Organizational Patterns for Security and Dependability: from design to application

Y. Asnar, F. Massacci and A. Saidane
University of Trento, Italy
C. Riccucci
Engineering Ingegneria Informatica S.p.A, Italy
M. Felici and A. Tedeschi
Deep Blue, Srl, Italy Italy
P. El Khoury, K. Li and M. Seguran
SAP Research, France
N. Zannone
Eindhoven University of Technology, The Netherlands

ABSTRACT

Designing secure and dependable IT systems requires a deep analysis of organizational as well as social aspects of the environment where the system will operate. Domain experts and analysts often face security and dependability (S&D) issues they have already encountered before. These concerns require the design of S&D patterns to facilitate designers when developing IT systems.

This article presents our experience in designing S&D organizational patterns, which we have gained in the course of an industry lead EU project. We use an agent-goal-oriented modeling framework (i.e., the ***SI**** framework) to analyze organizational settings jointly with technical functionalities. We demonstrate how this framework can assist domain experts and analysts in designing S&D patterns from their experience, validating them by proof-of-concept implementations, and applying them to increase the security level of the system.

Keywords: Security, Dependability, Organizational patterns, Socio-technical systems.

INTRODUCTION

Security and Dependability (S&D) are critical aspects in the development of IT systems (Anderson 2001). The usual approach towards the inclusion of S&D concerns within a system is to identify security requirements after system design. Unfortunately, this makes the process inefficient and error-prone, mainly because security mechanisms have to be fitted into a pre-existing design which may not be able to accommodate them.

The literature in requirements engineering has highlighted the importance of analyzing S&D aspects since the early phases of the software development process (Giorgini et al. 2005a, Liu et al. 2003). It is also well accepted that S&D cannot be considered as purely technical issues but should be analyzed together with the organizational environment (Anderson 1993). In this direction, goal-oriented approaches (Dardenne et al. 1993, Bresciani et al. 2004) have gained momentum in the community showing their relevance to model and analyze security issues within the organizational setting. This has spurred the definition of several goal-oriented frameworks for security requirements engineering (e.g., Giorgini et al. 2005a, Elahi et al. 2007).

Requirements analysis, thus, is an iterative process where domain experts and analysts have to collaborate to elicit and analyze S&D requirements, besides the functional requirements of socio-technical systems. Often these security needs are common or "similar" to problems that security experts have seen before, and consequently the solution can be "similar" as well. The idea of using S&D patterns to provide solution to security requirements stems from this simple observation above. Patterns have been adopted into software engineering as a method for object-based reuse (Gamma et al. 1994) and security patterns (Yoder et al. 1997, Schumacher 2003) have been proposed to capture and structure collective experience in the S&D domains and make this know-how available and exploitable for application designers. This transfer of knowledge is intended to improve the quality of the developed systems from an S&D point of view. However, most S&D patterns presents in the literature are technical patterns. Here we focus on organizational patterns where we consider the overall interactions between human and software components and the relative dependencies.

This article presents the process for capturing, validating, and applying S&D organizational patterns. Our proposed patterns have been used in widely different industrial contexts: Air Traffic Management (ATM) and e-Health Smart Items. In our work, we have adopted the ***SI**** modeling framework (Massacci et al. 2008), an agent, goal-oriented framework that extends the ***i**** framework (Yu 1997) with the concepts of ownership, trust, delegation, and event for capturing and analyzing security and trust requirements of socio-technical systems (Giorgini et al. 2005a), designing access control policies (Massacci et al. 2008), and risk analysis (Asnar et al. 2008). In this article, we show how the ***SI**** framework has been used by industries for the capturing of S&D organizational patterns and their validation by proof-of-concept implementation. We also discuss how patterns can be applied to a system so that it has a sufficient level of security.

The article is organized as follows. Next, we introduce S&D organizational-patterns, their elicitation process using the ***SI**** modeling language (§2) and the formal frameworks underlying ***SI**** for pattern validation (§3). We present an excerpt of our library of S&D organizational patterns (§4) and describe how these patterns can be used in real systems (§5). Finally, we discuss related work and conclude with lessons learned from the case studies (§6).

DESIGNING S&D ORGANIZATIONAL PATTERNS

At the organizational level, a system can be seen as a set of interacting actors (e.g., humans, organizations, software agents), each of them is in charge of a set of goals that must be accomplished. A S&D pattern at organizational level must therefore capture how these actors and the relationships among them can be evolved in order to meet a S&D requirement. Along the line suggested in our previous work [Busnel et al. 2008, Campagna et al. 2009], we define S&D patterns as triples of the form <*Context; Requirement; Solution*> where:

- **Context** defines the context and the conditions of applicability of the pattern;
- **Requirement** encodes the needs of a system to enhance its S&D level;
- **Solution** specifies how the requirement is achieved.

An organizational solution corresponds to a list of modifications applied to an initial system where the S&D requirements are not fulfilled; and that results to a different system where they are fulfilled. These modifications aim to revise the structure of the organization (when it is possible) or the procedures and policies governing the system. Organizational solutions can be

various and can be implemented, for instance, by adding software or hardware components, by requiring a contract signed by two actors to solve a lack of trust, or throughout a new distribution of efforts among the actors involved in the system.

In order to explain how such patterns are constructed we will use the ATM scenario as a running example.

> **Case study 1** ATM is the dynamic and integrated management of air traffic flow to minimize delays and congestion while guaranteeing safety and efficiency of operation in the airspace. In particular we focus on Air Traffic Controllers (ATCOs) who are in charge of issuing instructions to flight crews in order to maintain separation among aircraft. The airspace is statically organized into adjacent volumes, called sectors, and airways. Each sector has a predefined capacity (i.e., the maximum number of flights that can be safely managed) and is operated by a team of two ATCOs, consisting of an Executive Controller (EC) and a Planning Controller (PC), working together as a team and sharing the responsibility for safe and efficient operations within the sector (Serenity Consortium 2008).

The process for the design of S&D organizational patterns (Table 1) starts with the elicitation of system requirements. Initially, analysts together with domain-experts identify the stakeholders relevant to the system and categorize them into three different types of actors: role, agent, and group. Afterwards, stakeholders' goals are also identified and refined using goal analysis (i.e., refinement, contribution, and means-end analysis). Afterward, the objectives, entitlements, and capabilities of each actor are identified.

Table 1- Design of S&D organizational patterns

Step	Description
A Elicit requirements	
A.1 Identify stakeholders	Identify the actors involved in the system along their goals
A.2 Identify strategic rationale	Analyze the objectives, entitlements, and capabilities of actors
A.3 Identify trust relationship	Analyze actors' expectation concerning the behavior of other actors
A.4 Identify dependencies	Identify execution dependencies and permission delegations between actors
A.5 Identify risk factors	Identify uncertain events that may obstruct the system and define the acceptable risk level
B Analyze elicited requirements	Analyze the model whether it has fulfilled S&D requirements and risks are below the acceptable level
C Refine requirements	Improve the model such that it fulfill S&D requirements
D Abstract the context and solution	Generalize identified context and solution, for easy reuse
E Validate the pattern	Verify whether the pattern actually satisfies S&D requirements using the OR or the ST Solver

Notice that the design process does not explicitly separate the S&D features (eg entitlements, and risk factors) from functional features (eg functional objectives and system obstructions). They are dealt with in the same step. In traditional Software Engineering, analysts model functional requirements and afterward elicit and analyze S&D requirements. This approach is fragile because it could happen that S&D requirements conflict with functional requirements (Example 1), are not effective (Example 2), or even that functional requirements themselves are the source of risks (Example 3).

> **Example 1** The ATM system must provide necessary data easily for performing any Air Traffic Control (ATC) related activities. As S&D requirement, ATCOs should be authenticated each time

they want to use any service provided by an ATM system (i.e., including gathering any data: flight plan, weather, runway, map, etc.).

In practice, the authentication requirement hinders the easiness of ATCOs in obtaining necessary data. Analysts must manage potential conflicts through trade-off decisions that best meet their requirements. For instance, analysts can employ authentication mechanisms only for critical activities of the ATM system (e.g., controlling aircraft) and not for non-critical activities (e.g., gathering weather data).

> **Example 2**. Flight plans should be kept confidential by employing some encryption techniques. Unfortunately, this security measure will not be effective because each flight plan is printed in a flight strip to ease ATCOs' work. Therefore, analysts should elicit other ways to maintain the confidentiality of flight plans.

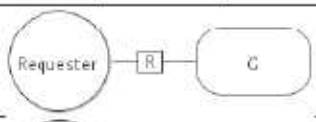
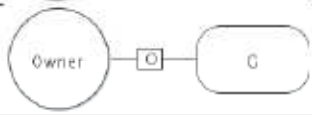
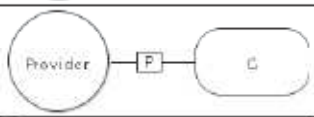
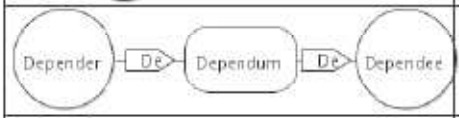
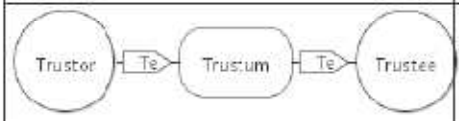
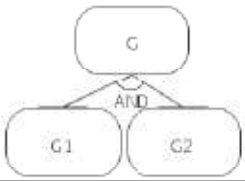
A functional requirement can also be the source of a vulnerability that might compromise the security and dependability of the system.

> **Example 3**. The ATM system is required to be connected with Internet to provide arrival and departure time in the real time, but it is too risky because it allows hacker attacks from the Internet.

Working with S&D and functional requirements at the same time avoid these problems.

To support the steps in Table 1 we need to have a language that is more precise than natural language and more amenable to verification. In this work, we have adapted the ***SI**** modeling framework (Asnar et al. 2008, Massacci et al. 2008), an extension of the ***i**** framework (Yu 1997) with concepts specific to security and dependability. The ***SI**** framework borrows from ***i**** the concepts of actor, goal, task, resource, and introduces the concepts of objective, entitlement, capability, delegation, trust, and event for modeling and analyzing S&D risks and requirements. We present them in Table 2 along with their graphical notation.

Table 2 - SI Concepts and Relationships*

	Agent	an active entity with concrete manifestation
	Role	abstract characterization of the behavior of an active entity
	Goal	strategic interest of an actor
	Resource	physical or informational entity
	Request	denote the objectives of actors
	Own	denote entitlements of actors
	Provide	denote capabilities of actors
	Execution Dependency	transfer of objectives (or permissions) from the depender to the dependee
	Trust of Execution	trustor's expectation about the ability and dependability of the trustee
	Decomposition	combines AND and OR refinements of a root goal into subgoals, modeling a finer goal structure
	Contribution	identify goals and tasks that contribute positively or negatively in the fulfillment of the goal
	Means-end	identify goals, tasks, and resources that provide means for achieving a goal

Interested readers are referred to (Asnar et al. 2008, Massacci et al. 2008) for a comprehensive description of the ***SI**** framework.

Example 4 In the ATM scenario, workers have to cope with complex activities where tight coordination among them are crucial. In particular, where the failure of one subgoal has a negative impact to another subgoal and compromises the correct functioning of the whole activity. Paula (PC) and Luke (EC) are part of Sector SU-Team which is responsible to achieve

"managing air traffic" in Sector SU, and each of the team members is in charge to achieve part of the aircraft management duties.

The identifications of stakeholders is easy in this example (A.1 and A.2) and the process proceeds with the identification of interdependencies between stakeholder. The ***SI**** framework employs two types of dependencies (see Table 2): an actor (*depender*) can depend on another actor (*dependee*) for the fulfillment of a goal, or an actor (delegator) can delegate the permission to achieve a goal to another actor (*delegatee*). By depending for the fulfillment of a goal, the *depender* becomes vulnerable. The *depender* can fail to fulfill her goal because the *delegatee* does not fulfill the assigned responsibilities. Similarly, a delegator takes risks because the *delegatee* can misuse the granted permission. Indeed, the dependability of a system does not only depend on the reliability of a machine, but also depends on the interdependency among agents in the system.

This is captured by dependencies/delegations to *untrusted* actors. When the delegator grants permission to the *delegatee*, it does not imply that the delegator trusts the *delegatee*. Sometimes, the delegator should grant permission for accessing a resource to someone who he does not trust. This situation emerges when an actor must delegate the fulfillment of a goal or give permission in which it does not have any power to choose another agent (e.g., defined by regulations) or it does not have any other option (e.g., no trusted *delegatees*).

Moreover, analysts need to identify uncertain events that might obstruct the fulfillment of a goal. These uncertain events can have negative or positive impact to the goal; risk is an uncertain event with negative impact, while opportunity is the one with positive impact. To mitigate a risk, one can employ a treatment (depicted as a task) that reduces the likelihood of the event or alleviates the negative impact of the event.

Example 5 A planner (i.e., Paula) sometime is exposed to an event of "immediate traffic increase" that hampers the achievement of the goal "plan flights". Some measures (e.g., partial air space delegation, air traffic limitation policy, assist traffic planning) can be used to mitigate the risks introduced by the event, but analysts must analyze carefully the side-effects of those measures.

Figure 1 shows the ***SI**** model resulting from this discussion in the concrete case of Example 4.

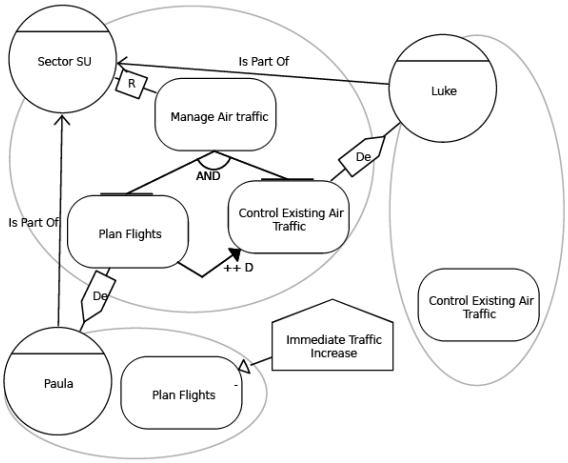


Figure 1- Initial organizational structure within ATC Team

Once requirements have been captured and modeled, domain experts and analysts should verify if the designed system satisfy S&D requirements. This can be done, for instance, using the formal frameworks behind the ***SI**** framework. The analysis may end up with the identification of some problems (eg events) that can obstruct the fulfillment of the S&D requirements.

> **Example 6** In Figure 1, the goal "plan flights" is threaten by the event of "immediate traffic increase" which is defined as unavailability of the flight plan. The occurrence of the event obstructs the achievement of "plan flights" which later can increase the (D)enial evidence for the achievement of "control existing air traffic".

Domain experts need to revise the requirements and provide solutions that increase the security level of the system. A possible solution is depicted in Figure 2. It consists in adding mitigation "assist traffic planning" performed by a team mate (Luke), such that the risk is less severe.

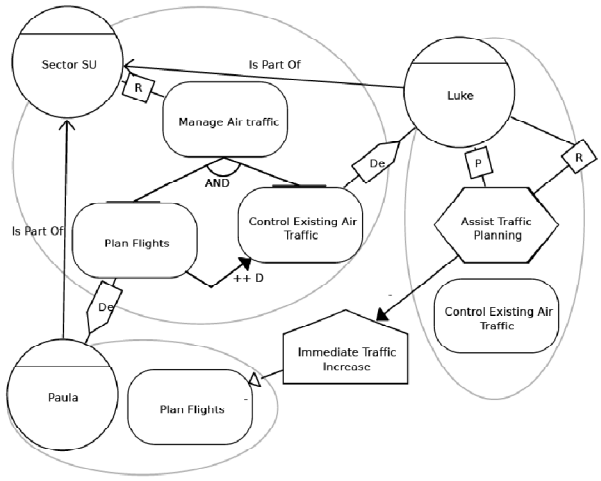


Figure 2 - Revised organizational structure within ATC Team

To generate an S&D pattern the context analyzed in Examples 4 and its solution should be abstracted from the contingent situation. This abstraction is the key step in the definition of S&D patterns. An abstraction of the solution is depicted in Figure 3. The resulting pattern consists of application pre-conditions (i.e., the context), the requirements to be fulfilled, and the solution.

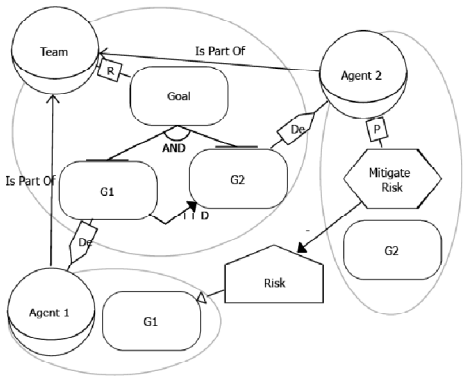


Figure 3 - Collaboration in Small Group for a Risky Activity

SECURITY AND DEPENDABILITY ANALYSIS

The aim of ***SI**** analysis (Figure 4) is to formally verify whether S&D patterns fulfill the S&D requirements for which they have been designed. Essentially, this analysis is founded on two basic formal frameworks: Secure Tropos (Giorgini et al. 2005b) that verifies security properties (e.g., availability, authorization) and Tropos Goal-Risk (Asnar et al. 2007) that assesses the level of risk. In previous works, we have presented the formal definition of ***SI**** models. This analysis can be performed using the ***SI**** Tool[1] .The ***SI**** Tool assists analysts in representing an S&D pattern in terms of ***SI**** diagrams, and then transforms those graphical models into formal specifications and verifies their compliance with S&D requirements.

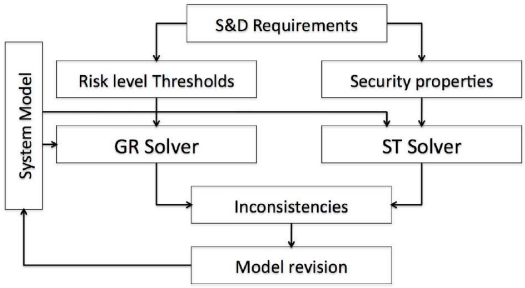


Figure 4 - SI analysis process*

Logical Assessment of S&D Properties (ST Solver)

According this approach (Giorgini et al. 2005b) the semantics of ***SI**** concepts are defined using the *Answer Set Programming* (ASP) paradigm. The ASP paradigm is a variant of *Datalog* with negation as failure. This paradigm supports specifications expressed in terms of facts and Horn clauses, which are evaluated using the stable model semantics. A fact consists of a relation symbol, called predicate, together with an appropriate number of well formed arguments. We distinguish two types of predicates: extensional and intensional.

A predicate is extensional if it does not appear on the left-hand side of any clause. Extensional predicates represent primitive concepts in ***SI****. All other predicates are called intentional, and they are used for requirements verification. The ***SI**** Tool supports the translation of Secure Tropos diagrams into ASP specifications.

> **Example 8** Figure 5 shows the list of facts corresponding to the context diagram of a collaboration in small group pattern given in Figure 3.

To validate S&D patterns, the framework supports formal analysis of rules and constraints (i.e., rules where the head is empty). Rules (or axioms) are used to complete specifications in order to derive the information needed for pattern validation. For instance, designers need to identify who is in charge of achieving goals, performing tasks, and delivering resources. To this intent, Secure Tropos uses the following axiom (Giorgini et al. 2005b):

(Ax1) aim(A,S) ← request(A,S)
(Ax2) aim(A,S1) ← aim(A,S) ^ and decomposition(S,S1,S2)
(Ax3) aim(A,S2) ← aim(A,S) ^ and decomposition(S,S1,S2)
(Ax4) aim(A,S) ← aim(B,S) ^ dependency(B,A,S)

[1] Available at *http://sesa.dit.unitn.it/sistar_tool/*

Ax1 states that an actor aims to achieve a goal, perform a task or deliver a resource if this is one of his objectives. *Ax2* and *Ax3* propagate objectives through AND decomposition, and *Ax4* through dependency.

> **Example 9.** Applying such axioms to our example, one can infer that the Team aims to achieve goal Goal because of Ax1, and G1 and G2 because of Ax2 and Ax3. The reasoning system also infers that role1 aims to achieve G1 and role2 aims to achieve G2 because of Ax4.

The complete specification can be used by designers to verify if the model complies with desirable security properties (Table 3). The framework supports domain experts and analysts through the use of constraints (Gelfond et al. 1991). Constraints are formulations of conditions which must not be true in the model. If all constraints are not simultaneously satisfied, weaknesses or vulnerabilities may occur in the actual pattern solution.

Table 3 - Security Properties

Availability	
Pro1	Actors delegate the execution of their (direct or indirect) objectives only to actors that they trust.
Pro2	Requesters can satisfy their objectives; that is, they have assigned their objectives to actors that have the capabilities to achieve them.
Pro3	Requesters are confident their objectives will be satisfied; that is, they have assigned their objectives to actors that have the capabilities to achieve them and are trusted.
Authorization	
Pro4	Actors delegate permissions on their (direct or indirect) entitlements only to actors they trust.
Pro5	Owners are confident that their entitlements are not misused; that is, permission on their entitlements is assigned only to actors they trust.
Pro6	Actors, who delegate permissions to achieve a goal, execute a task, or furnish a resource, have the right to do so.
Availability & Authorization	
Pro7	Requesters can achieve their objectives; that is, they have assigned their objectives to actors that have both the permissions and capabilities to achieve them.
Pro8	Requesters are confident to achieve their objectives; that is, they have assigned their objectives to actors that have both the permissions and capabilities to achieve them and are trusted.
Pro9	Providers have the permissions necessary to accomplish assigned duties.
Privacy	
Pro10	Permission has been granted to actors who actually need it to perform their duties.

> **Example 10** Analysts need to verify the availability of goal G for an actor team in the model of Figure 3. To verify the compliance of the model with Pro2 in Table 3, Secure Tropos uses the following constraint:
>
> *(Pro2)* ← *request(A,B,S)* ^ *not can satisfy(A,S)*

The analysis reveals the presence of weaknesses in model: the requester cannot satisfy G because he depends for the achievement of the (sub)goals on actors that do not have the capabilities to achieve the assigned duties.

```
role(team).
role(role_1).
role(role_2).
is_part_of(role_1,team).
is_part_of(role_2,team).
goal(goal).
goal(g1).
goal(g2).
event(event).
task(mitigate_risk).

and_decomposition2(goal,g1,g2).
pos_contribution(g1,g2).
neg_contribution(event,g1).
neg_contribution(mitigate_risk,event).
request(team,goal).
provide(role_2,mitigate_risk).
dependency(team,role_1,g1).
dependency(team,role_2,g2).
\normalsize
```

Figure 5 - Extensional Description in ASP

Inconsistencies of properties might be due to either unspecified requirements or conflicting requirements. Thus, detecting and solving inconsistencies helps analysts to detect implicit and unspecified requirements, understand system vulnerabilities, and identify and evaluate solutions for mitigate vulnerabilities. Failing to resolve such inconsistencies can erode the acceptance of S&D patterns by application designers.

Quantitative Risk Estimates (GR-Solver)

To reason about risk in a ***SI**** model we follow the techniques from (Asnar et al 2008).. Essentially, we identify two types of evidence for each construct (being it goal or resource): satisfaction (SAT) and denial (DEN) evidence. SAT quantifies the value of satisfaction evidence of particular constructs - goals to be fulfilled, tasks to be executed, resources to be furnished, and events occurrences. Conversely, DEN represents the evidence that the goal will be denied. These two attributes are quantified into three range qualitative values: full, partial, and none. Notice the concept of evidence differs from the notion of probability. In the probability theory, if the probability of E is 0:34 then we may assume the probability of E does not occur as 0:66 (i.e., P(:A) = 1 –P(A)). However, the evidence of denial cannot be derived for the satisfaction one. Somehow, they appear as two independent variables as proposed in the Dempster-Shafer theory of evidence (Dempster 2008).

The evidence -SAT and DEN- must be provided for each input-nodes and leaf-events such as events or basic objectives. Then forward reasoning propagates those inputs all over the ***SI**** model. Finally, the evidence of the top-goals (i.e., goal Goal in Figure 3) is calculated. By knowing the final evidence of top-goals, one may conclude whether the risk level (i.e., the level of denial evidence) of top-goals is acceptable or not. If not, then mitigation objectives needs to be introduced (i.e., reduce likelihood or severity) as depicted in the solution in Figure 3.

As mentioned in Table 1, the model of the system is finally analyzed to ensure whether S&D requirements are fulfilled and calculate the risk level of the model. On the base of the results, analysts refine the model (the strategic rationale, dependency, or trust relations) such that all the S&D requirements hold and introduce a set of treatments (i.e., tasks) so that the risk levels are acceptable. In Example 5, we want to ensure the availability and the reliability of aircraft management in *Sector SU*. The analysis reveals that the failure of one subgoal has a bad impact to the satisfaction of another subgoal. If *Paula* fails to perform her responsibility to plan incoming traffic, then *Luke* will have difficulties to manage current traffic.

> **Example 11** We want to ensure the safe operation of the team. The aircraft management in Sector SU must always be executed correctly (availability and reliability), even if one of the team member is temporarily absent.

The analysis demonstrates that the system fails if one of the agents is unavailable or faulty.

ORGANIZATIONAL PATTERNS LIBRARY

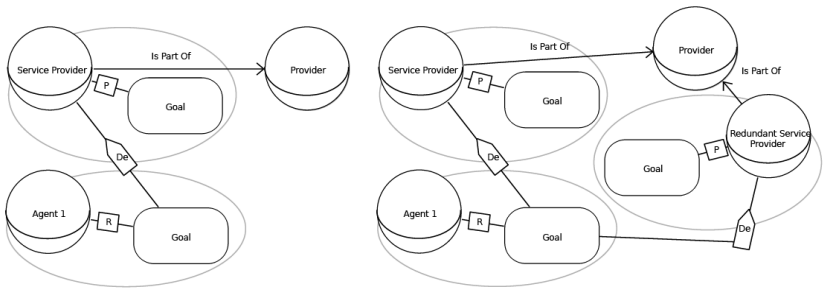
In this section, we provide the natural language description of the organizational patterns extracted from the different case studies. The proposed patterns have been formalized and validated using the ***SI**** framework and detailed information can be found in (Serenity Consortium 2009). The validation aimed at verifying that the new organizational structure of the system fulfils the S&D requirements identified earlier. As explained before, the context and the solutions are specified by means of ***SI**** models. However, in this section we show only few ***SI**** models because of space constraints.

We tried to cover a wide spectrum of security management issues: legal patterns (14), privacy patterns (4), security patterns (11), dependability patterns (15), and a number of integration schemes (5) for combining different kinds of patterns.

As we have used the ATM case study as a running example, we show first some dependability organizational patterns are applicable to different application domains that we can immediately recognize in the Examples that we have just illustrated.

DP2 – Collaboration in Small Groups	
Context	This pattern deals with any situation in which it is essential to cope with complex activities where tight coordination among workers is crucial. The success of an agent (e.g., Agent 1) carrying its task is not only the interest of the agent itself, but also his (or her) team mate (e.g., Agent 2). If the Agent 1 fails to satisfy the goal G1 then the goal G2 is also disrupted. This condition may lead also to the denial of the Goal of the group.
Requirements	Reliability on Goal of the Group and Reliability on G1.
Solution	To prevent the goal G1 from failure, Mitigate Risk should be introduced to mitigate the Risk affecting the satisfaction of G1. Consequently, the goal G1 will not affect G2. In this setting, Agent 2 has capability to perform Mitigate Risk therefore it is its responsibility to perform it. Applying this pattern also means choosing the group size and its composition in terms of members capabilities, expertise, and knowledge.

The solution of this pattern is shown in Figure 3. Instead the following figure describes the context and the solution for another dependability pattern that we have informally discussed.



(a) Context (b) Solution

Figure 6 - Redundancy for Reliability

DP1 – Redundancy for Reliability	
Context	The Agent requests the execution of one critical Goal to the Service Provider. The Service

	Provider is trusted to deliver Goal but the Agent would like to be sure of the results reported by the Goal. The ***SI**** model in Figure 6(a) depicts the context.
Requirements	The Agent shall have enough information to be able to rely on the Service Provider s reported Goal information.
Solution	The Agent shall request the same Goal from a redundant Service Provider and result shall be validated between the two reported Goals. The ***SI**** model in Figure 6 (b) depicts the solution.

DP5 – Assistance Through Experience

Context	This pattern is concerned with effects of prescribed career trajectory amongst tightly coupled groups of workers in safety critical, control room settings. The agents Supervisors have thoroughgoing knowledge and experience of positions below their present standing and therefore, how these different 'sub-roles' work to compose the overall activity. As a result a Supervisor has the ability to monitor, supervise, assess and explain to his sub-roles, as well as being able to fluidly assist and take over the sub-roles of other agents as required.
Requirements	Provide extra dependability trough personnel recruiting and career trajectory.
Solution	Be sure that composing work teams the personnel with higher roles and responsibilities have knowledge and experience of the activities that feed into their work. This pattern provides checking and redundancy: 1. Workers in higher roles are experts in the roles below them. 2. Workers in higher roles have an overall view of the system. 3. Workers in higher roles can supervise the roles below them. Workers in higher roles can carry out the duties of those in roles below if and as required.

In terms of security patterns we have tried to see how far we could get in terms of coverage. In Figure 7, we show the coverage in red of our S&D Patterns on ISO 27001 security management standard concepts in order to show the generality and coverage of security management notions by the considered patterns. Of course such coverage is not exhaustive as there can be significantly more patterns covering a a single aspect of ISO 27001.

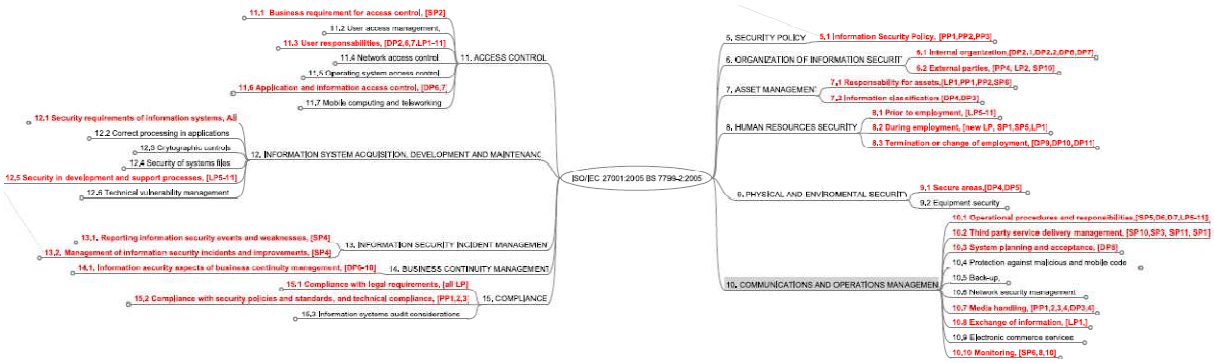


Figure 7 - Our S&D Patterns Library coverage of ISO-27001

In the following, we present some sample of those security organizational patterns.

SP1 – Proof of Fulfilment for Ensuring Non-Repudiation	
Context	To accomplish its daily tasks an organization exploits its social infrastructure by decomposing each task into subtasks that are then distributed/delegated among groups of actors having pre-defined relations. The Delegator requests the achievement of a commitment and delegates its execution to the Executor, but the former has no warranties that the latter takes the responsibility of achieving the commitment.
Requirements	The Delegator shall have evidence that the Executor cannot repudiate his commitment.
Solution	The Delegator refines the commitment into two sub-parts. The first part is used to check the evidence about responsibilities taken by the Executor and the second represents the actual desire of fulfilling the commitment. To achieve the commitment, the Delegator delegates the execution of the commitment to the Executor together with a request for a proof of commitment. Once received, this proof ensures the Delegator that the Executor has taken the responsibility of the commitment.

SP2 – Artefact generation as an audit trail	
Context	This pattern concerns with any situation in which there is the need to share information, keep track of modification and promoting non-repudiation. Agent 1 and Agent 2 share Resource 1. Both Agent 1 and Agent 2 can read, modify and update data stored on Resource 1. This allows Agent 1 and Agent 2 to share information between and to keep track of the history of modifications on Resource 1.
Requirements	It is necessary to enhance accountability by promoting non-repudiation and recording work and traceability of modifications. Additionally, the recorded data need to be stored securely, i.e., secured against unauthorized access and modifications, but this is not part of this pattern.
Solution	Guarantee the possibility to retain details of individual work, allow attachments, comments, on the shared resource. It shows what has been changed, by whom, and why, maintaining a record always available for the workers. The resource is enhanced with an annotation system.

PATTERNS USAGE

At runtime, the patterns library is used for ensuring that the fulfillment of the S&D requirements will be maintained whatever happen in the system and its environment (Serenity Consortium 2008). In fact, the generic S&D solutions are implemented as executable components that are invokable by the applications through their known interface. The system reacts to context change by selecting the appropriate patterns to be deployed. At organizational level, we define different categories of implementations depending on the organizational infrastructure used by an organization to perform its business activities.

In this section, we show the deployment of the previously presented patterns in the context of the different scenarios.

eHealth Smart Items based scenario

In this section we show how our S&D patterns have been deployed in the eHealth scenario.

Case study 2 The objective of remote healthcare systems is to remotely monitor the patient health status and provide the necessary assistance. To reach this objective, healthcare systems should support the interaction and collaboration between doctors, pharmacists, patients, social workers and emergency medical teams especially during emergency situations.

Patient's health condition can be monitored through various wearable medical sensors worn as a washable smart T-shirts. All these sensors form the Body Sensor Network (**BSN**). The measured data are collected and pre-processed by a personal mobile hub such as a Personal Digital Assistant (**PDA**). Similarly, the patient's house is equipped with a sensor network and a local server, which centrally processes the sensor data, for monitoring the activity of the patient and the environmental setting. The information collected by the BSN and smart home are sent to the Monitoring and Emergency Response Centre (**MERC**), the organization responsible for the maintenance and storage of patient's medical data, such as the Electronic Health Record (**EHR**). The MERC processes such data to have a constant snapshot of the patient's health status and promptly initiates proper healthcare procedures when a potential emergency alert is identified. The logical architecture of the prototype is represented in Figure 8.

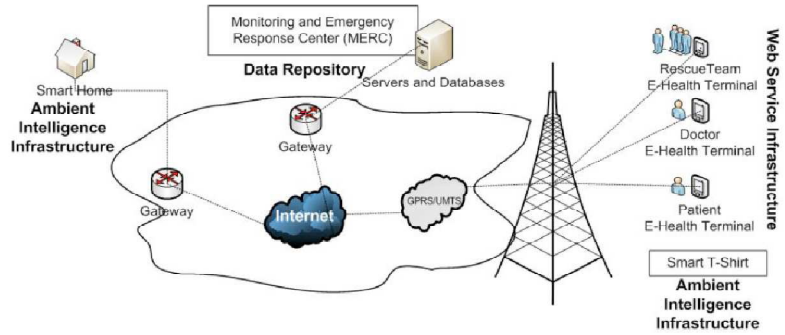


Figure 8 - Logical architecture of the smart items scenario

The smart items prototype aims at providing reliable monitoring capabilities and appropriate facilities to react in case of emergency. These activities are refined into: (1) enhancing situation awareness; (2) support decision making; (3) monitoring actions; (4) giving alerts and requests for urgent actions; (5) improving coordination and cooperation among the different actors. All these activities are tightly connected with the usage of the S&D patterns listed above, as reported in the following table.

Table 4 - Matching prototype functionalities to S&D patterns

Feature	DP1	SP1
Situation Awareness	X	X
Decision Support	X	-
Action Monitoring	X	X
Request for Action	-	X
Coordination	-	X

The pattern **DP1** (Redundancy for reliability), is applied to the Smart Items case study in several points. In the deployment process we followed the guidelines offered. The redundant service provider has to be installed and activated. Results of the redundant service provider has to some extends be homogeneous to the initial service provider's results. In such cases the mean is calculated and provided back to agent 1. If not then, in addition to the mean returned to agent1, a warming is also reported. Hereafter, we particularly highlight at the implementation level how

we applied these guidelines. For example, in the Smart House, we had only one sensor reporting the status (open/closed) of the fridge's door. As the case for several other sensors, we installed a complementary sensor that plays the redundant service provider role. The first detects if the fridge is open and the second detects if somebody is in front of the bridge. Since we want to ensure reliability of the open action of the fridge, we applied the reliability pattern. Thus we used the second sensor as the redundant one function that checks the following: when sensor 1 reports that the fridge s door is open then sensor 2 has to detect that somebody is near the fridge (both sensors return Booleans values so the checking is trivial).

Moreover, the pattern **SP1** (Proof of Fulfillment for Ensuring Non-Repudiation), is also applied in this prototype by using these provided guidelines: The executor has to sign and encrypt a message showing the fulfillment of the assigned work. This message is sent back to the benefiter as a proof of fulfillment. We learnt this best practice from this pattern and we applied it in several location of our prototype. An example of the implementation is applied to the JSP page used by Bob for sending the alert request. An alert is highly critical for the life of the patient. The WS alert WS (invoked by this JSP page) initiates the emergency process in the MERC s workflow. For auditing reasons, the alert WS should be able to proof the fulfillment of its expected actions.

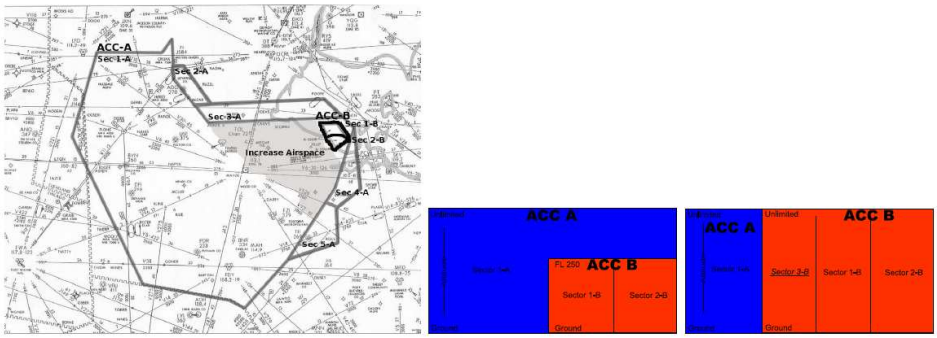
The application of this pattern is by adding a notification operation to the operations supported by the alert WS. This notification is sent by the alert WS to the JSP to acknowledge the initiation of processing of the alert request. This notification signs the alert message identifier by the alert WS (the signature encompasses the name and emergency level of the patient to eliminate some XML types of attacks) and then send it back to the JSP page, that is (required to be) stored as an xml document and that is shown as a notification at the JSP page. Part of the monitoring for this pattern is to verify whether resource holding the proof of commitment has been tampered. Basically, if the resource is not the same as the message sent by the alert WS, then it has been tampered (by checking the hashing).

A live demo of the system including the deployment of patterns is also illustrated in a video

http://www.disi.unitn.it/~massacci/Download/SERENITY-MPEG.mpg

Air Traffic Management scenario

The Air Traffic Management (**ATM**) Scenario distilled robust solutions based on complex socio-technical organizations in order to provide a dependable assistance service to flights. In particular, the ATM Scenario concerns a case of **Re**-sectorisation (Figure 9) and Partial Airspace Delegation resulted from an unplanned increase of air traffic exceeding the sector capacity in an Italian Area Control Centre (**ACC**). The unusual and unexpected increase of traffic is due to two contemporary events: a World Convention in Germany and an **ATCO**'s strike in progress in the France ACCs, at West of Italian ACCs.



(a) Sectors in an Airspace (b) Airspace before re-sectorization (c) Airspace after re-sectorization

Figure 9 - Re-Sectorisation of Airspace

Peculiarities of the ATM Scenario are: (1) the stress on organizational and management aspects of S&D; (2) the focus on safety, dependability and resilience, more than on security. The S&D Patterns drawn from the analysis of the ATM scenario, and described in previous Section, explicitly state the roles that are supposed to be played by human operators. A runtime framework has been developed (ATM Coordination Tool- ATC) to deal with S&D solutions comprising human elements in order to improve the coordination among ATCOs. ATC aims to improve safety and efficiency of ATM by helping ATCOs in their daily work and by performing the following activities: (1) enhancing situation awareness; (2) support decision making; (3) monitoring actions; (4) giving alerts and requests for urgent actions; (5) improving coordination and cooperation among ATCOs. All these activities are tightly connected with the usage of the S&D patterns listed above, as reported in the following table.

Table 5 - Matching S&D patterns to prototype functionalities

Feature	SP4	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8	DP9	DP10
Situation Awareness	-	-	X	X	X	X	X	X	X	X	X
Decision Support	-	-	-	X	X	X	-	-	-	X	X
Action Monitoring	X	X	-	-	X	-	-	X	-	-	-
Request for Action	-	X	X	-	-	-	X	-	-	-	-
Coordination	X	-	-	-	-	-	-	X		X	X

The S&D patterns are used for: (1) enabling proactive functionalities for managing S&D properties provided by ACT; (2) ensuring the usage of already developed and fully validated S&D solutions. Thus ACT supports controllers activities described above by implementing the following functionalities:

- (F1) Reminder - ACT reminds the controllers the Internal Permanent Instructions and any other information related to the daily situation;
- (F2) Communicator - ACT supports communication between controllers, i.e., discovery of active communication channels, delivery of messages etc. ;
- (F3) Recorder the ACT records the commands issued by controllers in case of multiple possibilities and records all the controller inputs and actions in case of deviation from the existing rules (for statistical, reporting and auditing purposes);

- (F4) Advisor: the ACT supports controllers in establishing alternative action plans in order to deal with specific situations. Also the ATM daily activities enhanced by ACT and the four ACT s main functionalities are connected, as shown in the following table.

Table 6 - Matching prototype features to controllers activities

Feature	F1	F2	F3	F4
Situation Awareness	X	X	-	X
Decision Support	X	-	-	X
Action Monitoring	-	-	X	-
Request for Action	-	X	-	X
Coordination	X	-	-	-

We will report some practical examples about the usage of organizational patterns in the ATM Scenario, with respect to the relevant activities identified above and the specific ACT functionalities that support and improve them.

Situation Awareness. In ATM, the situation awareness of ATCOs is already widely implemented through a rich activity context full of technology , e.g., by means of electronic strip bay, radar display, weather forecast display. Notwithstanding ACT improves existing technologies, by using in particular two patterns: Multiple Representations of Information and Public Artefact. The ACT GUI and the Reminder and Communicator functionalities are designed on the basis of these patterns.

> **Example 12** the ACT User Interface is different for the different roles (Executive, Planner or Supervisor) by highlighting different information in a different manner.
>
> **Example 13** ACT let communicate different actors and permits to share common information.

Monitoring Action. One can envision ACT as a watchful tool during critical performances or unexpected solutions. By using the RECORDER Functionality, ATCOs can monitor and record actions and decisions taken in risky situations in order to report them to the qualified authority. Related patterns are: Artifact as an Audit Trail and Evolution of Procedures.

> **Example 14** Some months later a similar situation is presented to another control team in a different ACC. The Supervisor checks in the System if there is a further procedure to manage safely the high traffic and the System propose the Delegation of Airspace already adopted, recorded by the tool and sent to the competent authority for becoming a new procedure.

Request for Action. ACT is a proactive tool able to promptly request for urgent and pressing activities to be performed by means of the REMINDER and ADVISOR functionalities. Patterns involved are, in particular: Public Artefact and Multiple Representations of Information.

> **Example 15** The PLC Paula notices on the ACT display that notwithstanding the re–sectorisation there is still a report for an increase of traffic exceeding the capacity of Sector SU 1.

Coordination (Cooperation Support). ACT, by its Communicator Functionality, can be used as a non intrusive dialogue tool, enhancing communication activity among ATCOs that permits to notify news or requests, ask for help and assistance, exchange information, interactively share and modify documents. Patterns involved are: Collaboration in Small Groups, Public Artefact (both in Example 12) , Assistance trough Experience (Example 13).

> **Example 16** Robert and Mary (the Supervisors of the 2 ACCs) consider the possibility to perform a vertical re-sectorisation of Sector SU 1 extending the upper limit of Sector N from FL 280 to unlimited.

Example 17 The PLC Paula notifies the expected increase of traffic to the SUP Robert, by communicating the amount of expected traffic. Paula points out the situation to Robert asking for a solution.

In this section we showed how S&D organizational patterns account for ATM organizational and procedural aspects, thus enabling the definition and implementation of software functionalities that support the ATM activity and enhance the safety and the overall resilience of the System (Di Giacomo et al. 2008).

RELATED WORKS AND CONCLUDING REMARKS

Many works have been proposed for modeling and evaluating dependable mission critical systems (SQUALE Consortium 1999, Stamatelatos et al. 2002, Van Lamsweerde et al. 2003). The most appropriate type of model framework depends on the complexity of system behavior and the S&D requirements to be taken into account.

State-space based models include Markov models and high level approaches which have an underlying Markov model. These models provide great expressiveness for dependability specifications and have been extensively applied to model the dependability and to analyze the hardware software reliability (Zhang et al. 2005). This class of models includes queuing networks and Generalized Stochastic Petri Nets (Betous-Almeida et al. 2002, Fota et al. 1998), which have been the most commonly used. Recently, many frameworks based on UML (Gabor et al. 2000) have been proposed. Generally, they propose to embed Petri Nets for describing the system properties and validating the solutions.

In this article, we have presented how to capture the knowledge of S&D experts using the ***SI**** framework, how to elicit organizational S&D patterns and how to deploy them into different scenarios. We present in this section the concluding remarks expressed by the prototype developers who used the S&D patterns to build real systems. The usage of the organizational patterns varies according to the infrastructure deployed by the organization.

The smart items scenario is composed of a combination of three publicly available infrastructures namely WIFI, GSM, and GPRS communication networks. This results in a hybrid communication architecture that meets the high reliability needs strongly required by our e-health case study. Patients' e-health remote assistance is accomplished by means of a set of workflows properly defined in, stored, and maintained by the MERC. The MERC comprises of hybrid infrastructure made four types of infrastructures, namely, SOA web services, human resource, Data Protection Directive (Directive 95/46/EC) and the sensors and actuators in the Smart House.

In the ATM scenario, we have investigated how S&D patterns can inform at runtime the reaction to threats or attacks. In particular, we are concerned about how, during the unfolding of unexampled threats or attacks, S&D patterns could enable an ad-hoc reaction. In this context, S&D patterns provide models for reaction processes because: (1) patterns represent the portion of the reaction process that is relevant to S&D features; (2) reaction processes adjust their behavior in order to reflect changes into S&D patterns;(3) S&D patterns specify what parts of the processes can be automatically interpreted and those parts that are left to interpretation to users. The prototype relies on the runtime framework (Serenity Consortium 2008a), which manages the matching between a reaction plan, its execution and the S&D patterns. The actors participating in

the reaction process exploits, directly via the run-time framework console or indirectly via the interface or logic of the application, the knowledge formalized by S&D patterns.

REFERENCES

Anderson, R. (1993). *Why cryptosystems fail*. In Proceedings of the 1st ACM Conference on Computer and Communications Security

Anderson, R. (2001). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley.

Asnar, Y., Bonato, R., Giorgini, P., Massacci, F., Meduri, V., Riccucci, C., & Saidane, A. (2007). *Secure and Dependable Patterns in Organizations: An Empirical Approach*. In Proc. of RE'07. IEEE Press.

Asnar, Y., Moretti, R., Sebastianis, M., & Zannone, N. (2008). *Risk as Dependability Metrics for the Evaluation of Business Solutions: A Model-driven Approac*h. In Proc. of DAWAM'08.

Betous-Almeida, C., & Kanoun, K. (2002). *Stepwise construction and refinement of dependability models*. In In Proceedings of DSN.

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

Busnel, P., El Khoury, P., Li, K., Saidane, A., & Zannone, N. (2008). *S&D Pattern Deployment at Organizational Level: A Prototype for Remote Healthcare System*. In Proc. of STM'08.

Compagna, L., Khoury, P. E., Krausov´a, A., Massacci, F., & Zannone, N. (2009). *How to Integrate Legal Requirements into A Requirements Engineering Methodology for the Development of Security and Privacy Patterns*. Artificial Intelligence and Law, 17(1), 1–30.

SQUALE Consortium, (1999). *SQUALE: Security, Safety and Quality Evaluation for Dependable Systems.* Tech. rep., SQUALE Consortium.

Dardenne, van Lamsweerde, & Fickas] Dardenne, A., van Lamsweerde, A., & Fickas, S. (1993). *Goal-directed Requirements Acquisition*. Sci. of Comp. Prog., 20, 3–50.

Dempster, A. (2008). The dempster-shafer calculus for statisticians. International Journal of Approximate Reasoning, 48(2), 365–377.

Di Giacomo, Felici, Meduri, Presenza, Riccucci, & Tedeschi] Di Giacomo, V., Felici, M., Meduri, V., Presenza, D., Riccucci, C., & Tedeschi, A. (2008). *Using Security and Dependability Patterns for Reaction Processes.* In Proc. of DEXA'08, (pp. 315–319). IEEE Press.

Elahi, G., & Yu, E. (2007). *A goal oriented approach for modeling and analyzing security trade-offs*. In Proc. of ER'07, LNCS 4801, (pp. 375–390). Springer.

Fota, N., Kaaniche, M., & Kanoun, K. (1998). *Dependability Evaluation of an Air Traffic Control Computing System*. In In Proceedings of the 3rd IEEE IPDS.

Gabor, H., & Istvin, M. (2000). *Quantitative Analysis of Dependability Critical Systems Based on UML Statechart Models*. In In 5th IEEE International Symposim on HASE.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Giorgini, P., Massacci, F., Mylopoulos, J., & Zannone, N. (2005a). Modeling Security Requirements Through Ownership, Permission and Delegation. In Proc. of RE'05, (pp. 167–176). IEEE Press. Full version in *International Journal of Information Security* 5(4): 257-274 (2006).

Giorgini, P., Massacci, F., & Zannone, N. (2005b). Security and Trust Requirements Engineering. In FOSAD 2004/2005, LNCS 3655, (pp. 237–272). Springer-Verlag.

Kolmogorov, A. N. (1956). Foundations of the Theory of Probability. Chelsea Publishing Company, 2 ed.

Liu, L., Yu, E. S. K., &Mylopoulos, J. (2003). S*ecurity and Privacy Requirements Analysis within a Social Setting*. In Proc. of RE'03, (pp. 151– 161). IEEE Press.

Massacci, F., & Zannone, N. (2008). *A Model-Driven Approach for the Specification and Analysis of Access Control Policies.* In Proceedings of Information Security'08, vol. 5332 of LNCS, (pp. 1087–1103). Springer-Verlag.

Schumacher, M. (2003). *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. Springer-Verlag.

Serenity Consortium (2008a). A7.d4.2 - scenario s&d solutions. Tech. rep., ww.serenity-project.org.

Serenity Consortium (2008b). *A6.D3.2 - specification of serenity architectur*e. Tech. rep., www.serenity-project.org.

Serenity Consortium (2009). *The final set of S&D patterns at organizational level*. Tech., www.serenity-project.org.

Stamatelatos, M., Vesely, W., Dugan, J., Fragola, J., Minarick, J., & Railsback, J. (2002). *Fault Tree Handbook with Aerospace Applications*. Tech. rep., NASA.

Van Lamsweerde, A., Brohez, S., Landtsheer, R. D., & Janssens, D. (2003). *From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering*. In In Proc. of RHAS.

Yoder, J., & Barcalow, J. (1997). *Architectural Patterns for Enabling Application Security*. In Proc. of PLoP'97.

Zhang, Z., Shen, H., Defago, X., & Sang, Y. (2005). *A Brief Comparative Study on Analytical Models of Computer System Dependability and Security*. In Proceedings of the 6th Int. Conf. on PDCAT.