2016 IEEE 24th International Requirements Engineering Conference Workshops

Towards security modeling of e-voting systems

Cristiano De Faveri, Ana Moreira, João Araújo and Vasco Amaral

NOVA LINCS, Department of Computer Science
Faculty of Science and Technology, Universidade NOVA de Lisboa
c.faveri@campus.fct.unl.pt, {amm,joao.araujo,vmm}@fct.unl.pt

***Abstract*—As voting systems evolve from paper ballots to electronic voting (E-voting) applications, we have noticed significant efforts to develop real-world securer solutions. E-voting systems are security-critical systems that require early identification of security requirements and controls based on the analyses of potential vulnerabilities, threats, attacks, and associated risks. General purpose modeling languages and current tool support to model security concerns exist. However, they lack a comprehensive solution that includes tool support for verification of security goal completeness and risk analysis in specific domains. Also, communication between stakeholders in large-scale systems is difficult, specially because security is not the core skill of many requirements engineers. To overcome these challenges in the electronic voting domain, we developed EVSec, a domain-specific visual modeling language. EVSec is process-centric language and allows modelers expressing activities and social interactions, while identifying security concerns with associated risks. Comprehensive tool support provides security goals completeness and assists users on the identification of critical parts of the model with higher security risks. We used EVSec to model the Brazilian national election, demonstrating its adequacy.**

I. Introduction

As voting systems evolve from paper ballots to Internet-based applications, efforts to develop real-world solutions have helped identify several challenges [1] in this domain, such as those posed by security and usability. Poorly designed voting systems can jeopardize an entire election, and history shows that results may lead, at best, to local confusion, and, at worst, to violence and even civil war [2], [3]. Potential benefits of electronic voting (e-voting) are the suppression of unwanted human errors and the reduction of costs of the tallying process [4], [5]. Democratic elections using e-voting solutions, in countries such as Norway, Estonia, Spain, Belgium and Brazil, have shown that the winning margins of a candidate could be inferior to the error margins of the voting systems themselves, what makes elections an activity worth automating [1]. The fact that electronic voting processes can open space for electoral fraud, particularly with the significant associative risks of remote voting, makes e-voting systems security-critical [6]. When networked, these applications are vulnerable to fraud, compromised computers, DoS attacks, and other threats typical from such heterogeneous environments.

Although remote ballots exempt the voter to travel to a poll site, coercion can become a major concern, specially in countries with unstable political situations. Thus, authenticity, privacy and verifiability are key security issues that should be assured to guarantee vote legitimacy, while allowing transparency in every step of the voting process. Despite advances in e-voting technologies, satisfying every single election goal and requirement still remains challenging [7]. Moreover, security-transparency dichotomy demands mechanisms to improve communication between different levels of stakeholders. On the one hand, security mechanisms have to be identified and implemented throughout the election process. On the other hand, every activity within the process must be transparent to anyone interested in knowing how the system works. Security-critical systems require the early identification of security requirements [8], [9], as well as tool support for reasoning about vulnerabilities, threats and countermeasures.

Many security modeling advances exist, including approaches based on UML [10]–[15], goal-oriented methodologies [8], [9], [16], [17] and process-centric models [18], [19]. Many of these focus either on early security requirements engineering or else on the design time requirements modeling (mapping security requirements to system components). However, all these fail in assisting engineers with a comprehensive language and toolset that address domain-specific security concerns, such as those found in e-voting systems.

We created EVSec, a domain-specific visual language equipped with security-related elements to be employed during the design of a voting process. EVSec's supporting environment enables analysis of the models and prompts the modeler with typical security requirements that may have been forgotten along the process. EVSec is a process-centric language that allows expressing activities and social interaction between actors participating in elections, while identifying security issues and keeping traceability between security goals, requirements, threats, vulnerabilities, countermeasures and associated risks. Besides, EVSec offers mechanisms to identify alternatives among security controls and to document the rational behind their choices. This introduces opportunities to promote communication among different stakeholders, as for example when justifying the investment on a particular security control. EVSec also provides risk analysis support by allowing identification of threat likelihood and consequences. Tool support identifies critical parts of the model with higher risks, and verifies security goal completeness, considering particularities of e-voting domain. We used EVSec to model the Brazilian national election process, demonstrating the feasibility of the language in both modeling security concerns

978-1-5090-3694-3/16 $31.00 © 2016 IEEE
DOI 10.1109/REW.2016.37

and performing risk analysis.

The remainder of this paper is organized as follows: Section II offers an overview of the election processes and security requirements of electronic voting. Section III presents the main voting concerns (functional and non-functional) as the result of a domain analysis. Section IV describes the main characteristics of EVSec, and Section V specifies the Brazilian national election scheme. Finally, Section VI discusses the related work, and Section VII draws conclusions and points directions for future work.

II. Background

A. Election Process

In a democracy, a voting system represents a method by which voters make a choice between well-defined options, often as part of an election or a policy referendum. Such systems aim at finding out the winner or winners of an election, by determining how individuals express their vote and how votes are tallied. Electronic voting can support multiple forms of election under distinct regulations, and make the voting, tallying and counting processes much faster, when compared with a manual process [5]. Electronic voting systems can be one of two types [20]: physical supervision, and remote e-voting. Supervised e-voting requires authorities or governmental representatives to manage and supervise the voting in person. In general, supervised e-voting is supported by voting machines, also known as direct-recording electronic (DRE) voting machines [4]. DREs keep information about candidates and record votes from the voters. From the voters' point of view, while DREs demand physical polling locations, remote e-voting is performed within the voters' sole influence and without any authorities' supervision. Internet voting is an example of remote e-voting, with voters using any Internet-enabled device and proper software to cast their ballots. So, Internet voting is more flexible, allowing voters to vote even in extreme cases, such as astronauts living and working on board of spacial stations[1].

Critical elections (e.g., those to choose public officers) require a voting schema that satisfies certain general level of security and also particular requirements imposed by the nature of voting systems. Organizing an election depends mostly on the amplitude of the balloting, the applied technology and how the demands and constraints will be addressed in the system. In general, an election process is comprised of four phases : *(i)* setting up the election, *(ii)* running the election, *(iii)* tallying the votes, and *(iv)* disclosing the results. Setting up an election involves establishing protocols, security parameters and needed arrangements to constitute parties, roles, candidates and the suffrage. The running phase aims at collecting and recording the votes within a defined time-frame. Once voting is concluded and votes are collected, tallying phase starts. As a result, winners and the classification of each candidate are made public. All these phases should be transparent for anyone wanting to ensure correctness of tallying, which makes privacy and verifiability one of the major challenges when designing e-voting systems [21].

[1]http://www.nasa.gov/mission_pages/station/expeditions/expedition18/vote.html

B. E-voting Security Requirements

Designing and implementing e-voting systems requires taking into account a set of general and domain-specific security requirements. General security requirements, as found in every networked system, are non-functional requirements (NFRs) related to availability, integrity, confidentiality, and robustness [22]. However, specific e-voting requirements represent particular aspects of the domain such as fairness, abstention, and incoercibility. As these requirements frequently conflict, trade-off analysis is often required during the system design. Additionally, completeness is a key aspect of security-critical systems. So, engineers should be aware of potential missing security requirements during the modeling phase.

Based on the existing literature, we compiled a list of security requirements and solutions for common issues in e-voting systems [1], [5], [23]–[25]: *(i)* **Eligibility and authentication:** only eligible voters are allowed to vote. Valid voters should be identified during the set-up phase of the election. The eligibility requirement demands some security mechanism, such as authentication, to guarantee that only authorized voters will cast ballots; *(ii)* **Uniqueness:** eligible voters are allowed to vote only once in a certain candidate in the election. Although this requirement aims at guaranteeing a candidate does not have more than one vote from a particular voter, some decision design can relax this constraint in favor of non-coercion as shown ahead; *(iii)* **Privacy:** everyone can know the voters and the candidates, but only the voters can know their votes. The vote of a voter must remain secret. Thus, a voter should not be provided with a receipt that prove she voted on a given candidate [26]; *(iv)* **Availability:** an eligible voter must always succeed to cast a ballot during the election time-frame; *(v)* **Integrity:** a vote must not be removed or changed after recorded/saved in the ballot box; *(vi)* **Accuracy:** recording and counting votes must be error-free. Invalid voters must be discarded during the tallying process; *(vii)* **Verifiability and Accountability:** any step of the voting process can be inspected by any independent entity to prove a vote was counted correctly. Satisfying this universal verifiability property against other requirements frequently makes design of e-voting solutions more complex [24]. All procedures that change the state of the system must be identifiable, together with the responsible for that change. This includes every administrative procedure performed by authorities; *(viii)* **Abstention:** blank and null votes need to be considered as valid votes, and cannot be replaced by any other vote; *(ix)* **Fairness:** no one should be able to compute counting partially, to keep the election impartial until its end; *(x)* **Robustness:** the voting system must be robust against failures and attacks. The system should be resilient against corruption by authorities and voters (passive attacks), tolerant to hardware failure, and external attacks; *(xi)* **Incoercibility:** a voter should not be coerced to manipulate the way a vote is cast. An adversary may force a voter to abstain from casting a vote, or may even offer rewards in return for his vote. Also, an adversary could represent another eligible voter at any stage of the voting scheme, by obtaining the voter's information that should guarantee his privacy. However, a voting scheme may relax uniqueness property to allow voters to overwrite their vote in favor of non-coercion.

EvSec covers the above discussed security requirements. Other NFRs, such as scalability, flexibility, learnability and cost, can contribute positively or negatively to security requirements. Despite the importance of these (and other) NFRs, this paper does not cover trade-off analysis between them and security.

III. Domain Analysis for E-voting

We performed a domain analysis to identify the main voting concerns (functional and non-functional), focusing particularly on security modeling. The domain analysis process identifies, captures and organizes the fundamental characteristics (common and variable across products) aiming at supporting reuse when creating new software products of the same family [27]. It is part of the domain engineering process, which also encompasses domain design and domain implementation.

A. Domain Concerns

A comprehensive understanding of the concepts and rules that govern the voting process is necessary to model different e-voting systems. Also, the environment and the technology involved on e-voting operations may require different security requirements and controls. For example, a typical e-voting election process, using physical supervised model, starts with the identification of the voter in the poll place. Depending on the election's setup, this could be made directly by the voter or by some authority that checks visually the credentials of the voter. Once the voter is eligible and authorized to vote, the ballot box is enabled to cast the vote. Remote e-voting systems, on the other hand, are required to implement some voter identification mechanism, which poses specific security controls to avoid impersonation.

The (functional) concerns identified are captured and structured in the feature model [28] depicted in Figure 1. Although we could have chosen a different model to express the taxonomy of the domain (a class diagram, for example), a feature model represents well the abstract concepts and offers an overview of the variabilities of the domain.

The first level of features ($F1$ to $F7$) collects the top level of concerns that range from the execution mode of the election, to the various types of people involved in an election, to the various phases of the whole process, to the organization the election respects, to the votes themselves, and, finally, to the poll place. The execution mode can be physically supervised, depending on specific places for voting, or managed remotely, as observed on internet-based voting systems. Relevant roles that are part of elections comprehend authorities ($F2.3$), staff ($F2.2$), valid voter ($F2.1$), valid candidate ($F2.4$). The later may have sub-candidates such as vice-president, for example, in which voting for a president automatically implies voting on associated vice-president. Ballot boxes ($F7$) can be distinguished by different technologies, such as DREs ($F7.3$), mobile devices ($F7.1$) or PC-like terminals ($F7.2$). DREs may have different features for accessibility ($F7.3.1$), input and output modes ($F7.3.2$ and $F7.3.3$) and security perimeter criteria ($F7.3.4$ and $F7.3.5$). Elections have well-defined phases as explicitly described on features $F3$. Notably, tallying and publishing features can be performed post-election or real-time. In general, elections are composed by organizational entities in which candidates are grouped in parties ($F4.2$) and regional agencies that manage the election. Other organizations can be identified, depending on how the election is structured. The feature vote ($F5$) represents the kinds of vote (e.g., null, absence justification, concrete, blank). Finally, poll place ($F6$) refers to locations where voting occurs or how voters and candidates are separated in zones, sections or virtual places. For example, the same national election may elect senate candidates and also regional candidates.

B. Security Analysis

Security is a broad area which encompasses methods and techniques to protect valuable assets. The main goals of computer security are confidentiality, integrity and availability (a.k.a. CIA triad) [29]. However, accountability and non-repudiation are also security goals representing the identification of responsibility, proof of integrity and origin of data. We refer to security concepts captured from [29]–[34], ISO 27002 (Information technology — Security techniques — Code of practice for information security management) and the Internet Security Glossary — RFC 2828[2] as described next.

Valuable assets are system resources that users and owners wish to protect. They are categorized into hardware, software, data, communication facilities, and networks. Assets can be abused and the damaging entities are called attackers. Attackers (or adversaries) can be insiders or outsiders to the organization owning the assets. Insider attackers are agents who initiate attacks within the security perimeter of the organization. Outsiders are those who initiate attacks outside the security perimeter. To perpetrate an attack, adversaries must know some vulnerability in a system. A vulnerability is a weakness in a system's design, implementation, or operation and management. It can be categorized into process, physical, natural, social and technical. A threat is a potential situation for violation of security, i.e., a possible danger that might exploit a vulnerability. Threats can be classified under different dimensions, as human, technical or environment threats, such as temperature and humidity, fire and smoke. Attacks are characterized by a threat action carried out to cause some undesirable violation of security. Attacks can be active or passive in the sense that they can provoke system stimuli to exploit some vulnerability or they can simply eavesdropping transmissions to learn and make use of information. Attacks

[2]http://www.ietf.org/rfc/rfc2828.txt

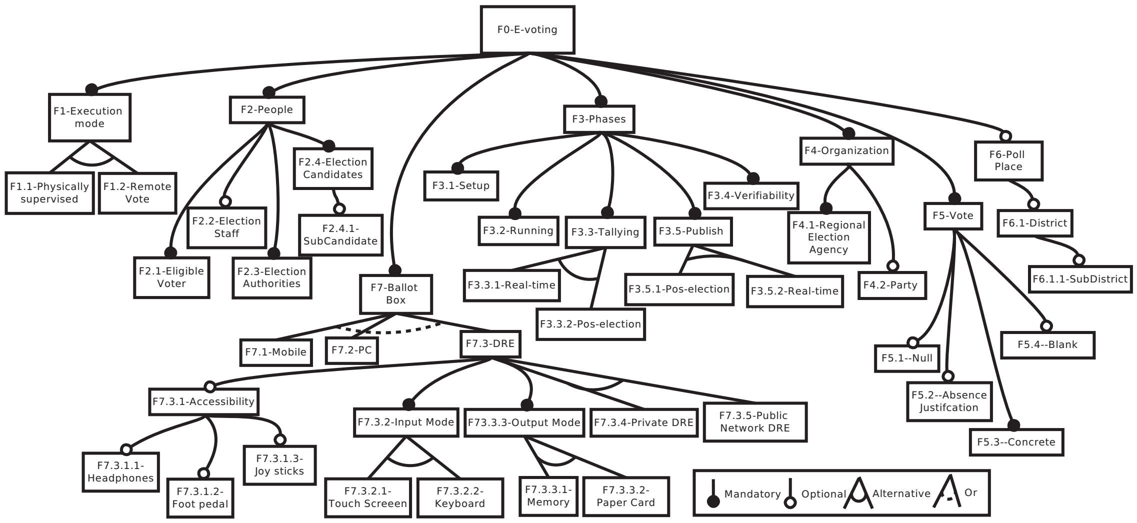


Fig. 1: E-voting feature model

have different purposes that vary from reconnaissance, denying a service, or getting access to unauthorized privileges.

To mitigate the harm caused by adversaries, countermeasures are necessary. A countermeasure represents an action, procedure, or technique that discovers and reports threats, vulnerabilities or attacks so that corrective actions can be taken. Countermeasures may be preventive, detective or corrective. Preventive countermeasures aim at avoiding any damage to a system by implementing proper security mechanisms. When attacks take place, detective mechanisms try to identify harmful actions and adversaries to halt the offensive. Last, corrective countermeasures occur to recover the system and minimize harm extension when previous defense mechanisms fail. Multiple countermeasures may be required to fulfill a security requirement. For example, authentication may be a security requirement that needs to be implemented by multi-level login mechanisms, such as integrated user/password and biometrics. Risks associated to unwanted incidents may be calculated as a probabilistic assessment of loss. These are the security concepts that our domain-specific modeling language (DSML) currently supports.

IV. EVSec Modeling Language

A. Language Overview

EVSec is a DSML designed to model e-voting processes and related security issues along with their potential countermeasures and associated risks. EVSec's main users are requirements engineers who need to model e-voting systems. EVSec offers improved expressiveness of the voting domain which could facilitate the communication between the election authorities during the early stages (planning) of the process modeling. Also, EVSec offers security syntax and semantics models, allowing to express both the defender and attacker's point of view. The primary objectives of EVSec are: *(i)* **Capturing social and technical requirements of e-voting systems** so that whenever an interaction between human and organizations may raise security issues that technologies are supposed to attenuate or circumvent. *(ii)* **Identifying security requirements early** in the development phases so that e-voting security concerns can be addressed with tested, recurrent solutions, not depending on a deep analysis of threats and vulnerabilities. On the other hand, some processes may require critical scrutiny to identify vulnerabilities, threats, and attacks that may potentially jeopardize the assets' integrity. EVSec's flexibility allows assigning security controls to the assets, or refining into potential attack vectors until reaching mechanisms that mitigate them. *(iii)* **Supporting risk analysis** requires information about the existence and severity of vulnerabilities and threats. This information can be provided by experts who reason about critical assets in the system. Thus, designers can decide where to put their effort to reinforce security policies and controls. *(iv)* **Integrating software engineering and security**, by offering a structured model for security to assist requirements engineers to settle security issues, at least in the initial modeling phases. This does not replace security experts, as only they know how the system can be threaten, who are the potential offenders, and how these are combined in the environment where the system will be deployed. *(v)* **Improving communication with stakeholders**, by providing resources to facilitate communication between election authorities and requirements engineers in such way that models of e-voting processes can be understood with minimum training effort.

B. EVSec Metamodel

The abstract syntax of EVSec is represented by its metamodel and describes the concepts of the domain that are relevant for the language. Figure 2 presents a concise version of EVSec metamodel, representing process entities (yellow), e-voting entities (blue), and security entities (purple). The central class of the model is *EletronicElectionModel*, which represents the aggregation of top-level classes in the model. The class *Node* is an abstraction of entities that allows association (links) relationships. Specialized from the entity *Node*, the class *NamedNode* represents entities that can have a name in the model. The class *AssetNode* represent entities that can be flagged as an asset in the system. In EvSec, assets can be classified into hardware, software, data, communication, and none.

Elements of the voting domain are represented by people (class *Person* and its respective subclasses *Staff*, *Voter*, and *Candidate*), ballot boxes (class *BallotBox*), a vote (class *Vote*), a party (class *Party*), a vote processor (class *VoteProcessor*) and a dashboard (class *DashBoard*). Votes can be set to represent a blank vote, a concrete vote or a null vote. Vote processor represents a device that can perform the tallying of votes. The dashboard is the entity responsible for publishing the outcome of the election.

A process is expressed by activities and transitions between activities. An activity (class *Activity*) can be signed as internal or external. Internal activities are those under responsibility of the system being developed. On the other hand, external activities are those outside the system, i.e, the system does not have any responsibility about its execution. Activities can also be defined as top-level activities that can be decomposed in smaller activities during the modeling process. Subactivity (class *SubActivity*) connects two activities in the same diagram or in different diagrams. The *Initial* and *Final* classes represent the beginning and the end of a process or sub-process. The *Story* class represents entities connecting any element to an external use case, use story or any requirement description that can give the user more details about the requirement. The class *Document* is a generic element that represents any data required or produced by an activity. Activities can be grouped by using entities of the class *Entity*. Activities within a group are supposed to be executed in parallel. The elements of the diagram can be organized in partitions (class *Partition*), similarly to swimlanes used in UML activity diagrams.

EVSec provides a set of links that allow engineers to specify the relationship between entities and how the information flows between the activities and agents (people and machines). Inspired by the concepts of IDEF0 function modeling methodology [35], activities are connected to other entities using four main links: input, output, mechanism and control (classes *Require*, *Produce*, *Mechanism* and *Control* respectively). *Require* means that a source entity requires the target entity to execute an activity. *Produce* means that a source entity produces the target element during an activity. *Mechanism* and *Control* represents a resource that is required to execute an activity and the constraints for a certain activity, respectively. *Connection* is used to connect eligible entities to security entities. *Delegate* is between a person and an activity that is delegated to be executed by another person. Class *Flow* represents the flow of execution between activities due to an event and/or a condition. Additionally, class *Mitigation* is used to indicate a security requirement or control mitigating a security threat or vulnerability.

Security concerns are mapped into six elements: attack, attacker, vulnerability, threat, security requirement, and security control (classes *Attack*, *Attacker*, *Vulnerability*, *Threat*, *SecurityRequirement*, and *SecurityControl*, respectively). Risks may be assigned to vulnerabilities, threats and attacks, by associating a likelihood within a discrete list (e.g., rare, unlikely, possible, likely and almost certain) and a severity. Severity ranges from insignificant to doomsday level, and is used to compose the overall risk of a potential offensive. Security requirements are assigned to security goals, which encompass general security goals (confidentiality, integrity, availability, accountability and non-repudiation) and security-specific goals (eligibility, uniqueness, privacy, verifiability, accuracy, long-term privacy, fairness, robustness, receipt-freeness, incoercibility). Security goals are used for completeness analysis purposes. Also, security controls offer descriptions indicating the pros and cons of choosing them. These controls can be flagged as "best choice", "denied" or "for further discussion" indicating in the model which security controls have been selected or discarded. This provides valuable information for different stakeholders, justifying the reasons why a set of controls was elected.

C. EVSec Notation

EVSec uses meaningful symbols to represent security and e-voting concepts, facilitating stakeholders' understanding. To model process activities, we used simple shapes (e.g., rectangle, oval, square) much like those found in UML activity diagrams. For the e-voting and security domains, a set of icons was proposed to represent each domain entity. Figure 3 presents the most relevant elements used by EVSec.

Given the voting and security concepts identified, we designed the EVSec concrete syntax by trying to balance expressiveness and simplicity while following the Moody's physics of notations (PON) principles [36]. PON comprises a prescriptive theory that includes a set of principles for designing cognitively effective visual notations, namely: semiotic clarity, perceptual discriminability, semantic transparency, complexity management, cognitive integration, visual expressiveness, and graphic economy.

Taking the PON principles, we discuss the design decisions of EVSec visual notation. We categorize our notation within five groups, namely: people, vote, security, links and miscellaneous. In the group people, we used three different icons to symbolize a candidate (more formal garment), a voter (with informal garment) and a staff (a person using a badge). Votes are represented by an icon with different colors, being red for null votes, white for black votes and green for concrete votes. In the security group, we use a bomb to represent an attack, the image that resembles a thief to represent an attacker, a separated link of a chain to represent a vulnerability, a biohazard-style icon to represent threats, a shield as security requirements and a padlock to represent a security control. Security control flagged as "best choice" is represented by padlocks with a green check symbol. In the

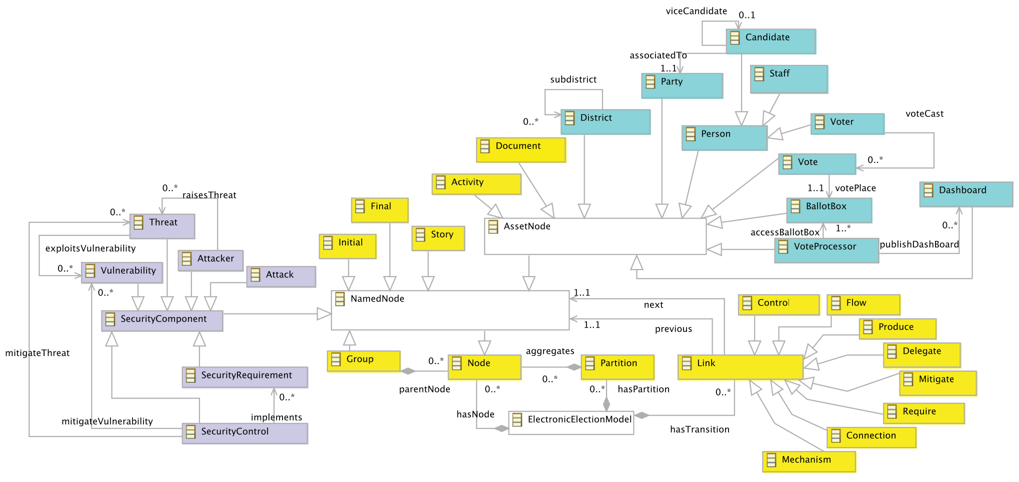


Fig. 2: EvSec metamodel

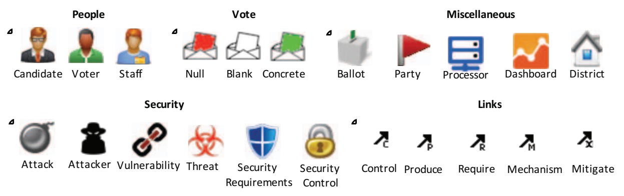


Fig. 3: EvSec partial notation

group miscellaneous, the concept of a ballot is represented by a box with a vote. Parties are represented by a flag, a processor by computers connected in a network, dashboard as a graphic and districts are represented by the symbol of a house. Links are encoding with text to reinforce and expand the meaning of the symbol. We decided to minimize the use of multiple symbol colors to avoid variations that may be hard to grasp by modelers. As a consequence, modelers depend on tool support to inspect an element to check its properties. Such trade-off is an attempt to alleviate the accidental complexity problem that critically affects visual modeling languages [37], [38].

D. *Tool Support*

The construction of EVSec models is supported by an Eclipse-based tool. This tool facilitates modeling different phases of e-voting systems, checking models for security completeness and computing security risks associated with each activity. Engineers can design processes by starting with high-level activities, and refine them to more concrete activities that can be specified with different models, offering multiple views to distinct stakeholders. Because EVSec is built on top of EMF technologies, integration is accomplished with minimum effort, with a number of tools and frameworks available in the Eclipse community. Completeness is supported by verifying the model security goals against the pre-defined list that corresponds to the domain analysis security requirements. Thus, anytime a modeler requests security completeness checking, the tool analyses the model instance looking for missing security goals. The tool can also show the list of security requirements that address a given security goal.

Risk analysis, as explained in previous section, shows the parts of the model that are more security-critical. The risk level is computed by the combination of the impact that would result from the threat event with the likelihood of the event occurring. We map qualitative values of impact and threat likelihood to a quantitative scale following the well-established guide for conducting risk assessments in information technology systems provided by the National Institute of Standards and Technology (NIST 800-30) [39]. Although EVSec qualitative risk scale differs from the terms used by the NIST 800-30 guide, the correspondence is straightforward as presented in Tables I and II. We chose these EVSec qualitative terms because they are closely related to the semantics of each level. Corresponding qualitative scales, however, denote default values and can be adjusted by modifying tool parameters, according to particular needs.

TABLE I: Correspondence between EVSec and NIST 800-30 threat likelihood assessment

EQLS	NQLS	QTS	EVSec semantics
Rare	Very Low	0.04	Adversary is highly unlikely to initiate the threat event
Unlikely	Low	0.2	Adversary is unlikely to initiate the threat event
Possible	Moderate	0.79	Adversary is somewhat likely to initiate the treat event
Likely	High	0.95	Adversary is highly likely to initiate the threat event
Almost Certain	Very High	1	Adversary is almost certain to initiate the threat event

EQLS: EVSec Qualitative scale; NQLS: NIST Qualitative scale; QTS: Common quantitative scale

TABLE II: Correspondence between EVSec and NIST 800-30 impact assessment

EQLS	NQLS	QTS	EVSec semantics
Insignificant	Very Low	0.04	Negligible adverse effect based on a minor security breach in a single area
Minor	Low	0.2	Limited adverse effect based on a security breach in one or two areas
Moderate	Moderate	0.79	Serious adverse effect based on limited systemic (and possibly ongoing) security breaches
Major	High	0.95	Severe or catastrophic adverse effect based on ongoing systemic security breach
Catastrophic	Very High	0.99	Multiple severe or catastrophic adverse effects based on major systemic security breach
Doom day	-	1	Multiple severe or catastrophic adverse effects based on multiple instances of major systemic security breaches

EQLS: EVSec Qualitative scale; NQLS: NIST Qualitative scale; QTS: Common quantitative scale

The overall risk level is identified by using values described in Table III. We use the same scale provided by NIST 800-30, in which very-low risks (blue) are those with risk factor within 0.00 and 0.04. Low-level risks (green) are considered between 0.05 and 0.20, moderate-level risks (yellow) are those within 0.21 and 0.79, high-level risks (orange) are those within 0.80 and 0.95, and finally, very high-level risks (red) are those between $0,96$ and 1.00. We compute risks by analyzing individual security elements (threat, vulnerability and attack) and by combining them according to their links in the model. For example, a threat T can be assigned with a particular likelihood and severity, which will result in a risk level. This threat can exploit a vulnerability V, which is assigned with a likelihood and severity. Together, the overall risk of having the threat T exploiting the vulnerability V is calculated by the mean of the likelihood and severity, following this formula:

$$R(X) = \frac{1}{n}(\sum_{i=1}^{n} L(X_i) \times S(X_i)) \tag{1}$$

where R is the overall risk over a set X of security elements in the model. $L(X)$ and $S(X)$ are the quantitative scale of likelihood and severity assigned to a security element X_i, respectively. The number of security elements are represented by n.

TABLE III: Risk-Level Matrix, extended by [39]

Threat likelihood	Impact					
	A	B	C	D	E	F
Rare	0,0016	0,008	0,0316	0,038	0,0396	0,04
Unlikely	0,008	0,04	0,158	0,19	0,198	0,2
Possible	0,0316	0,158	0,6241	0,7505	0,7821	0,79
Likely	0,038	0,19	0,7505	0,9025	0,9405	0,95
Almost Certain	0,04	0,2	0,79	0,95	0,99	1

Impact: A-Insignificant, B-Minor, C-Moderate, D-Major, E-Catastrofic, F-Doom day

V. Case Study

We conducted a case study to model the national Brazilian election process using EvSec. Brazil is one of the biggest democracies in the world with more than 135 million electors[3] who cast their votes electronically since 1996, when DREs were first introduced. To identify Brazilian election goals and requirements, we refer to a set of public documents containing norms, legislation, and official reports. These documents can be found at the Brazilian Supreme Electoral Tribunal website (http://www.tse.jus.br/).

The national Brazilian election process comprises a set of activities to organize, inspect, and execute elections according to the present electoral legislation. The electoral tribunal is responsible for examining parties and candidates, ensuring the compliance of the legislation, rules, and regulations during the electoral period and judging trials concerning the election. Although the voting process represents the central point in an election, there are many other activities that occur before and after casting the vote. For example, the candidature process, the electoral logistic (including the preparation and certification of each DRE), and financial analysis and approval are conducted before the voting process occurs. Additional checking procedures, statistics gathering and taking office procedures are typical activities performed after winners are known. In this case study, we focus on the voting process, although any other phase could be equally modeled using EVSec.

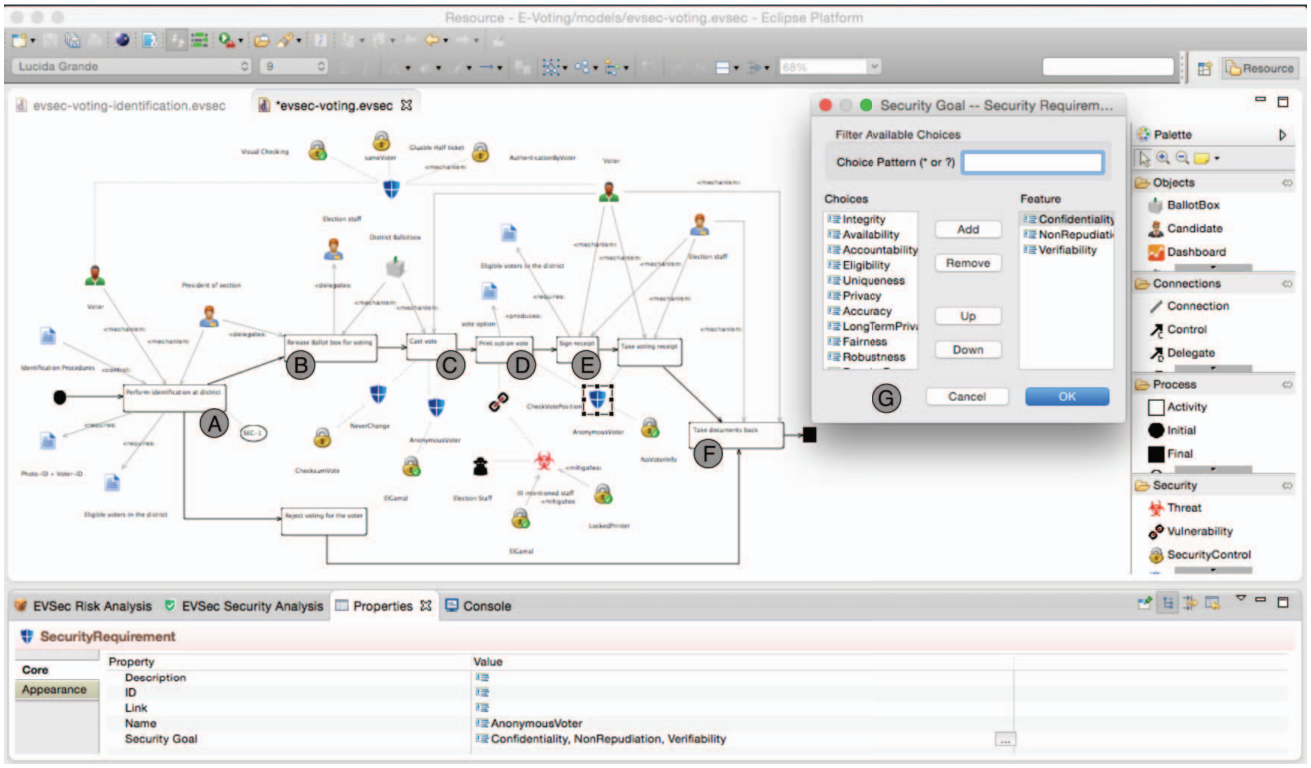
Voting using DREs is accomplished in three main steps: voter identification, casting, and tallying. These steps are part of an atomic process that occurs during voting to mitigate frauds based on forged public documents. Eligible voters must be registered in advance, as voting is currently mandatory for all Brazilian citizens from 18 to 70 years old. Thus, eligible voters must bear a voter ID[4]. This personal card contains the person's name, his birth date, a unique number, and other specific voting data (e.g., district and issuing date).

Ballot boxes are prepared for each district in which different candidates are competing for distinct positions, such as deputy, senator or president. Candidates must be affiliated to a party that also must be registered in advance to be validated. E-voting preparation starts with the definition of the top-level activities of an election: *(i)* setup the election *(ii)* run the election *(iii)* tallying votes *(iv)* disclose results.

During a voting session (see Figure 4), the voter travels to the poll place (district) and presents her credentials (A). As voter IDs have no photo, voters are required to present also a citizen card containing a photo. The president of the district checks the information and if valid, delegates to a staff representative the task of unlocking the ballot box to the voter (B). Then, the voter casts his vote (C), the vote option is printed (but not showed to the voter) (D), the voter signs a

[3]http://www.tse.jus.br/eleicoes/estatisticas/estatisticas-eleitorais-2014-eleitorado

[4]A biometric identification system has been gradually implemented since 2008 by the Brazilian government (http://www.tse.jus.br/eleicoes/processo-eleitoral-brasileiro/funcionamento-do-processo-eleitoral-no-brasil)



Property	Value
Description	
ID	
Link	
Name	AnonymousVoter
Security Goal	Confidentiality, NonRepudiation, Verifiability

Fig. 4: EVSec voting session model

receipt to confirm she voted (E), and the staff returns the ID cards and a copy of the receipt (F).

Security Modeling. While printing the vote, *CheckVotePosition* vulnerability has been identified, as observed in Figure 4. This weakness emerges when the vote can be revealed by comparing the printed option with the last voter. The threat, in this case, is an ill-intentioned staff, who could find on the printer the last vote. To mitigate this problem, two security controls are assigned: the application of a solution that keeps the voter options private (using Elgamal cryptography solution), and lock the printer inside the ballot box to prevent unauthorized access.

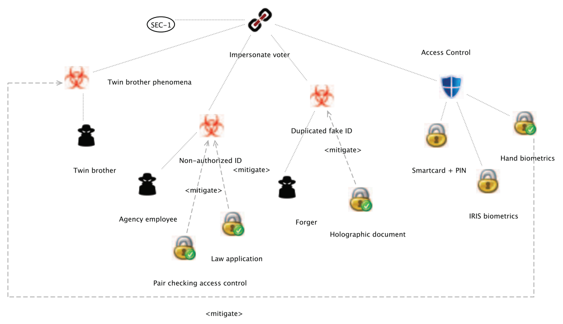
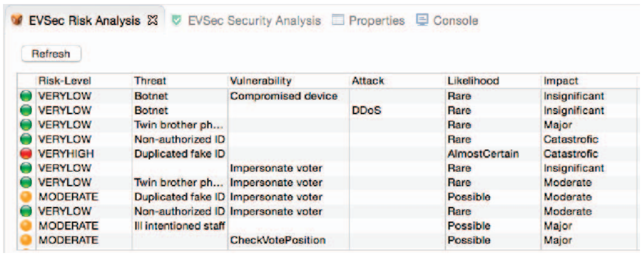


Fig. 5: Security model associated to the identification activity

During the identification process, vulnerabilities and threats were also identified, as defined by the sub-activity SEC-1 in Figure 5. The vulnerability *impersonate voter* raises when someone assumes a different identity on a temporary basis. As the identification is performed by a card containing the voter photo, some threats may emerge: *(i)* Twin brother phenomena: when exchanging the voter with his twin brother (impersonation). The risk likelihood could be set to rare, depending on the context of the elections, demography, etc., but the severity could be major, representing a breach of security in the process. *(ii)* Non-authorized ID: when authorities produce legitimate duplicated voter IDs for distinct voters. This could be controlled by the application of the law enforcement and pair checking access control on facilities of card manufacturing. In EvSec, such attackers may be set as internal. *(iii)* Duplicated fake ID: when bogus cards are emitted, compromising the entire election. The forger (attacker) can reproduce a card to be used by non-eligible voters. A countermeasure for this situation could be the use of some holographic technique to improve security of physical documents.

To mitigate the problem of impersonating, access of voters to the poll section needs to be controlled by, for example, using a PIN or biometrics mechanisms. Analyzing the identification model, there is an option to implement hand biometrics controls which could mitigate the twin brother threat.

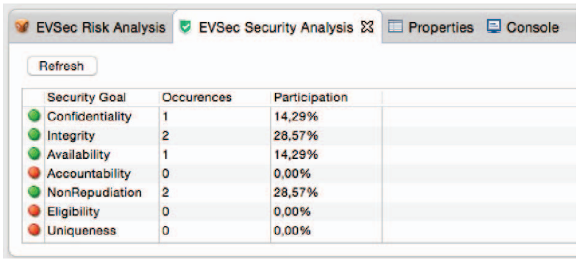
Risk Analysis. In EVSec, risks may be assigned to threats, vulnerabilities or attacks. Risk analysis is processed by reasoning about all models within the project and computing the risk of each security element (threat, vulnerability and attack). The risk analysis report (Figure 6) shows calculated risks for each security element individually and for combinations of security elements, such as risk of vulnerability–threat, threat–attack, and threat–attack–vulnerability. A visual colored scale points distinct risk levels in which the red icon represents very high risks and green icon very low risks. This allows engineers to focus on parts of the system that require special attention.



	Risk-Level	Threat	Vulnerability	Attack	Likelihood	Impact
🟢	VERYLOW	Botnet	Compromised device		Rare	Insignificant
🟢	VERYLOW	Botnet		DDoS	Rare	Insignificant
🟢	VERYLOW	Twin brother ph...			Rare	Major
🟢	VERYLOW	Non-authorized ID			Rare	Catastrofic
🔴	VERYHIGH	Duplicated fake ID			AlmostCertain	Catastrofic
🟢	VERYLOW		Impersonate voter		Rare	Insignificant
🟢	VERYLOW	Twin brother ph...	Impersonate voter		Rare	Moderate
🟠	MODERATE	Duplicated fake ID	Impersonate voter		Possible	Moderate
🟢	VERYLOW	Non-authorized ID	Impersonate voter		Rare	Moderate
🟠	MODERATE	Ill intentioned staff			Possible	Major
🟠	MODERATE		CheckVotePosition		Possible	Major

Fig. 6: EVSec Risk Analysis

Security Goal Completeness. Each security requirement is assigned to general and voting-specific security goals (Figure 4 - G). Thus, EVSec is able to analyze security aspects of the model and establish the model completeness. For example, missing voting security goals, such as incoercibility and uniqueness, can be easily verified by EVSec, by analyzing model structures. While these goals are not fully specified by the modeler, EVSec keeps models on incomplete state. The security analysis report (Figure 7) shows the status of security goals in the model, the occurrences of each goal and its participation in the model. Security goals without participation in the model are signaled with red icon. Intuitively, green icons represent some participation of the security goal in the model.



	Security Goal	Occurences	Participation
🟢	Confidentiality	1	14,29%
🟢	Integrity	2	28,57%
🟢	Availability	1	14,29%
🔴	Accountability	0	0,00%
🟢	NonRepudiation	2	28,57%
🔴	Eligibility	0	0,00%
🔴	Uniqueness	0	0,00%

Fig. 7: EVSec Security Analysis

Threats to Validity. This research was intended to investigate the feasibility of EVSec in modeling a real e-voting scenario. We chose the Brazilian case because it represents a large e-voting system involving a complex logistic, which has been successful employed in several elections. To understand the procedures and constraints involving Brazilian elections, a number of public documents containing laws, regulations and procedures were scrutinized. However, there is a lack of available (semi)formal models (such as a BPMN model) used during the development of the e-voting process. As such, one threat to internal validity consists of any misinterpretation introduced during our elicitation. This can lead to mistakes in the modeling process without knowing how it affects the investigation. An external threat concerns limitations to generalize the results of a study to other voting schemes, like an internet-based voting. In this sense, EVSec might evolve to express other concepts in its metamodel accordingly. With respect to the threats of construct validity, we underline the effect of using EVSec notation to express the concepts we believe they should express. Similarly, threats to conclusion validity might draw bias conclusions from a single study. Bearing in mind this is an initial effort towards a consolidated e-voting security modeling language, such validations are part of our future work.

VI. Related Work

EVSec models specific e-voting security concerns, and offers tool support for security completeness and risk analysis. These are the major differentiating characteristics when compared with the existing related work described next.

Several goal-oriented based approaches have been used to address security requirements. For example, Liu et al. [16] proposes a i*-based systematic approach to elicit and analyze security requirements. In [8], an extension of KAOS [40]–[42] is proposed to specify security concepts such as anti-goals, anti-requirements, vulnerabilities, attackers and threats. Secure Tropos [9] is a security-oriented extension of Tropos methodology that proposes the analysis of security issues starting from the business modeling phase. Similarly, social-technical security modeling language (STS-ml) [17] provides resources for security requirements specification under social, authorization and information views. In the first decade of the new century, UML-based security modeling has been explored by profiles that extend the language with stereotypes to integrate security elements. For example, SecureUML [10] is designed to integrate access control using RBAC (role-based access control) in application models. Similarly, UMLSec [11] applies stereotypes along with tags and constraints to formulate security requirements and assumptions on the system environment. Misuse cases [12] and security use cases [13] are designed to specify and analyze security threats and security requirements, respectively. Also, CORAS [14], [15], a model-based risk management process for security-critical systems, uses UML models to capture the concept of adverse environment by introducing the concept of “Threat Scenario". More recently, Almorsy and Grundy [43] present a visual language to model enterprise security management process (SecDSVL) and maintain traceability between different levels of abstraction of security details. SecBPMN [18] and SecureBPMN [19] extend BPMN with security aspects represented by graphical syntax linked with existing elements of a BPMN model.

VII. Conclusions and Future Work

EVSec is a DSML that assists engineers modeling e-voting processes not only to identify security requirements (white side), but also attacks, threats and vulnerabilities (black side) that may compromise the voting process. EVSec keeps tracking of security control choices, including the rational behind these choices, and assists engineers during risk analysis. By offering abstractions in the same level of the e-voting domain, EVSec seeks to promote better communication between different stakeholders involved in the election process. Tool support allows to verify security goal completeness and identify critical parts of the model concerning to higher level risks. We believe that users familiarized with UML and security concepts would find a smooth learning curve to start modeling with EVSec. The presented case study, simulating a Brazilian national election scenario, illustrated how EVSec can assist requirements engineers on specifying activities of each phase of the election and identifying potential security vulnerabilities, threats and adequate countermeasures since the early stages of the system development.

We are currently planning further evaluation of EVSec with other e-voting schemes, as well as its usability with real users using the Moody's physics of notations principles. Also, support for risk analysis can be enriched with additional integration with risk methodologies that offer guidance to reason about threat likelihood and severity. Finally, security requirements and tactics catalogs could assist requirements engineers in the task of identifying appropriate solutions to mitigate vulnerabilities and threats.

Acknowledgments

The first author is assisted by CAPES foundation (process 0553-14-0). The authors are grateful for the support given by FCT and NOVA LINCS Research Laboratory (Ref. UID/CEC/04516/2013) during the development of this work.

References

[1] M. Stenbro, "A survey of modern electronic voting technologies," Master's thesis, Institutt for telematikk, 2010.

[2] F. Lehoucq, "Electoral fraud: Causes, types, and consequences," *Annual review of political science*, vol. 6, no. 1, pp. 233–256, 2003.

[3] P. Collier and P. C. Vicente, "Violence, bribery, and fraud: the political economy of elections in sub-saharan africa," *Public Choice*, vol. 153, no. 1-2, pp. 117–147, 2012.

[4] J. M. Davis III and S. Thomas, "Direct recording electronic voting machine and voting process," Dec. 10 1996. US Patent 5,583,329.

[5] G. Dini, "A secure and available electronic voting service for a large-scale distributed system," *Future Generation Computer Systems*, vol. 19, no. 1, pp. 69–85, 2003.

[6] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," in *Advances in Cryptology—AUSCRYPT'92*, pp. 244–251, Springer, 1993.

[7] D. Zissis, *Methodologies and Technologies for Designing Secure Electronic Voting Information Systems*. PhD thesis, UNIVERSITY OF THE AEGEAN, 2011.

[8] A. Van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *26th ICSE*, pp. 148–157, IEEE, 2004.

[9] H. Mouratidis and P. Giorgini, "Secure tropos: a security-oriented extension of the tropos methodology," *JSEKE*, vol. 17, no. 02, pp. 285–309, 2007.

[10] T. Lodderstedt, D. Basin, and J. Doser, "Secureuml: A uml-based modeling language for model-driven security," in *UML 2002*, pp. 426–441, Springer, 2002.

[11] J. Jürjens, "Umlsec: Extending uml for secure systems development," UML '02, (London, UK, UK), pp. 412–425, Springer, 2002.

[12] I. Alexander, "Misuse cases: Use cases with hostile intent," *IEEE Software*, vol. 20, no. 1, pp. 58–66, 2003.

[13] D. G. Firesmith, "Security use cases," *Journal of object technology*, vol. 2, no. 3, 2003.

[14] K. Stolen, F. den Braber, T. Dimitrakos, R. Fredriksen, B. A. Gran, S.-H. Houmb, M. S. Lund, Y. Stamatiou, and J. Aagedal, "Model-based risk assessment-the coras approach," in *iTrust Workshop*, 2002.

[15] F. den Braber, T. Dimitrakos, B. A. Gran, M. S. Lund, K. Stølen, and J. Ø. Aagedal, "The coras methodology: model-based risk assessment using uml and up," *UML and the Unified Process*, pp. 332–357, 2003.

[16] L. Liu, E. Yu, and J. Mylopoulos, "Analyzing security requirements as relationships among strategic actors," in *SREIS'02, Raleigh, North Carolina*, 2002.

[17] E. Paja, F. Dalpiaz, and P. Giorgini, "Designing secure socio-technical systems with sts-ml," in *iStar*, pp. 79–84, 2013.

[18] M. Salnitri, F. Dalpiaz, and P. Giorgini, "Modeling and verifying security policies in business processes," in *Enterprise, Business-Process and Information Systems Modeling*, pp. 200–214, Springer, 2014.

[19] A. D. Brucker, I. Hang, G. Lückemeyer, and R. Ruparel, "Securebpmn: Modeling and enforcing access control requirements in business processes," in *17th ACM SACMAT*, pp. 123–126, ACM, 2012.

[20] T. M. Buchsbaum, "E-voting: International developments and lessons learnt," *Electronic Voting in Europe Technology, Law, Politics and Society*, pp. 31–34, 2004.

[21] X. Yi and E. Okamoto, "Practical internet voting system," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 378–387, 2013.

[22] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer, 1st ed., Oct. 1999.

[23] Y.-Y. Chen, J.-K. Jan, and C.-L. Chen, "The design of a secure anonymous internet voting system," *Computers & Security*, vol. 23, no. 4, pp. 330–337, 2004.

[24] K. Sampigethaya and R. Poovendran, "A framework and taxonomy for comparison of electronic voting schemes," *Computers & Security*, vol. 25, no. 2, pp. 137–153, 2006.

[25] M. R. Clarkson, S. N. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," IEEE, 2008.

[26] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections (extended abstract)," STOC '94, (USA), pp. 544–553, ACM, 1994.

[27] R. Prieto-Díaz, "Domain analysis: An introduction," *ACM SIGSOFT Software Engineering Notes*, vol. 15, no. 2, pp. 47–54, 1990.

[28] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," tech. rep., DTIC Document, 1990.

[29] W. Stallings and L. Brown, *Computer Security*. Pearson Education, 2008.

[30] B. Guttman and E. A. Roback, *An introduction to computer security: the NIST handbook*. DIANE Publishing, 1995.

[31] R. Anderson, *Security engineering*. Wiley, 2008.

[32] D. Gollmann, *Computer Security*. Wiley, 2nd ed., 2008.

[33] M. Uma and G. Padmavathi, "A survey on various cyber attacks and their classification," *International Journal of Network Security*, vol. 15, no. 5, pp. 390–396, 2013.

[34] G. Loukas, D. Gan, and T. Vuong, "A taxonomy of cyber attack and defence mechanisms for emergency management networks," in *PERCOM*, pp. 534–539, IEEE, 2013.

[35] Y.-L. Chen *et al.*, "Idef0 function modeling," in *Modeling and Analysis of Enterprise and Information Systems*, pp. 98–122, Springer, 2009.

[36] D. L. Moody, "The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering," *Software Engineering, IEEE Transactions on*, vol. 35, no. 6, pp. 756–779, 2009.

[37] C. Atkinson and T. Kühne, "Reducing accidental complexity in domain models," *Software & Systems Modeling*, vol. 7, no. 3, pp. 345–359, 2008.

[38] C. Gralha, J. Araújo, and M. Goulão, "Metrics for measuring complexity and completeness for social goal models," *Information Systems*, vol. 53, pp. 346–362, 2015.

[39] R. S. Ross, "Guide for conducting risk assessments (nist sp-800-30rev1)," *The National Institute of Standards and Technology (NIST), Gaithersburg*, 2012.

[40] A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *SCP*, vol. 20, no. 1, pp. 3–50, 1993.

[41] A. Van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE TSE*, vol. 26, no. 10, pp. 978–1005, 2000.

[42] A. Van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *RE 2001*, pp. 249–262, IEEE, 2001.

[43] M. Almorsy and J. Grundy, "Secdsvl: A domain-specific visual language to support enterprise security modelling," in *ASWEC*, pp. 152–161, IEEE, 2014.