks and defences. The *i** framework is not specific to security, thus allowing diverse issues to interact. For instance, In the model, another related player the Copyright Holder is also depicted, on whose behalf RIAA (Recording Industry Association of America) has filed the well-known lawsuit against Napster for the infringement of music copyright, which constitutes part of the environment of the Napster system.

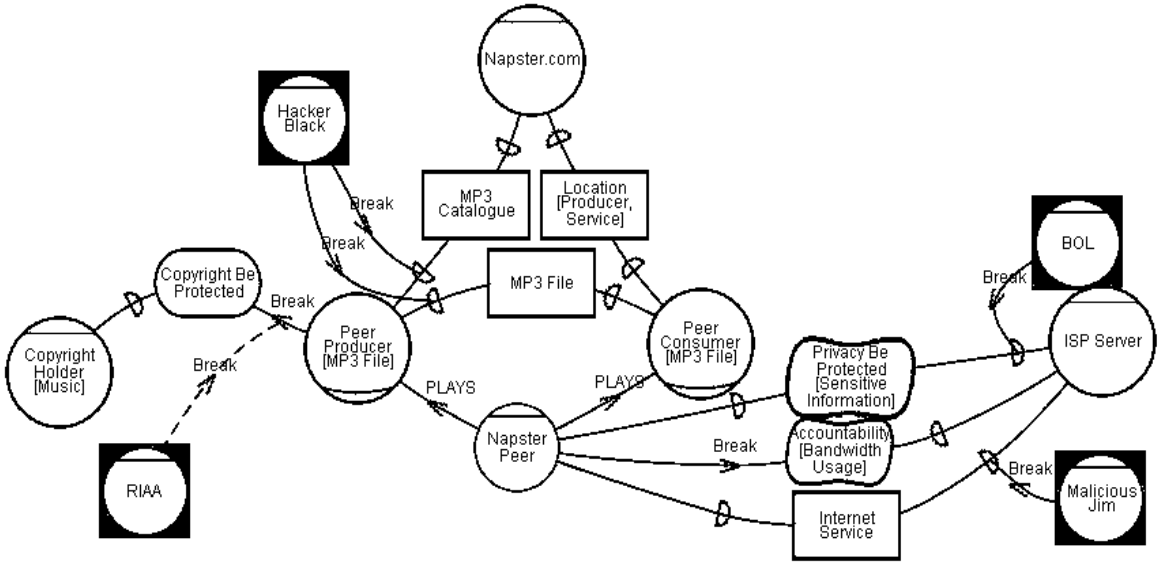

Figure 3: Dependency model of the Napster system.

4. Towards A Systematic Approach

In this section, security requirements in P2P systems are explored using the proposed strategic modeling approach composed of agent-oriented analysis, goal-oriented analysis and scenario-based analysis. The agent-oriented approach is based on the idea that abstract role patterns can be modeled and reused. For a particular system, actual system components and functions can be viewed as specializations and instantiations of generic patterns or the combination of several patterns. The goal-oriented analysis is based on the idea that relationships on various kinds of requirements can come from generic knowledge organized in catalogues. The scenario-based analysis can be considered as an elaboration of the other two kinds of analysis.

4.1 Agent-oriented analysis I: abstract role dependency analysis - the generic pattern

The first step of agent-oriented analysis is to identify the generic role dependency patterns in the domain. Two meta-level roles: User [Service] and Provider [Service] are shown in Figure 4. The cooperation between these two kinds of roles covers the most common relationships between collaborating actors. A User depends on the Provider for some Service. This is the fundamental functional requirement of the system that bridges the two sides of the collaboration. Non-functional requirements arise from both sides, as expectations on the other side.

To identify the dependency relationships between a pair of actors, one first needs to find out what service is in need, and add it as a goal, task or resource dependency according to the type of freedom the depended party ("dependee") has. Then, using a catalogue (to be presented in section 4.4) as a reference, we elaborate the model according to whether the two roles involved have trust, security, or privacy expectations on each other. If they do, they are represented as softgoal dependencies in the model.

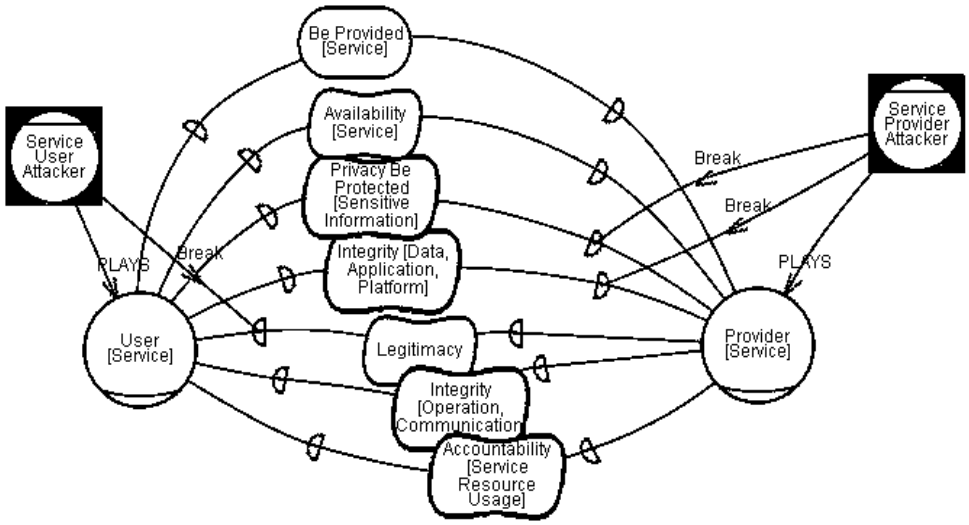

Figure 4: User-provider relationships.

The model in Figure 4 shows that, in general, a User depends on a Provider to provide some service (goal dependency Be Provided [Service]). Thus, it leads to the user's concerns on the Availability of the service requested, and Privacy Be Protected, especially Sensitive Information such as his personal data, service usage history, etc. Also he may want Integrity [Data, Application, Platform] guaranteed by the Provider. The Provider in turn depends on the User for Legitimacy, Integrity [Operation, Communication], and Accountability [Service Resource Usage].

After establishing the dependency relationships, we consider the roles that attackers could potentially play. Potential threats/attacks to the system are modeled as negative contributions of different strengths (*BREAK*, *HURT*, *SOME-*) to dependency links. The immediate victims of these attacks are the actors depending upon the attacked dependencies.

4.2 Agent-oriented analysis II: role-agent hierarchy in file-sharing P2P systems - Mapping the reality into the concept

The next step in agent-oriented analysis is to figure out what abstract roles each system player is playing. Figure 5 shows a role-agent hierarchy of Napster/Gnutella-type protocols. The top part of this diagram represents generic roles which apply to most P2P systems. Each role is an abstraction of certain functions or capabilities. The lower part of the diagram depicts agents that are specific to Napster/Gnutella-type systems. These agents may play (*PLAYS* link) one or many abstract roles, thereby inheriting the capabilities and social relationships of the roles. Any player in the real world is an instance of one of these agent classes.

The two areas in dotted lines (not part of the notation), highlight the main difference between Napster and Gnutella in terms of their roles. In Napster, peers play two roles: Producer and Consumer of MP3 files. They are connected through the introduction of Napster.com, the only central controller in the system. In contrast, Gnutella has a pure distributed architecture, its peers are playing all three roles: Producer, Consumer and Introducer. Thus, in the Napster system, the study of trust/security needs to be conducted among peers and from each peer to Napster.com, while in Gnutella only the peers are to be assessed. This basic pattern of relationships can be used to answer questions such as: Who are the major players in the business domain? What kinds of relationships exist among them? The role-agent hierarchy is used to combine abstract relationship patterns such as that shown in Figure 4 to derive the relationships between systems players in specific P2P systems systematically (later in Figure 7).

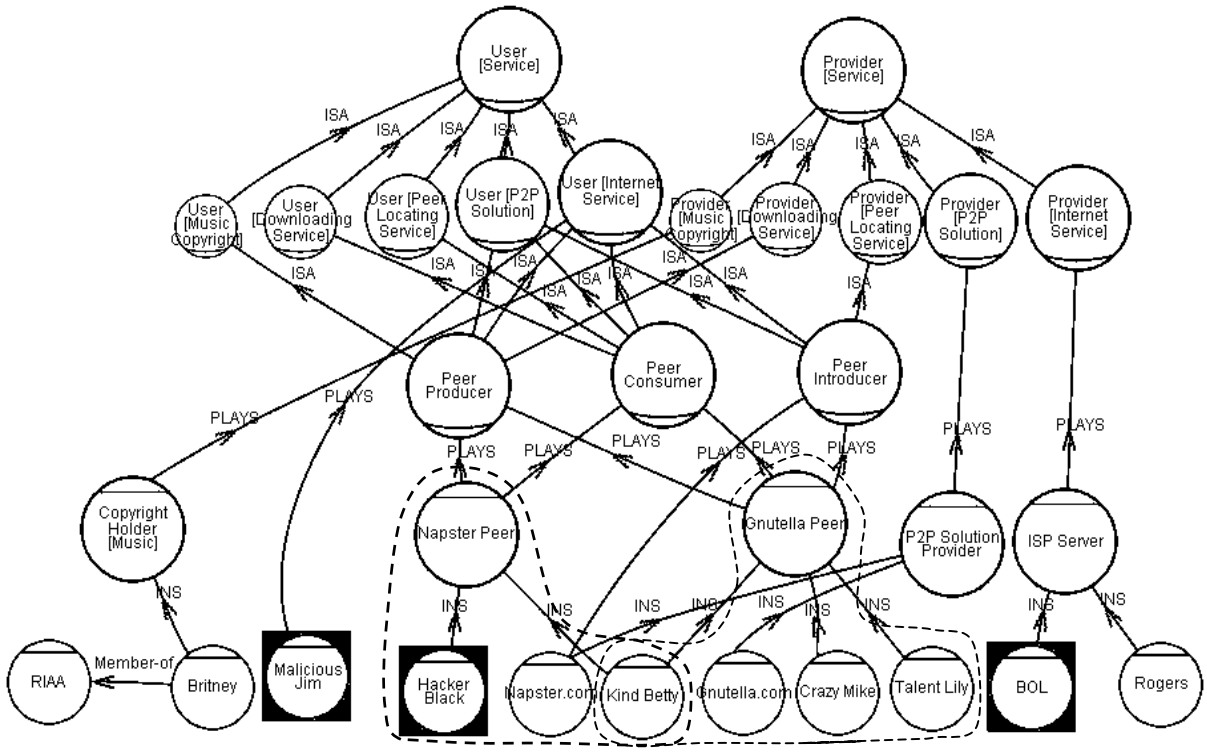

Figure 5: Role-agent hierarchy, for general and Napster/Gnutella-specific roles.

4.3 Agent-oriented analysis III: dependency derivation along actor associations

By combining the dependency relationships in Figure 4 and the role hierarchy in Figure 5, we can study pairwise dependencies between Producer, Consumer, and Introducer. All three pairs can be considered as User vs. Provider dependencies for a specific service. Therefore, they can inherit the dependencies between general User and Provider. Since some parameters are substituted with more concrete objects as the analysis moves to lower levels, some dependencies may no longer apply, i.e., inheritence of dependencies is selective. However, when there is insufficient domain knowledge, the intentional dependencies are inherited automatically along actor association links such as *ISA* and *PLAYS*, and will not be refined until the knowledge is available.

Figure 6 shows how the relationships between Peer Consumer vs. Peer Producer, and Peer Consumer vs. Peer Introducer are derived from higher-level roles. The dotted broad arrows indicate steps in the derivation. For a Provider [Downloading Service], the Legitimacy of the User is not at issue, since the service is generally open to everyone. Integrity of user's operation is not required either. However, the Provider does care if service provision consumes too much of her bandwidth (Accountability [Bandwidth]). The dependencies between User [Peer Locating Service] and Provider [Peer Locating Service] are produced in a similar fashion. Then, Peer Consumer receive both user roles' dependencies, while the Producer receive only those from Provider [Downloading Service], and the Introducer receive the ones from Provider [Peer Locating Service]. Further down at the agent level, dependencies are taken over in a similar fashion by the agents playing the corresponding roles. An interesting point at this level is the dropping of the Availability dependency between peers. This is so because if one peer cannot provide a service any more, there will be other candidates to take his place. However, Availability is still an expectation to the whole group of peers. This situation explicitly distinguishes Napster from Gnutella. The Availability [Peer Locating Service] is expected of Napster.com, but in Gnutella, this is kept as an expectation to the whole Gnutella peer group at the agent class level.

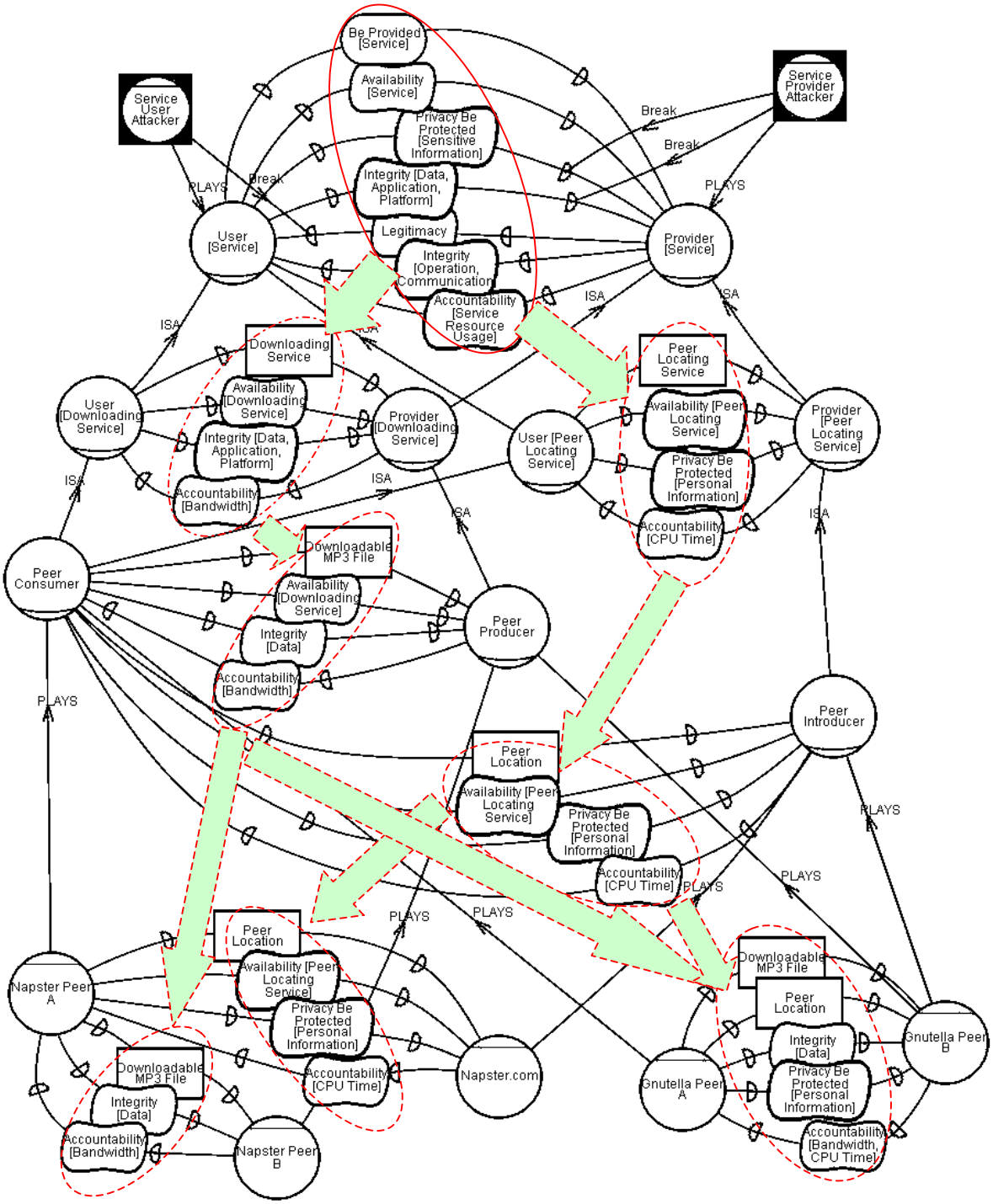

Figure 6: Dependency derivation.

4.4 A catalogue of security related non-functional requirements in P2P domain

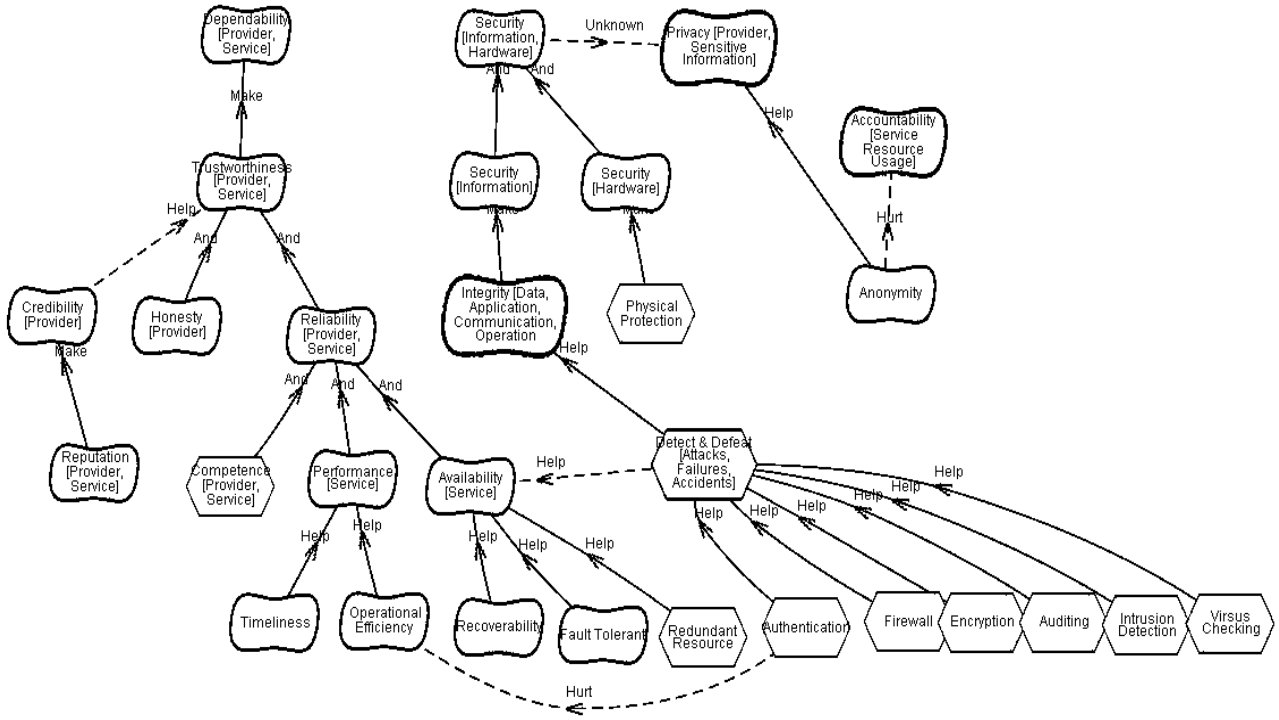

Figure 7: Catalogue of security-related requirements for P2P computing.

In Figure 7, we catalogue security-related requirements for P2P. The links between these non-functional requirements represent the contributions or side-effects of one softgoal to another. Security requirements originate from the intention of an actor to depend on others (providers) for some service. Thus, the Dependability of the provider needs to be decided (one of the highest-level goal in the model). If a provider is trustworthy for a given service, then she is dependable (the contribution link from Trustworthiness to Dependability is *Make*, which means "sufficiently positive"). The provider is trustworthy if she is Honest and Reliable (both are necessary, so modeled as *And* contributions) [Mayer95].

Trust implies less need for security protection [Schneider99]. In a situation of lower risk, people tend to give more trust, while in a high risk environment, people rely more on protection than on trust. Credibility of the service provider enhances trust (represented as a *Help* correlation link). Reliability depends on the provider's Competence, also satisfactory Performance and Availability of the service in question [Mayer95].

Security interacts with other non-functional requirements. For example, the security measure Detect & Defeat [Attacks&Failures&Accidents] helps keep the system up, therefore it *Help*s Availability as a side effect. The nature of such interactions can be hard to determine (*Unknown*), operationalized from softgoals into tasks, the impact of Security on Operational Efficiency can be determined for the specific operationalization Authentication. In an open network environment such as P2P, Privacy is another major concern. In a secure system, the privacy of the participants is protected from external attackers, but is exposed to internal actors such as administrators. In such a case, we use parameters e.g. Privacy [whom, what] to indicate what information is to be protected, and for whom. The catalogue is not meant to be an exhaustive and precise description of all the interactions between the requirements related to security. Instead, it serves as a general reference for supporting an initial analysis.

The link types are used to propagate labels in a qualitative reasoning procedure when the catalogued knowledge is applied in a specific situation. *And* means that all subgoals are necessary for satisfying the parent goal. *Help* means that the subgoal contributes positively, but not sufficiently to fulfil the parent goal. *Hurt* means a negative but insufficient contribution to deny the parent goal. *Make* means that the contribution is positive and sufficient, *Break* means negative and sufficient. *Some+* means positive to an unknown degree, *Some-* means negative to an unknown degree [Chung00]. These coarse-grained qualitative distinctions are used during early stage requirements analysis to determine whether further goal refinements and search for solutions are warranted, based on the notion of satisficing [Simon81].

Subsequent analysis may use finer-grained evaluation methods and specific metrics for various classes of non-functional requirements.

4.4 Case studies: analyzing attacker's intentions and possible means of attack

Below, we illustrates two case studies of the application of the proposed approach. First we put on the shoes of a random attacker on the Internet, and look into his intentions and his different means of attack. Then a legal attack situation for Napster is modeled to illustrate the forming of copyright holder's attacking intention and means of attack.

4.4.1 Case 1: Internet attacks

At the agent level of the hierarchy in Figure 5, surrounding players of the P2P system come into the picture, e.g., ISP server. Figure 8 focuses on the agents related to Internet service. Note that the dependencies between agents include not only ones inherited from the roles they play, but also ones that only apply to the agents themselves. For example, User [BOL Internet Service] depends upon BOL for Network Connection, while the dependencies between general User and Provider still apply. At the same time, specific needs such as Friendlier User Interface can also be directed to BOL. As things have been pushed to the instance level, dependencies between agents playing the same role also need to be identified. In the model, we can see that Internet Users depend on each other for Integrity [Operation & Communication]. The intent of potential attacker Malicious Jim is to do harm to the Internet in order to prove prowess as a technology talent. This is represented as a goal of Be the Dark Lord of the Internet. There are several ways to achieve this goal (connected with means-ends links). For example, he may Steal Other User's Account, do Salami Attack or Replay Violation, send Virus or Worm, Intercept Network Traffic [Sensitive Information], Cause Denial of Service by flood or clog, etc. These attacks *BREAK* integrity dependencies between other users and the ISP. The break of dependencies can have other undesirable effects, such as rendering some service unavailable (represented with the side effect *Help* link to the dependency link connecting Availability). Once the attacker's intentions and methods of attack are analyzed, we can equip the system with counter-measures accordingly.

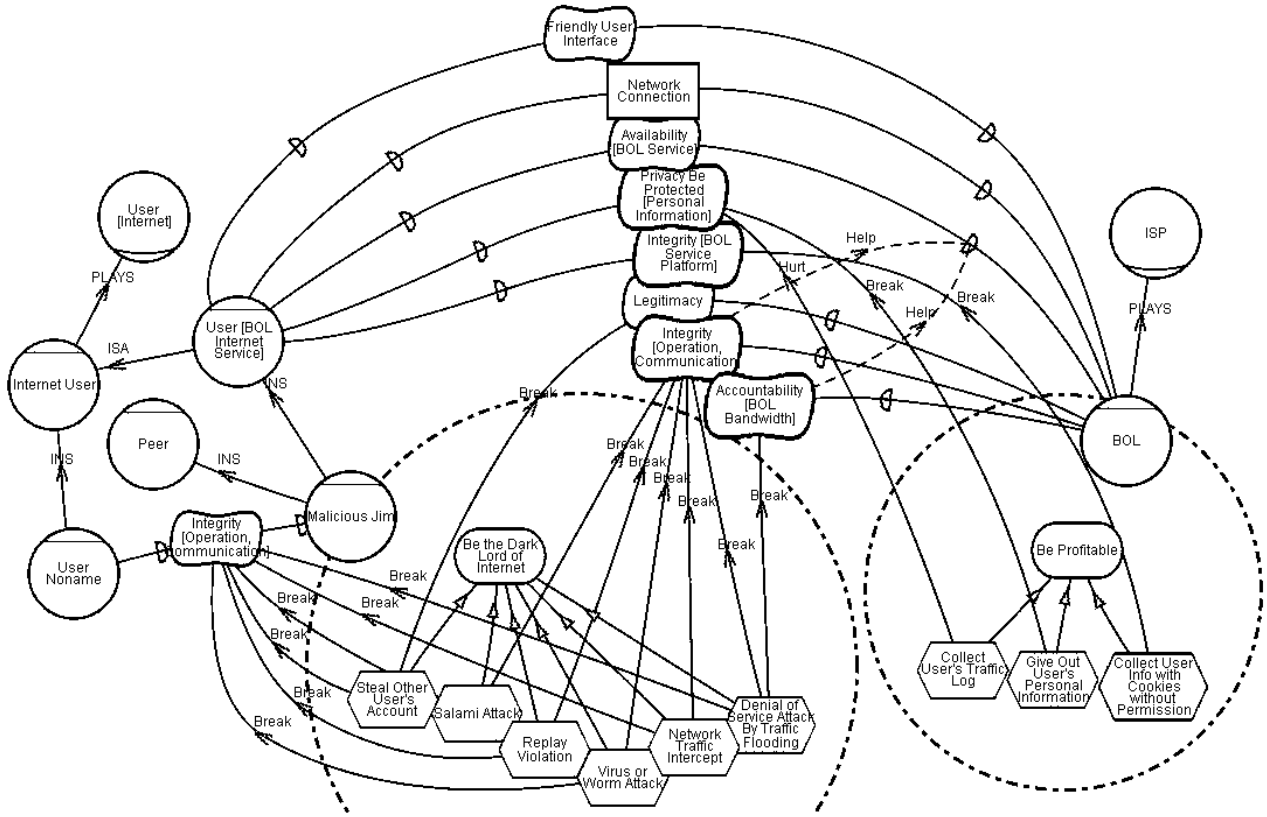

Figure 8: Attacks on the Internet.

4.4.2 Case 2: Legal Attack

In this case study, we show how the conceptual modeling approach can be used to analyze different situations in agent system. P2P systems exchanging different services require different levels of security, and there are cases that the exchanged service is patented or protected by copyright. Figure 9 is a simplified model of the players involved in the Napster lawsuit. In general, a User [Music Copyright] depends on the Provider [Music copyright] for Permission [Music Copying], while the Provider depends on the User to be Legitimate, who should Buy License [Music Copying]. In reality, the Copyright Holders assume that Napster Peer is playing the role of User [Music Copyright]. Napster Peers do not agree on this *Plays* relationship. This disagreement causes the Napster Peer's infringement of music copyright, because they are not going to buy licenses from the copyright holders (dotted *Break* link to the dependency Buy License[Music Copyright]), and they are therefore illegitimate users (dotted *Break* link to the dependency Legitimacy). The model shows that the Copyright Holders delegate the task of Settle Infringement in Court to RIAA, who files a lawsuit to the Court, so that the Napster Peer's copyright infringement behaviors can be prevented (dotted *Break* link to the dependendency link of Availability). [Yu01b] explores the use of the strategic modeling approach to analyze intellectual property management.

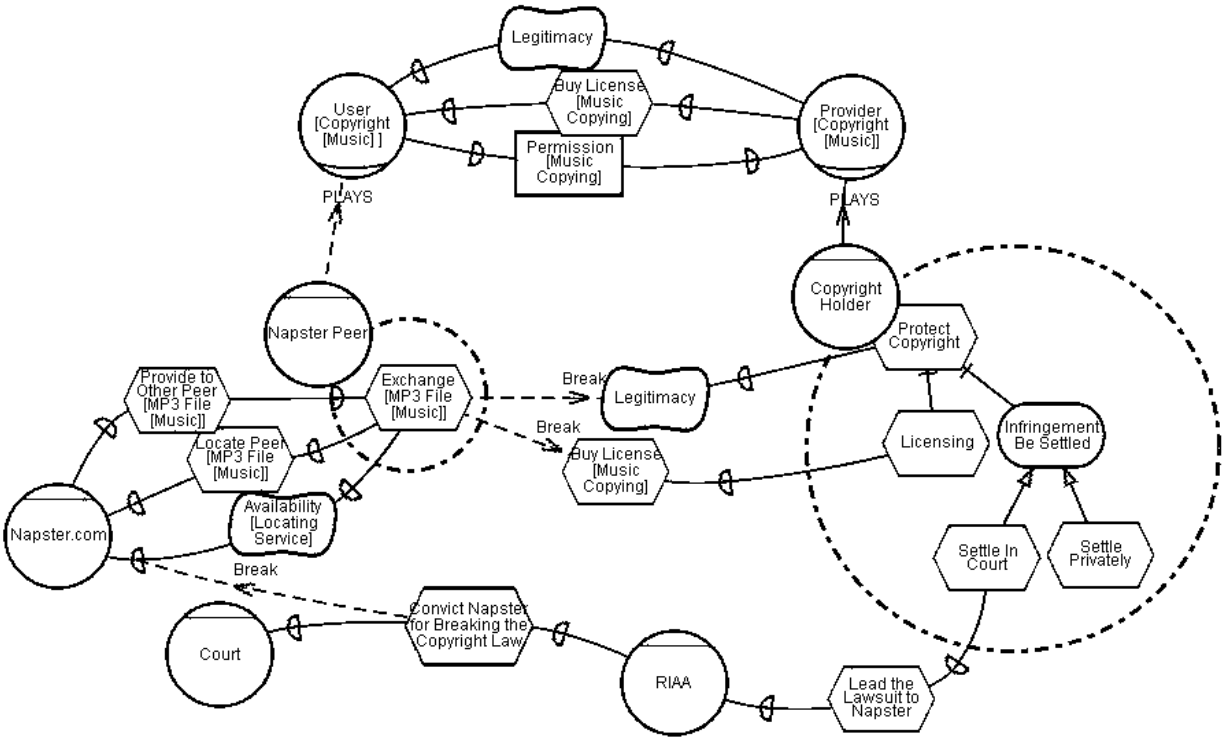

Figure 9: Legal attack.

4.5 Trusting vs. protecting – goal-oriented analysis and decision making

We next give a more elaborate example of goal refinement and how such refinements within an actor can lead to new dependencies among actors. With most of the surrounding players analyzed, we now have an idea about how secure the environment is, who are the possible attackers within and outside the system, what counter-measures can be sought and built into the system. As peers are free to enter or leave the peer network, and in choosing partners, we need to look at the collaborating pairs in the system, where most of the threats causing serious consequences come from (i.e., the insider attacker). In P2P, collaborative, or distributed systems, trust plays a central role in many respects.

Between a pair of cooperating peers, either one depends on the other for some service or they exchange services. Figure 10 models the decision-making structure from one agent's viewpoint. Before Exchange[Service] with Peer B, Peer A needs to first assess the dependability of Peer B. Thus Dependability becomes a sub-softgoal of the task Exchange[Service] (connected with decomposition link). Based on both the common sense knowledge (the catalogue provided in section 4.4) and the current situation, Peer A considers whether the sub-softgoals of Dependability can be satisfied. One subgoal

is whether Peer B is Trustworthy for the delegation of a specific service, the other is whether delegation of a service to B, introduces any threats to security (Be Secure). The sub-factors determining the trustworthiness of Peer B are whether B is Competent in providing the service, and whether B has Goodwill. These could be determined by Peer A's past experiences with B, or from the recommendation of a trusted partner - say, Peer C or from the record of some trusted third party, Authority. For Peer A to Be Secure, Privacy needs to be protected by B, and other security protection mechanism needs to be added as well. However, both of these two will only partially enhance the security (*Help* contribution). Privacy entails exposing as little sensitive information to Peer B as possible, while using Remailer to keep Anonymity is one means to reduce exposure. Protection mechanisms also include Virus Checking and Signature Verification.

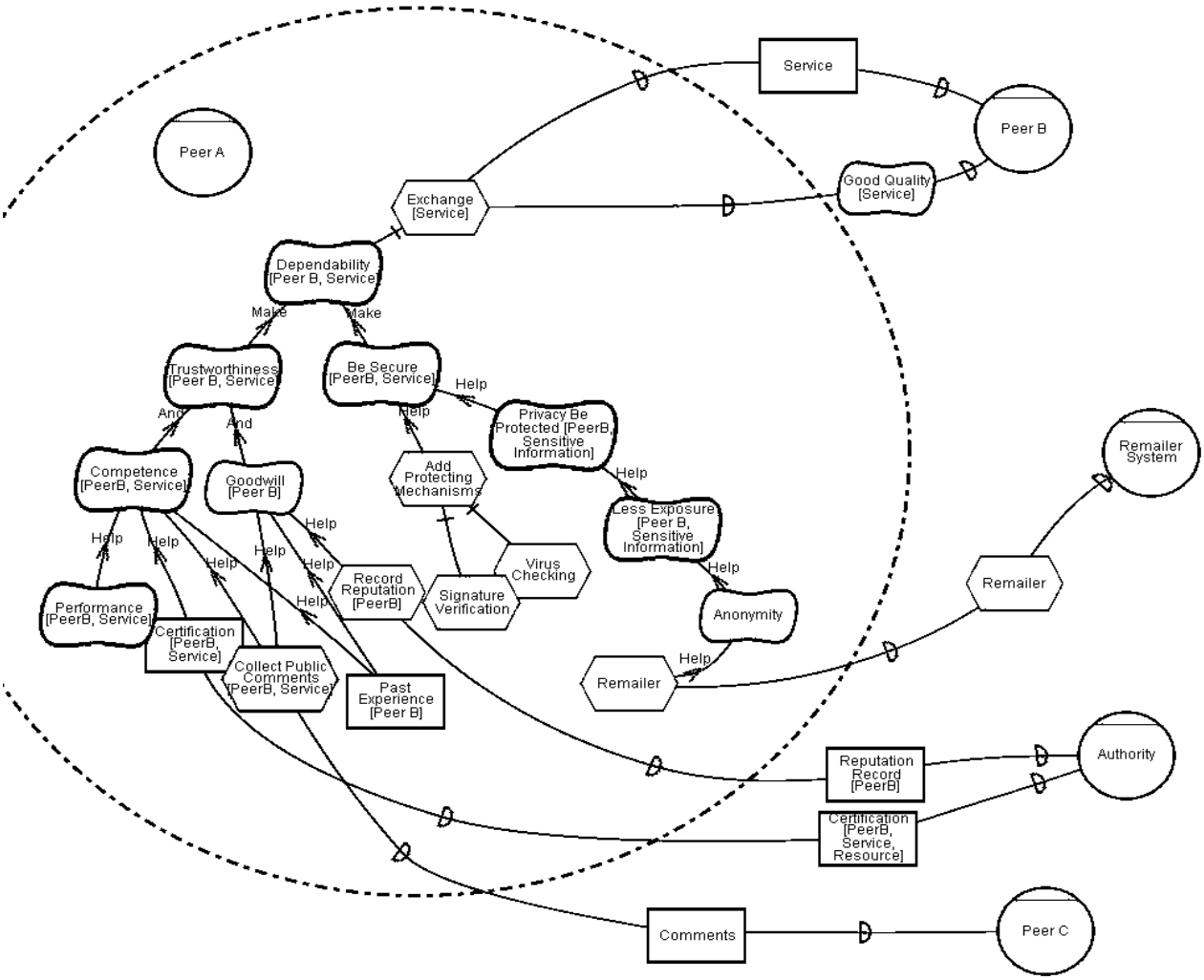

Figure 10: Trust vs. Protection.

4.6 Attack scenario analysis

So far, we have focused on the intentions of attackers and the potential consequences of their attack. More elaborate attacking and protection behaviours are explored in Figure 11. According to the model, the attacker peer intercepts an uploaded document to trace the author. This is modeled as an unexpected branch in the scenario path [Liu01], where the attacking action Trace Document to Author By Network Intercept *Break*s the peer's Untraceable dependency to the Web Server. This preliminary example scenario suggests a possible next step of the modeling process.

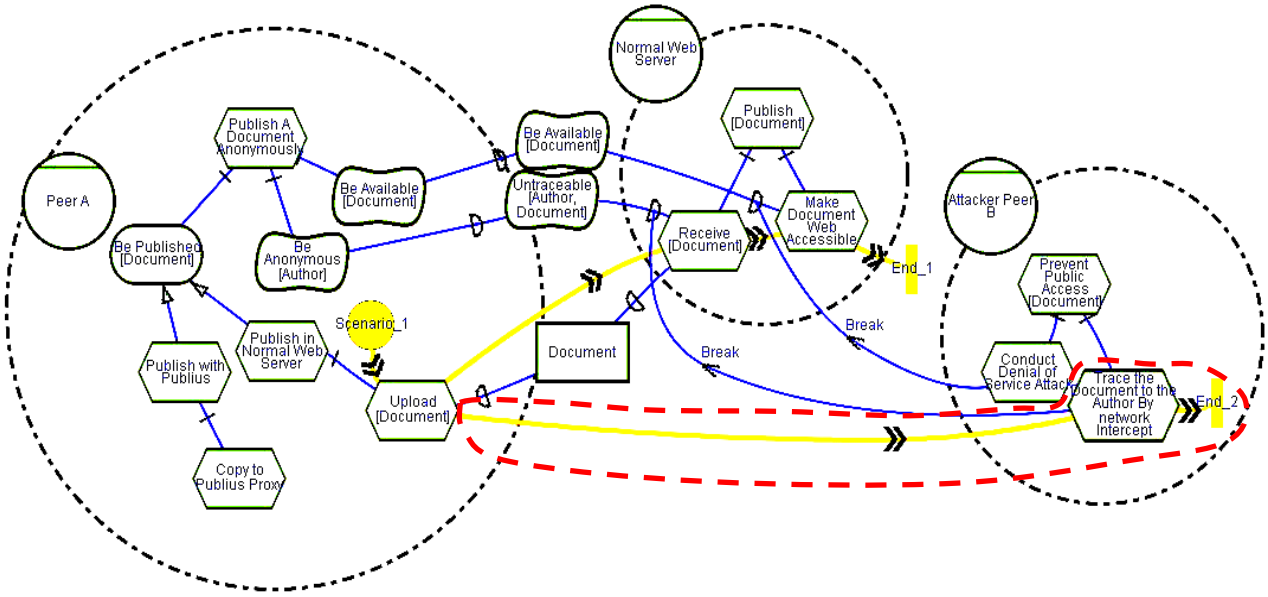

Figure 11: Example of an attacking scenario.

5. Related work

In the security literature, role-based analysis is primarily used for access control. In such works, roles are associated with permissions and privileges for different classes of users [Sandhu96], and hierarchical structures of roles are formally analyzed. In an open environment such as the Internet, users do not belong to one administrative domain, and they may act according to several roles simultanously. To identify and address potential security threats, interactions between actors playing these roles need to be considered as well.

Goal-oriented requirements engineering has received increasing attention over the past few years [Lam01]. Goal types and taxonomies are used to define heuristics for goal acquisition, goal refinement and requirements derivation [Lam01][Anton01][Chung00]. In the NFR framework [Chung00], a general catalogue for security softgoals was estabilished, along with operationalization methods. In KAOS [Lam00], *obstacles* are used to capture undesired properties of a system. These can be used to model and analyze security requirements.

Agent orientation is an emerging paradigm for software systems. Agent-oriented requirements modeling are thus intended as abstraction of behaviours of such systems [Yu01c]. In KAOS [Letier02], agents are active system components, whose goals and capabilities are analyzed based on state. In contrast, the *i** framework focused on strategic analysis, which explores relationships at an intentional level. Models resulting from strategic analysis can be further elaborated and adopted to build state-based models in the KAOS style [Castro02].

6. Discussion and conclusions

Security is an increasingly important issue for Internet-based information systems. New kinds of threats keep coming up, and many technologies have emerged to protect systems and data. Consequently, identifying and analyzing security requirements is an important element of the software engineering process [Rushby01]. Although security is generally addressed at a technical level, it is ultimately about social, legal and personal concerns of relevant stakeholders.

Recent research in requirements engineering has developed many modeling notations and approaches that help the elicitation and analysis of non-functional requirements, including security. In this paper, we have proposed a strategic modeling approach that can potentially help the elicitation, identification and analysis of security requirements. According to the *i** approach, security is not treated as an isolated concept with special semantics. Instead, it is treated as a non-functional requirement that arises in multi-agent collaborations. In such settings, agents depend on each other for some service. Some of these services may require more protection than others. For example, MP3 exchanging peers are taking fewer risks than secret sharing peers, so not much protection is required for the former. In P2P systems, the most general collaboration pattern is that of a user vs. a provider. Security is just one class of vulnerabilities (softgoal

dependencies related to security), there are also other non-functional vulnerabilities such as performance, cost, etc., and functional vulnerabilities implied by the goal, resource, or task dependencies.

This paper applies the *i** modeling framework to model and analyze security concerns in a P2P setting. In an actual application, one could expect the analysis to be more elaborate, taking more relevant factors into account. Understanding its security requirements makes it possible for protocol designers to furnish the environment with countermeasures against potential threats. Participants of a specific P2P system may also have a better idea of whom to trust in the system, what service and information of what sensitive degree can be exchanged in such an environment. Prospective partners may decide which peer community to join regarding the different security situations they may get into.

This approach is complementary to and benefits from the various theories and techniques currently being developed for specifically addressing security (e.g., [Schneier99]). *i** offers a structural representation of intentional relationships among actors and within actors, as well as structural concepts such as intentional agents and roles. These concepts provide a structural framework for integrating other concepts and techniques for dealing with security. For example, the security considerations in the JXTA platform [Sun] for P2P computing can be represented and analyzed using our approach. The structural representation offered by *i** shows dependency relationships among actors which give rise to security issues. At different JXTA layers, different abstract roles can be defined and analyzed. Softgoals and other functional goals of the roles in one layer are systematically refined and reduced until they are operationalized. Therefore, security considerations are taken into account while other associated goals are being addressed.

The *i** approach encourages and facilitates the analysis of security-related issues within the full operational and social context of relevant actors. The models can be used to encompass normal-case operational procedures, potential attacks, countermeasures against perceived threats, as well as factors not directly related. Although security is examined in the P2P domain, it is not a hard-wired concept in *i** framework, it is flexible enough to handle the different security concerns that may apply to a certain context or problem domain.

In the *Tropos* project [Castro02], a formal specification language *Formal Tropos* has been developed based on *i** concepts. Formal verification and validation techniques, such as model checking, are used to validate the adequacy and accuracy of *i** models [Fuxman01].

A companion paper outlines an agent-oriented goal refinement requirement process to address privacy during early stage design [Yu02]. This present paper focus on patterns of relationships among strategic actors and how they can be combined for analyzing security requirements. A prototype tool has been developed to support modeling and reasoning based on the *i** framework. A version of *i** is being proposed as part of a user requirements notation under ITU-T study group 17 [URN].

Acknowledgements. Financial support from the Natural Sciences and Engineering Research Council of Canada, Communication and Information Technologies Ontario, and Mitel Corporation are gratefully acknowledged.

Reference

[Anton01] Antón, A.I., Carter, R.A., Dagnino, A., Dempster, J.H. and Siege, D.F. Deriving Goals from a Use Case Based Requirements Specification, Requirements Engineering Journal, Springer-Verlag, Volume 6, pp. 63-73, May 2001.

[Castro02] Castro, J. Kolp, M. and Mylopoulos, J. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. To appear in Information Systems, Elsevier, Amsterdam, The Netherlands, 2002.

[Chung00] Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, 2000.

[Fuxman 01] A. Fuxman, M. Pistore, J. Mylopoulos, P. Traverso. Model Checking Early Requirements Specifications in Tropos, Proceedings Fifth IEEE International Symposium on Requirements Engineering (RE01), August 27-31, 2001, Toronto, Canada.

[Lam00] Lamsweerde, A. V. and Letier, E. Handling Obstacles in Goal-Oriented Requirements Engineering, IEEE Transactions on Software Engineering, Special Issue on Exception Handling, Vol. 26 No. 10, October 2000, 978-1005. Available at: //ftp.info.ucl.ac.be/pub/publi/2 000/TSE-Obstacles.pdf.

[Lam01] Lamsweerde, A. V. Goal-Oriented Requirements Engineering: A Guided Tour, Proceedings RE'01 - 5th IEEE International Symposium on Requirements Engineering, Toronto, August, 2001, pp. 249-263. Available at: //ftp.info.ucl.ac.be/pub/publi/2001/RE01.pdf.

[Letier02] Letier, E. and Lamsweerde, A. V. Agent-Based Tactics for Goal-Oriented Requirements Elaboration, Proceedings ICSE'2002 - 24th International Conference on Software Engineering, Orlando, May, 2002. Available at: //ftp.info.ucl.ac.be/pub/publi/2002/Icse02.pdf.

[Liu01] Liu, L. and Yu, E. From Requirements to Architectural Design - Using Goals and Scenarios. ICSE-2001 Workshop: From Software Requirements to Architectures (STRAW 2001) May 2001, Toronto, Canada. pp. 22-30.

[Mayer95] Mayer, R. C., Davis, J. H. and Schoorman, F D. An integration model of organizational trust, Academy of Management. The Academy of Management Review, Vol. 20(3), Jul 1995.

[Oram01] Oram, A. Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology. O'Reilly & Associates, 2001.

[Rushby01] John Rushby. Security Requirements Specifications: How and What. SREIS'01.

[Sandhu96] Sandhu, R., Coyne, E., Feinstein, H. and Youman, C. Role-Based Access Control Models, IEEE Computer, vol. 29, no. 2. Pp. 38-47. Feb, 1996. . Available at http://citeseer.nj.nec.com/107920.html.

[Schneider99] Schneider, F. B.: Trust in Cyberspace. Committee on Information Systems Trustworthiness, Computer Science and Telecommunications Board, National Research Council. Washington, D.C.: National Academy Press, 1999.

[Schneider99b] Schneider, F. B.: Reinventing Security, Trust in Cyberspace. National Academy Press, 1999. Pp.109-153.

[Schneier99] Schneier, B.: Attack Trees Modeling Security Threats. Dr.Dobb's Journal, December 1999. Also available at http://www.counterpane.com/attacktrees-ddj-ft.html.

[Simon81] Simon, A. H. The Sciences of the Artificial, Second Edition. Cambridge, MA: The MIT Press, 1981.

[Sun] Sun Microsystems. Security and Project JXTA. Available at: http://www.jxta.org/project/www/docs/SecurityJXTA.PDF.

[URN] URN Focus Group. User Requirements Notation. http://www.itu.int/ITU-T/studygroups/com17/urn/index.html. Also available at http://www.usecasemaps.org/urn.

[Yu97] Yu, E. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97) Jan. 6-8, 1997, Washington D.C., USA. pp. 226-235.

[Yu01a] Yu, E. and Liu, L. Modelling Trust for System Design Using the *i** Strategic Actors Framework. In: Trust in Cyber-Societies - Integrating the Human and Artificial Perspectives. R. Falcone, M. Singh, Y.H. Tan, eds. LNAI-2246. Springer, 2001.pp.175-194.

[Yu01b] Yu, E., Liu, L. and Li, Y.. Modelling Strategic Actor Relationships to Support Intellectual Property Management. Proceeding of 20th International Conference on Conceptual Modeling (ER-2001). Yokohama, Japan, November 27-30, 2001.

[Yu01c] Yu, E. Agent Orientation as a Modelling Paradigm. Wirtschaftsinformatik. 43(2) April 2001. pp. 123-132.

[Yu02] Yu, E. and Cysneiros, L.M. Privacy, Security and Other Competing Requirements, accepted to Symposium on Requirements Engineering for Information Security (SREIS'02), Raleigh, North Carolina, Oct 15-16, 2002.