Modelling and Analysing Resilience as a Security Issue within UML[*]

Ricardo J. Rodríguez
Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
Zaragoza, Spain
rjrodriguez@unizar.es

José Merseguer
Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
Zaragoza, Spain
jmerse@unizar.es

Simona Bernardi
Dpto. de Informatica
Università di Torino
Torino, Italy
bernardi@di.unito.it

ABSTRACT

Modelling system security is not common practise in software projects yet. Among other problems, there is not a widely accepted methodology which unifies the actual heterogeneity of security issues when addressing a whole security specification. Certainly, the reality is even worse since there is not an accepted or standard common notation for carrying out the security specification. In this work, we study how modelling security issues, specifically resilience, could be integrated in the MARTE-DAM framework, which allows the expression of performance and dependability requirements in UML models. We base this claim on the close relationship between security and dependability. Indeed, MARTE proposes a framework for non-functional properties specification (NFP), while DAM exploits it for dependability purposes. So, our goal is to take advantage of the common NFP framework while the dependability and security concerns are modelled in a unified view. On the other hand, we consider that the resulting security specification will be useful for developing model in which security related properties, such as availability, will be analysed. We will clarify these claims by means of an example.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages; K.6.5 [**Computing Milieux**]: Security and Protection

General Terms

Security, Performance, Measurement

[*]This work has been partially supported by the European Community's Seventh Framework Programme under project DISC (Grant Agreement n.INFSO-ICT-224498) and by the project DPI2006-15390 of the Spanish Ministry of Science and Technology.

Keywords

UML profile, security, secure systems modelling, Petri net

1. INTRODUCTION

Some years ago Devanbu et al. [5] identified that modelling security requirements has not ever been considered as a part of system modelling. Today, although some advances have been made [9, 13], the problems of addressing security modelling still remain. Indeed, they are aggravated since every day security covers an increasingly broad range of fields or communities that rarely interplay. For example, a comprehensive system security specification should address, among many other issues, hardware issues (e.g., vulnerable networks and nodes), coding bugs, software components reporting denial of service or services at flooding risk.

In this sense, we consider it important to provide a common framework in which such heterogeneity can be addressed. Such a framework will allow us to specify a broad range of security requirements as non-functional properties (NFPs) from different fields but in the same setting.

The Unified Modelling Language (UML) [16] is a well-known solution and a comprehensive modelling language that allows to specify from the architectural to the deployment system views. On the other hand, UML can be tailored for analysis purposes through *profiling*. In particular, MARTE [17] profile has enabled UML to specify and analyse real-time (RT) and embedded systems. Although focussed on RT, MARTE sub-profiles for performance and schedulability have been proved useful for the modelling and analysis of a wide range of application domains. Recently, the non-standard profile for Dependability Analysis and Modelling (DAM) [4] accomplished the same for dependability [3]. Indeed, as DAM is a MARTE specialisation, they play together to specify different types of NFPs in UML models, so jointly describing performance and dependability requirements. This approach could enlighten how to specify for security.

On the other hand, the tight relation between dependability and security was clearly given in [3], where dependability encompasses different system attributes, concretely reliability, availability, maintainability, integrity and safety. Security is defined as composite of confidentiality, integrity and availability. Now we recall some illustrative case about this relation. For example, fault-tolerance techniques aim to maintain system availability even in the presence of faults. Lately such techniques are being investigated [23] to maintain availability also in the presence of malicious attacks.

So, it is difficult to keep separated the specification concerns related to security and dependability. On the contrary, we defend the synergy they provoke. Consequently, in this work we start exploring how to fit the security non-functional specification within the MARTE-DAM framework.

The MARTE-DAM solution proposes a set of stereotypes and tagged values that allow the expression of NFPs, which are eventually attached to those UML model elements they affect. So, we propose to identify the minimum and necessary set of stereotypes to express security NFPs in a UML model. Such set will make up a profile, we name Security Analysis and Modelling (SecAM), that has to fit into the MARTE-DAM context to properly achieve the desired dependability-security relationship. Considering that SecAM will have to offer stereotypes and tags to cope with security NFPs in the heterogeneous setting previously described this is a real challenge we cannot completely address here. In this work, we limit our study to describe a first SecAM package: the one dealing with AVI properties (i.e., Attack, Vulnerability and Intrusion). We will also describe how the whole SecAM approach should fit within MARTE-DAM. There are many examples of resilient systems that could be benefited by our approach, such as those protecting critical infrastructures, e.g. banks, electronic commerce sites or networks of power stations.

Once the security NFP specification is accomplished, there is no doubt it has to be also exploited for security analysis of the system. In this case, certification is another choice that also could be addressed in this context. This paper focuses on the security analysis issue. UML models have been used in the literature to, by model transformation, get a variety of formal models (e.g., fault trees, Petri nets) which are useful for security analysis. In [10] can be found a summary of resilience model-based evaluation using analytical techniques. In our opinion, the SecAM profile should not hamper the reuse of such works in this new setting. In this work we show, through an example, how a translation of UML state-machines into Petri nets, from literature, can be tailored to consider also the SecAM NFP annotation. The resulting Petri net, which effectively embeds the security NFP annotation, allows to compute system availability metrics supporting different security parameters.

Before considering the paper's main contributions, it is worth recalling the context in which SecAM fits. An approach like MARTE enables UML to specify and analyse performance and schedulability. Actually, this specification increments the software engineer design work. Besides, the automatic generation and analysis of formal models, as a "by-product" of the software designs, frees the engineer from learning these usually complicated techniques. The loop will eventually be closed with tools capable of showing analysis results in the very same UML designs and capable of automatically enhancing these designs. The approach is so powerful that the inclusion of new specifications, in this case *security*, is possible and again complements the others, as will be shown in the remainder of the paper through SecAM.

In the sequel, Section 2 reviews the basics of MARTE-DAM. Section 3 presents our proposal for SecAM profile for UML. Section 4 describes the example where SecAM is applied, while Section 5 summarises how a formal model is obtained from UML-SecAM specification. Section 6 illustrates how some interesting results concerning system availability can be obtained from the formal model. The paper ends up revising the related work and summarising research directions in Section 7.

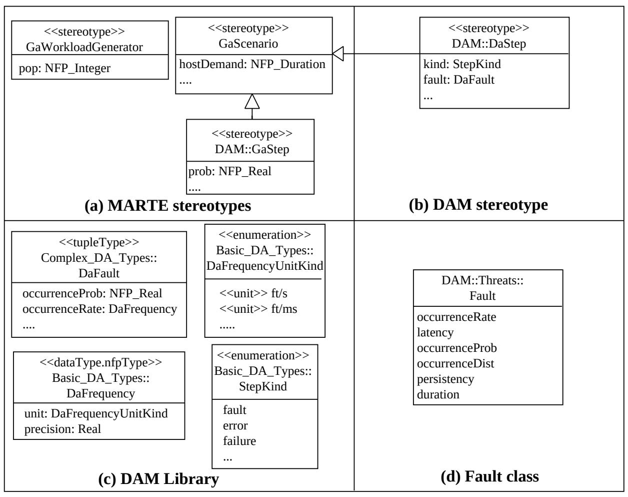
2. BACKGROUND

MARTE is a UML *lightweight* extension (i.e., through the use of UML stereotypes and tagged-values) for the modelling and analysis of real-time embedded systems. In particular, MARTE provides support for schedulability and performance analysis, and allows one to specify NFPs in these two fields, according to a well-defined Value Specification Language (VSL) syntax. The DAM profile is a MARTE specialisation for dependability analysis. Then, a MARTE-DAM annotation *stereotypes* the design model element it affects in the way UML proposes, i.e., by extending its semantics.

The entire set of MARTE stereotypes can be found in [17], while the DAM stereotypes, as well as the set of UML meta-classes that the stereotypes can be applied to, can be found in [4]. Figure 1 depicts an excerpt of the MARTE (a) and DAM stereotypes (b) and tagged values. According to UML, each stereotype is made of a set of tags which define its properties. For example, *GaScenario* stereotype has a *host-Demand* tag which is used to specify the CPU demand - in time units - for a scenario execution. Also a step execution can be specified (observe that *GaStep* is a sub-stereotype of *GaScenario*, then it inherits the tags of the latter). The types of the tags are basic UML types (e.g., integer, enumeration), or MARTE NFP types (e.g., *NFP_Duration*).

The DAM stereotypes specialise MARTE stereotypes (e.g., *DaStep*). Moreover, DAM enriches MARTE types library with basic and complex dependability types (c). The latter (e.g., *DaFault*) are composed of attributes that instead may be MARTE NFP types (e.g., *NFP_Real*), basic dependability types (e.g., *DaFrequency*), or enumeration types (e.g., *DaFrequencyUnitKind, StepKind*).

MARTE NFP types are data-types of special importance since they enable the description of relevant NFP aspects using properties such as: *value*, a value or parameter name (prefixed by the dollar symbol); *expr*, a VSL expression; *source*, the origin of the NFP - such as an assumed input parameter (*assm*), a requirement (*req*) or an estimated (*est*) parameter; and *statQ*, the type of statistical measure (e.g., mean).



<<stereotype>> GaWorkloadGenerator
pop: NFP_Integer
<<stereotype>> GaScenario
hostDemand: NFP_Duration
....
<<stereotype>> DAM::GaStep
prob: NFP_Real
....
<<stereotype>> DAM::DaStep
kind: StepKind
fault: DaFault
...
(a) MARTE stereotypes
(b) DAM stereotype
<<tupleType>> Complex_DA_Types:: DaFault
occurrenceProb: NFP_Real
occurrenceRate: DaFrequency
....
<<enumeration>> Basic_DA_Types:: DaFrequencyUnitKind
<<unit>> ft/s
<<unit>> ft/ms
.....
<<dataType.nfpType>> Basic_DA_Types:: DaFrequency
unit: DaFrequencyUnitKind
precision: Real
<<enumeration>> Basic_DA_Types:: StepKind
fault
error
failure
...
DAM::Threats:: Fault
occurrenceRate
latency
occurrenceProb
occurrenceDist
persistency
duration
(c) DAM Library
(d) Fault class

Figure 1: MARTE-DAM extensions.

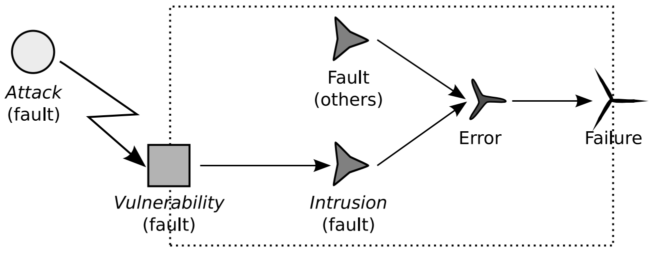


Figure 2: FEF and AVI chains.

3. SECAM PROFILE

The SecAM (Security Analysis and Modelling) profile aims at a comprehensive modelling of security issues in UML designs. As a general approach, we will define a domain model for each relevant security aspect (e.g., confidentiality, resilience or integrity). In this work, we address only the *Resilience* package.

Some definitions concerning existing possible threats in a system are needed before introducing the *Resilience* package. According to Avizienis et al.'s fault taxonomy [3], a *fault* is an (internal or external) impairment which exists in a system. When conditions needed for the activation of a fault are taking place, then the fault causes an *error*, defined as an internal detectable impairment. This error might provoke a *failure*, which is defined as an external visible impairment which causes a degradation in system functionality (e.g., denial of service on a server). In the dependability context, this causal relationship of fault, error and failure is named as FEF chain. This FEF chain is represented by the *DAM::Threats* package (see Figure 3 - top).

The *Resilience* package, shown in Figure 3 (bottom), has been constructed by revising the well-known AVI concepts in literature [1, 3, 8, 11]. The model is strongly supported by the Fault class from *DAM::Threats* package, which is a root for the whole AVI hierarchy. This means that vulnerabilities, intrusions and attacks are considered faults as in [3]. However, an abstract class *SecurityFault* allows to distinguish dependability faults (which are just faults) from security faults. The AVI model [22] is seen as a refinement of the FEF (fault/error/failure) chain [3]. In the AVI model the intrusion is only possible in the context of an attack that successfully exploits a system vulnerability, then the intrusion leads to an error which can provoke a failure [11]. Figure 2 depicts the AVI model and FEF chain. We have considered *Resilience* defined as the ability of a system to deal with unexpected conditions (intrusions in security context) without critical failures.

The Fault class (Figure 1(d)) has been extended with the following new attributes, taken from [3], useful for both the dependability and security domains:

- *occurrencePhase*: defines the phase of creation of the fault, it may distinguish between developmental (created during development phase) and operational (created during use phase) faults;
- *boundary*: establishes the limit where the fault is created. A fault is internal when created inside the system boundaries, otherwise it is external;
- *objective*: defines the intention of the fault, i.e., malicious (faults created with an intention of harming the system) or non-malicious;
- *causedBy*: defines the cause of the fault, i.e., a natural or a human-made cause;
- *intent*: specifies whether the fault is deliberate or not;
- *capability*: specifies whether the fault is accidental or it is due to incompetence.

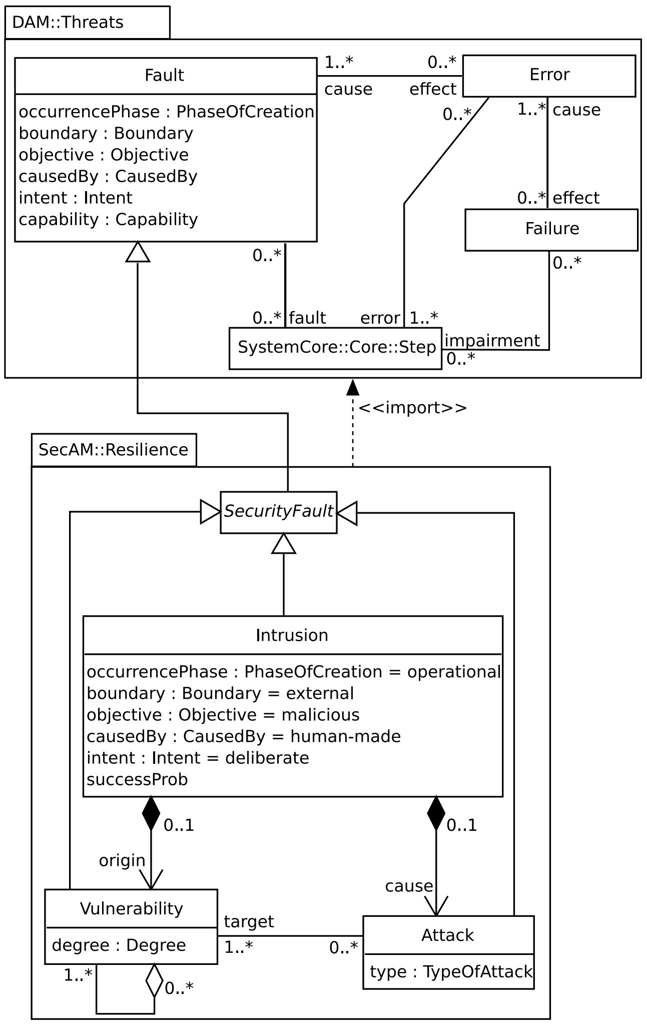


Figure 3: Resilience package.

The attributes added to the Fault class are of enumeration type and their possible values are shown in Figure 4.

Vulnerability is defined as an internal fault, either developmental or operational, which is present in a system. It can be either malicious or non-malicious, depending on whether the fault was created intentionally or not. The attribute *degree* provides a qualitative characterisation of the vulnerability, e.g., low, medium or high.

A vulnerability can be exploited by an *attack*, which is defined as an external, malicious and operational fault. The *type* of attack [8] can be active (e.g., Denial-of-Service or directly breaking into systems) or passive (e.g., sniffing or information gathering). Passive attacks do not try to gain access to the system but to collect information, so they can be a previous step for an active one.

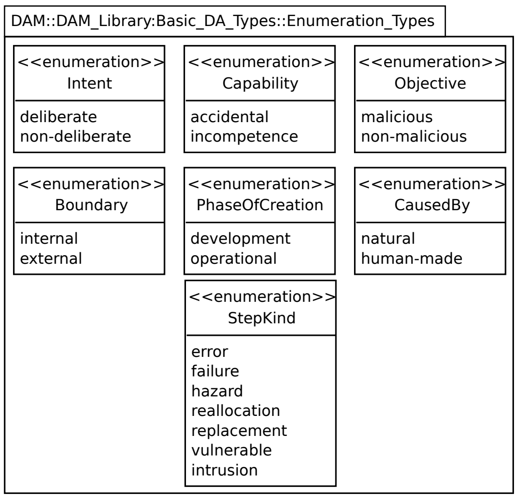


Figure 4: Attributes added to DAM library.

When the attack is successful, then it leads to an *intrusion*. So, intrusion is the consequence of a successful attack to an existing vulnerability in the system. We recall that intrusion is the only kind of security fault that can lead to an error, and this error can cause a failure in the system (e.g., unavailability).

In literature, intrusion is defined as an intentionally malicious operational and externally induced fault. Moreover, it is also seen as a composite fault [1]. According to this, we model an intrusion with a composite association with vulnerability and attack, so an intrusion only exists in the context of a vulnerability and an attack. The *successProb* attribute defines the probability of having a successful attack when exploiting a certain vulnerability.

Once the domain model is constructed, then we have to define the stereotypes and tagged values that will make up the SecAM Resilience package. We accomplished this task following the rules in [12, 21]. Moreover, we wanted to be sparing with the number of stereotypes since the goal is to provide the minimum number of artefacts to annotate the model. An overview of the SecAM profile is depicted in Figure 5, together with the *import* relations between SecAM, DAM and MARTE profiles. SecAM profile is composed of two main packages: a model library, that contains security-specific types, and the proper UML security extensions (i.e., stereotypes, attributes and constraints) that map the contents of the SecAM domain model previously presented.

The SecAM library, that includes the data-types used in the definition of the profile, is detailed in Figure 6. Each security fault class of the *Resilience* package is mapped into a complex data-type; to keep trace of the mapping, the latter is named with the name of the corresponding class prefixed by *Seca*. So complex type *SecaVulnerable* maps class *Vulnerable*, *SecaAttack* maps *Attack* and *SecaIntrusion* the *Intrusion* class. Regarding the latter, we remark the tags *origin* of type *SecaVunerable* and *cause* of type *SecaAttack*, since they map the corresponding vulnerability and attack.

In contrast with [12], we avoided to add a tagged value and an OCL constraint to represent the association-end between Attack and Vulnerable classes. We chose this solution because the association is implicit in the *SecaIntrusion* complex type through its attributes *origin* and *cause*.

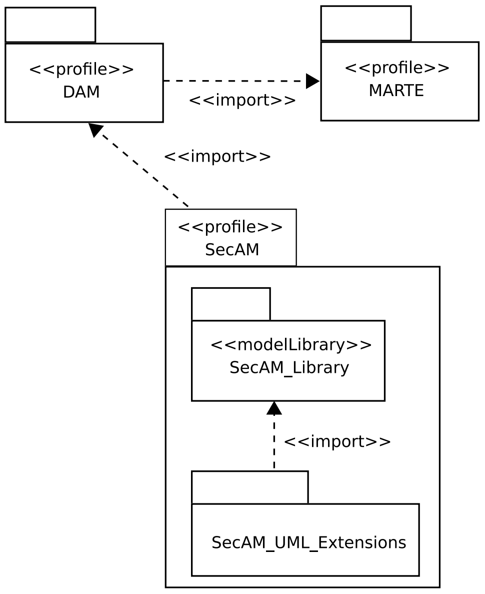


Figure 5: SecAM profile overview.

The SecAM extensions are shown in Figure 7. A new stereotype called ≪*SecaAttackGenerator*≫, with tag *attack* of type *SecaAttack*, models attack faults and will be useful to point out, in the UML design, the sources of attack. It extends the same UML meta-classes as its super-stereotypes (i.e., the DAM stereotype ≪*DaFaultGenerator*≫ and the MARTE stereotype ≪*DaWorkloadGenerator*≫) since it is a specialisation of the latter. We have also specialised the DAM stereotype ≪*DaStep*≫ in a new one, ≪*SecaStep*≫, with two tagged values: *vulnerability* of type *SecaVulnerable* and *intrusion* of type *SecaIntrusion*. The SecAM stereotypes can be applied to a wide set of behaviour-related elements covered by some UML meta-classes, such as: actions, activities, trigger events, transitions, and states in state charts diagrams, messages and interaction fragments in interaction diagrams. This small set of stereotypes, in conjunction with MARTE and DAM ones, enables a designer to annotate a UML design allowing to obtain a model that accounts for dependability-security parameters.

4. EXAMPLE

In this section, we aim to illustrate, through an example, how the stereotypes and tagged values in the SecAM's Resilience package should be used from a practitioner point of view. The example will be eventually useful to obtain a formal model and analysis figures to better understand the system availability.

We have supposed a system composed of an advanced firewall that integrates a controller (or monitor) internal device in charge of monitoring the behaviour of the firewall. The firewall is exposed to attacks from untrusted networks, which can be either WAN or external LAN. These attacks have an external origin and the objective is to harm the system, either compromising its availability (one of the attributes of security according to [3]) or gaining illegal access to the protected LAN (which could compromise system confidentiality and integrity, the other two attributes of security, also according to [3]). Each time a new message is received

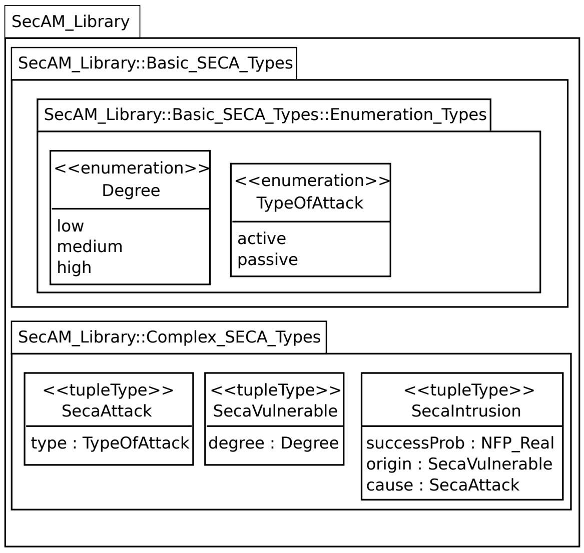

SecAM_Library
SecAM_Library::Basic_SECA_Types
SecAM_Library::Basic_SECA_Types::Enumeration_Types
<<enumeration>> Degree
low
medium
high
<<enumeration>> TypeOfAttack
active
passive
SecAM_Library::Complex_SECA_Types
<<tupleType>> SecaAttack
type : TypeOfAttack
<<tupleType>> SecaVulnerable
degree : Degree
<<tupleType>> SecaIntrusion
successProb : NFP_Real
origin : SecaVulnerable
cause : SecaAttack

Figure 6: SecAM library.

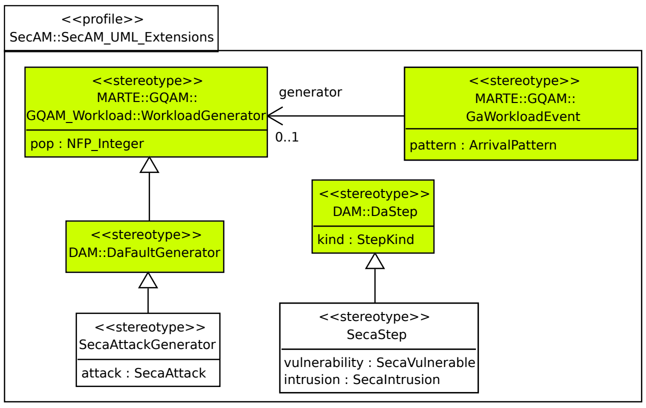

<<profile>> SecAM::SecAM_UML_Extensions
<<stereotype>> MARTE::GQAM:: GQAM_Workload::WorkloadGenerator
pop : NFP_Integer
generator
0..1
<<stereotype>> MARTE::GQAM:: GaWorkloadEvent
pattern : ArrivalPattern
<<stereotype>> DAM::DaFaultGenerator
<<stereotype>> DAM::DaStep
kind : StepKind
<<stereotype>> SecaAttackGenerator
attack : SecaAttack
<<stereotype>> SecaStep
vulnerability : SecaVulnerable
intrusion : SecaIntrusion

Figure 7: SecAM UML extensions.

from the untrusted network, a process inside the firewall component is in charge of attending it. There exists a limit in the number of processes created by the firewall.

Currently, the kind of critical information systems exposed to attacks are of much interest in literature (e.g., MAFTIA [1], CRUTIAL [23] or SITAR [24]). However, consider that we do not pretend to model intricacies of these kind of systems but only to exemplify the use of the Resilience package.

In contrast with the firewall, which is vulnerable, the monitor is invulnerable. It can be seen as an embedded system which is tamper-proof and it cannot be intentionally modified. Its mission is to activate a time-out to eventually check the firewall processes and to clean up those hung.

Figures 9 and 10 show the monitor state-chart and the process state-chart, respectively. Both state-charts have been annotated with MARTE and SecAM profiles. The structural view of the system is depicted in Figure 8.

The monitor, Figure 9, starts by setting a time-out with period *TOdelay* milliseconds. The *countDown()* activity has been annotated with MARTE stereotype ≪*gaStep*≫ to indicate its duration with tag *hostDemand*.

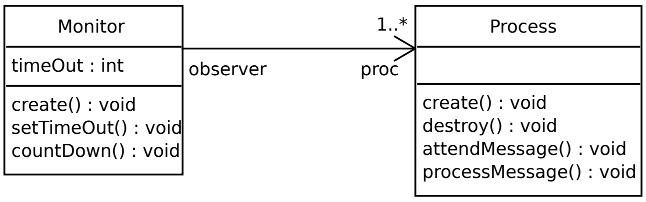


Figure 8: Class diagram.

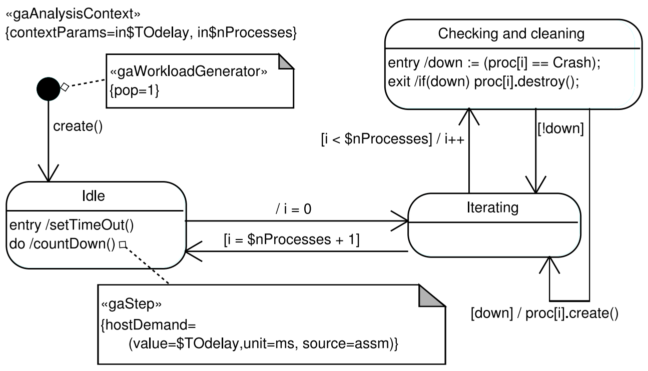


Figure 9: Monitor state-chart diagram.

Regarding the firewall (Figure 10), we have assumed a limitation in the population of processes, which is represented using variable *nProcesses*. An open workload generates messages (e.g., the message is coming through the WAN network) with an inter-arrival time defined as an exponential distribution with mean equal to *netLoad* milliseconds. Incoming messages are also stereotyped as ≪*secaAttackGenerator*≫, to specify that the message could be an attack. This attack is active (*type* tagged value) and it has been characterised with a certain occurrence probability (*attack*). The activity processing a message (the do-activity in *Processing* state) takes, on average, *Tprocess* milliseconds. It has been stereotyped as ≪*secaStep*≫ since it is the vulnerable part of the process, with a high degree of vulnerability. The state *Processing* owns two immediate outgoing transitions, the annotations attached to the latter are meant to resolve this conflict. When the incoming message is a successful attack, then an intrusion arises that leads to a process crash. Otherwise, the process comes back to the *Idle* state again. The transition from *Processing* to *Crash* is specified as an intrusion step and annotated with the probability of a successful attack (*success*).

Finally, we have supposed that when an intrusion succeeds (state *Crash*), the process starts some malicious activity.

5. OBTAINING A FORMAL MODEL

We will work out the UML state-charts (UML-SC) depicted in Figures 9 and 10 to obtain a formal model that will allow us to get some security metrics also considering performance parameters.

5.1 Conversion of UML-SC into Petri Nets

The topic of translating UML-SC into Petri nets has been extensively addressed in the literature. In this work, we follow the translation proposed in [15], but since it was given

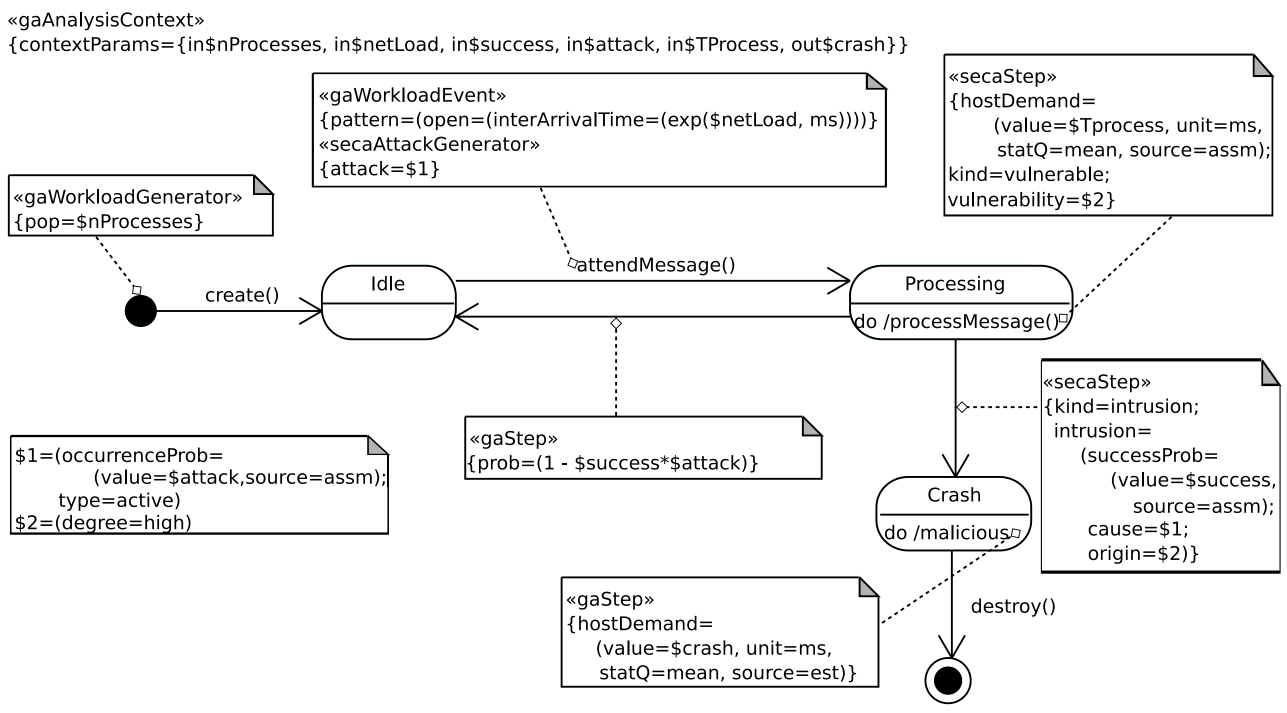


Figure 10: Process state-chart diagram.

for performance analysis purposes then some minor changes will arise. The algorithm in [15] has been implemented in the ArgoSPE [6] tool and it takes as input a UML-SC annotated with SPT [18] (Schedulability, Performance and Time profile, precursor of MARTE). In the following, we recall the translation approach in [15].

For each SC simple state, a PN place represents its entrance. The latter is followed by two causally connected PN transitions which represent, respectively, the entry action and the do-activity of the SC state. Entry actions are modelled by immediate PN transitions, since their execution time is assumed to be negligible. Do-activities are represented, instead, by timed PN transitions, which are characterised by one output place (i.e., the completion place) modelling the SC completion state. Each SC outgoing immediate transition is translated into a PN immediate transition with the completion place as its input place. For conflicting outgoing transitions, as it happens in Figure 10 in the `Processing` state, the algorithm adds immediate transitions with probabilities to stochastically resolve the choice. In this case, the probabilities are taken from the annotations attached to the transitions. The PN subnets derived from the translation of the SC simple states together with their outgoing transitions are then composed over interface places to get the PN of the whole SC. Finally, since the UML-SCs communicate via events, the final analysable PN model, such as the one in Figure 11, is obtained by composing the PN models of the different UML-SCs over places that model event mailboxes.

Since ArgoSPE does not deal with MARTE, nor DAM nor SecAM annotations, we have worked out the Petri net to incorporate such annotations. Chiefly, these annotations are converted into the net parameters, they are described in Table 1. However, it deserves special attention the `gaWorkloadEvent` annotation in Figure 10, since from it we have manually produced the subnet representing the open workload in Figure 11. As the reader can observe, we have also considerably simplified the other subnets in Figure 11 from the ones automatically obtained, so to gain readability while the behaviour is preserved.

5.2 Discussion of the Obtained Petri Net

The DSPN [2] (Deterministic and Stochastic Petri Net) depicted in Figure 11 has been obtained joining the three previously referred subnets: the open workload (that is, the incoming messages to the firewall), the behaviour of the each process and the monitor subnet.

The pattern of arrival for messages (event *attendMessage*, in Figure 10) is an open workload with an exponentially distributed inter-arrival time of mean value *netLoad*. In the DSPN, transition $T1|NetworkLoad$ models it with a firing rate equal to $\frac{1}{netLoad}$.

The limit number of processes created by the firewall (parameter *nProcesses* of the *pop* tag, annotated to the initial state in Figure 10) corresponds to the initial marking of place $P4|ProcessIdle$. The firing of transition $t2|attend\ Message$ represents the change of state from *Idle* to *Processing*. Transition $T3|processMessage$ models the do-activity in *Processing* state; its firing rate is set to the inverse of the parameter assigned to the *hostDemand* tag, i.e., $1/Tprocess$.

Once the do-activity has finished, the process changes state. The choice of the target state will depend on the processed message. Transition $t4|Intrusion$ models the case of a successful attack. The latter can occur with probability $attack \cdot success$, where *attack* is the probability that the message is an attack (*attack.occurrenceProb* tagged value of ≪*secaAttackGenerator*≫ stereotype), and *success* is the

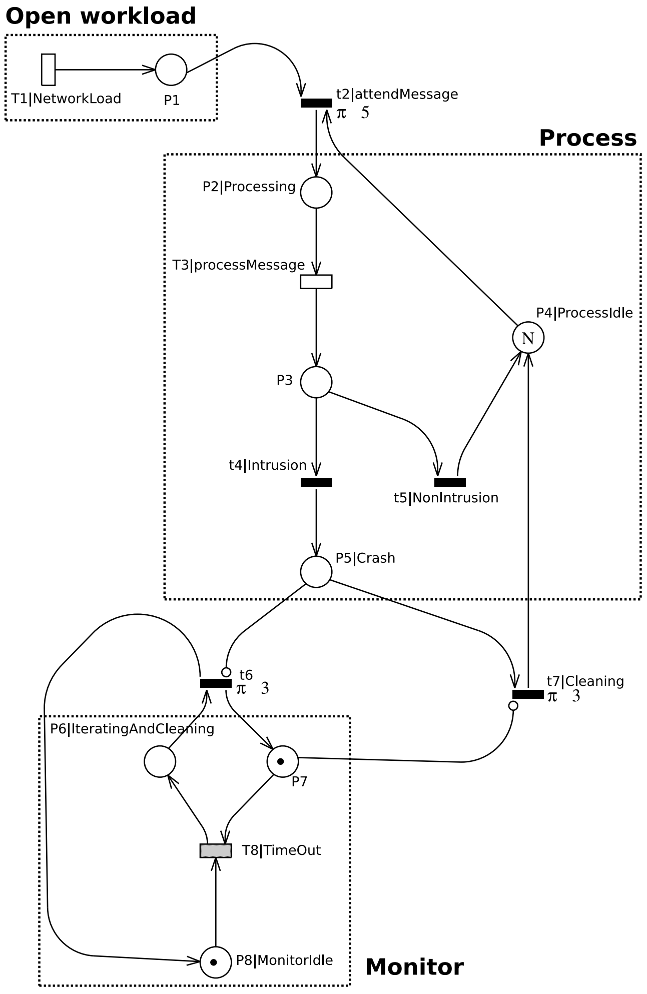


Figure 11: DSPN obtained from UML state-charts.

probability that the attack succeeds (*successProb* tagged-value of ≪*secaStep*≫ stereotype). If an attack is successful then the process has been intruded and, consequently, moves to the *Crash* state (place *P5*|*Crash*). Otherwise, the process moves back to its idle state (place *P4*|*ProcessIdle*). This last choice is modeled by transition *t5*|*NonIntrusion* and it will occur with a probability $1 - attack \cdot success$.

Place *P5*|*Crash* abstracts the do-activity *malicious* of the process in *Crash* state (Figure 10), with a mean duration *crash*. The crashed process will be instantaneously recovered once the monitor starts its cleaning activity. Therefore, the *crash* tagged value is the mean sojourn time in *Crash* state, that in our example corresponds to the "Mean Time to Detect an Intrusion" (MTTDI). In the Section 6 we have calculated the steady state availability as $\dfrac{MTTF}{MTTF + MTTDI}$, where MTTF corresponds to the mean sojourn time in the "up" states of the process (*Idle* and *Processing* states).

Regarding the monitor net, the initial marking in place *P8*|*MonitorIdle* represents the only monitor.

Transition *T8*|*TimeOut* represents the do-activity *countDown* in *Idle* state (Figure 9) and it is characterised by a deterministic duration equal to *TOdelay* (value of the *hostDemand* tag). Once it has fired, the cleaning activity starts: a token is set in place *P6*|*IteratingAndClearning* and the token in place *P7* is consumed. As place *P7* is empty, then transition *t7*|*Cleaning* is enabled and can fire if there are tokens in *P5*|*Crash* . This transition is abstracting the cleaning process made by the monitor (that is, the destruction and creation of any hung process): it is an immediate transition since we have assumed the duration of such an activity to be negligible. Once the cleaning of hunged processes has finished (that is, no tokens in place *P5*|*Crash*), then transition *t6* is enabled and can fire, so the monitor moves back to the *Idle* state (place *P8*|*MonitorIdle*) starting the count down again.

A summary of DSPN parameters is shown in Table 1.

Place	Initial marking	Value
P4|Idle	*nProcesses*	6

Transition	Parameter (type)	Value(s)
T1|NetworkLoad	1/*netLoad* (rate)	0.01, 0.05, 0.1/ms
T3|processMessage	1/*Tprocess* (rate)	0.2/ms
T8|TimeOut	*TOdelay* (delay)	1, 100ms
t4|Intrusion	$attack \cdot success$ (weight)	
t5|NonIntrusion	$1 - attack \cdot success$ (weight)	

Parameter	Values
attack	[0.01 . . . 0.5]
success	[0.01 . . . 0.5]

Table 1: DSPN parameters.

6. EXPERIMENTS AND RESULTS

We have analysed one aspect of the firewall system security, that is the availability, in steady state, of processes able to attend the incoming messages from the untrusted network. The DSPN model in Figure 11 has been then used to compute the steady state availability. Such metric, at DSPN model level, is defined as $\frac{MTTF}{MTTF+MTTDI} = 1 - \frac{E[P5|Crash]}{N}$, where $E[P5|Crash]$ is the mean number of tokens in place $P5|Crash$ (which represents the unavailable state of the process) and N is the total number of firewall processes. We have used the simulator implemented in GreatSPN [7] to analyse the DSPN model.

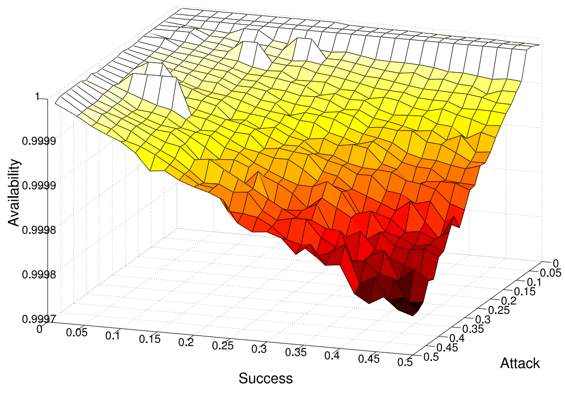
The objective of the experiments is to study the system availability under different assumptions of probability of attacks and probability of successful attacks, considering different types of workload and time-out durations. The model input parameters are summarised in Table 1 and the results are shown in Figure 12.

The six figures plot the availability trend by varying the probability of attack and of successful attack from 1% up to 50%. Each figure shows the steady state availability under a different combination of workload rate and time-out duration. In particular, we assumed three types of network workload:

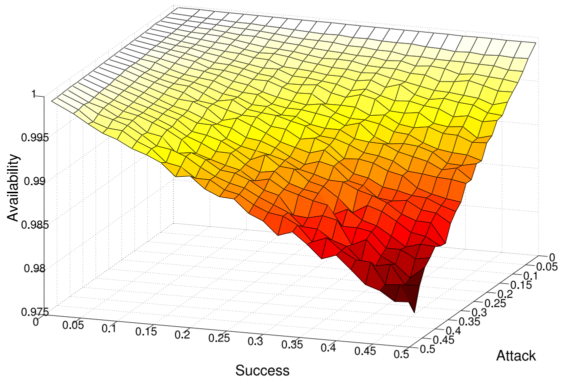
- low, i.e., 0.01/ms (Figures 12(a,b)),
- high, i.e, 0.05/ms (Figures 12(c,d)), and
- very high, i.e., 0.1/ms (Figures 12(e,f))

and, two types of time-out durations:

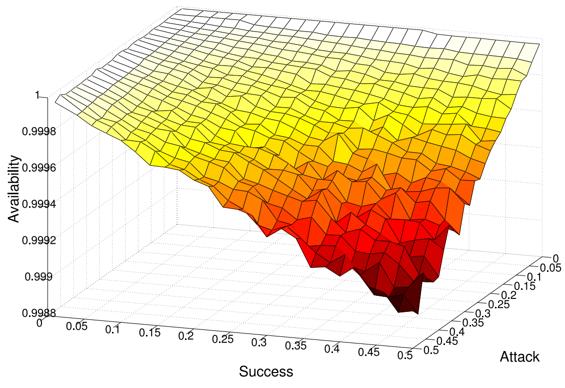
- short, i.e, 1ms (Figures 12(a,c,e)), and



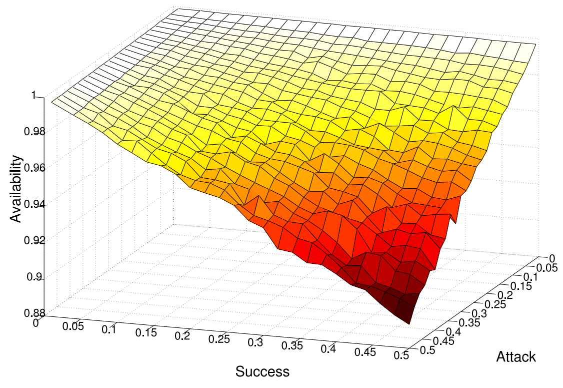
(a) low workload, short time-out



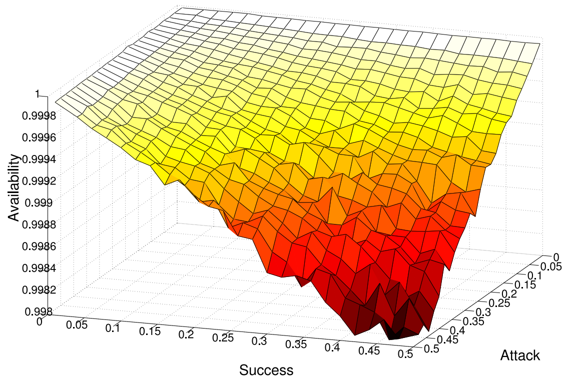
(b) low workload, long time-out



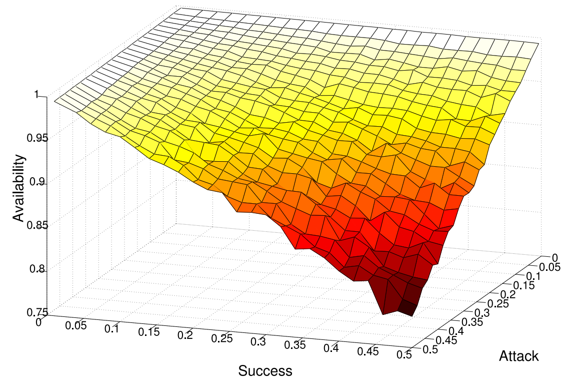
(c) high workload, short time-out



(d) high workload, long time-out



(e) very high workload, short time-out



(f) very high workload, long time-out

Figure 12: Steady state availability under different workload and time-out assumptions.

- long, i.e., 100ms (Figures 12(b,d,f)).

In all the experiments, as expected, the system availability is in inverse proportion to the probability of attacks and of successful attacks. The main difference among the figures is the slope of the surfaces that provides information on the availability decreasing factor, meaning that the system availability is sensitive to the network workload and monitor time-out assumptions. In particular, the decreasing factor is higher for higher workloads and for longer time-out duration. For example, it is equal to 0.021%, in case of low network workload and short time-out duration (Figure 12(a)), while it reaches 20.9% when very high network workload and long time-out duration are assumed (Figure 12(f)). Indeed, the messages incoming to the firewall are potential attack carriers, so the frequency of attacks increases from low to very high network workload (e.g., compare Figures 12(b,d,f)). On the other hand, when a short time-out duration is set (Figures 12(a,c,e)), the monitor is able to detect promptly an attack to the firewall and, then, immediately to recover by destroying the crashed process and creating a new one.

Observe that the isolated hills close to 100% availability, in Figure 12(a), are due to the simulation accuracy used. Nevertheless, their height is lower than 0.01%.

It is worth noting that, in the firewall example, false alarms (i.e., the time-out expires and no process is crashed), which often occur when short time-out duration is set, do not provoke side effects in the system. A different behaviour, and then availability trend, could be observed if timed activities, such as system integrity checks, have to be carried out as consequence of time-out expiration. The effect of false alarms on the system availability deserves further analysis which we aim to carry out as part of our future work.

7. RELATED WORK AND CONCLUSIONS

The closest related works are those approaches that in recent years have worked out the security specification in the UML framework, mainly [9, 13]. In some sense our proposal builds on them while it tries to embrace the more recent UML advances on non-functional specification. Moreover our proposal differs from them since we have also devised the need and interest on a joint dependability-security specification and analysis. SecureUML [13] focuses and makes an effort on annotating static UML design models, however behavioural models, such as state machines, where not addressed. In profiling approaches, such as MARTE, stereotypes are applied to meta-classes and this modelling feature supports their reuse through UML diagrams. For instance, our example has annotated only state machines, but our stereotypes, e.g. `SecaStep`, can be applied to `Action` UML meta-class since it inherits from `DaStep`. As a conclusion, also Activity diagrams, Sequence diagrams or Use Cases could be annotated with `SecaStep`.

The other approach, UMLsec [9], permits to stereotype a UML model to indicate some interesting security aspects and also to specify the static concept of security underlying in the UML model. However, it does not worry about how security aspects could have an influence on the throughput of the system, for example. Consider that our approach could hold these aspects since MARTE allows to specify the performance view. Another interesting aspect of both approaches is that they focus on the design phase and allow to check the models, for example, using a model checker.

Another work close to ours is [19]. Here, the authors analyse the performance of different security solutions while using the "aspects" modelling technique. This work is not focussed on giving a unified framework for security and performance specification, however the use of "aspects" helps to address together these crosscutting concerns.

Regarding derivation of UML dependability models into stochastic Petri nets, it is worth mentioning the works presented in [14, 20]. However, the approach in [20] is exclusively for the dependability field, without considering security issues, and likely very bound to AADL (Architecture Analysis & Design Language). In [14], which is closer to ours, Bondavalli et al. identify dependability attributes in early design phases of the system and construct a dependability model (as a Timed Petri Net) using graph transformation techniques in structural UML diagrams.

As a conclusion we think that our approach should bring UML annotated models where to carry out analysis of relevant dependability-security aspects. These analyses will also consider the system performance characteristics (activities durations or routing rates) which in some cases are very relevant for security, for example, allowing to measure the real impact of introducing more security layers (software or hardware).

As a future work, tools supporting the approach are necessary. However, using a standardised approach based on UML and MARTE is one way to ease reuse of existing tools (modelling tools but also analysis tools that are currently being developed for MARTE). Therefore, effort is only focused on the security analysis on top of existing tool sets. On the other hand, it has to also be pointed out that a lot of effort is necessary to adequately bring most of the security fields to this framework.

Acknowledgements

The authors would like to thank the anonymous referees who reviewed this paper and helped to improve it.

8. REFERENCES

[1] A. Adelsbach, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J. Laprie, D. Powell, B. Randell, J. Riodan, P. Ryan, and Others. Conceptual Model and Architecture of MAFTIA. Technical report, Department of Informatics, University of Lisbon, Lisbon, 2003.

[2] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In *Advances in Petri Nets 1987, 7th European Workshop on Applications and Theory of Petri Nets*, pages 132–145, London, UK, 1987. Springer-Verlag.

[3] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[4] S. Bernardi, J. Merseguer, and D. Petriu. A Dependability Profile within MARTE. *Journal of Software and Systems Modeling*, 2009. DOI: 10.1007/s10270-009-0128-1.

[5] P. T. Devanbu and S. G. Stubblebine. Software engineering for security: a roadmap. In *International*

Conference on Software Engineering (ICSE) - Future of Software Engineering Track, pages 227–239, 2000.

[6] E. Gómez-Martínez and J. Merseguer. ArgoSPE: Model-Based Software Performance Engineering. In S. Donatelli and P. S. Thiagarajan, editors, *ICATPN*, volume 4024 of *Lecture Notes in Computer Science*, pages 401–410. Springer, 2006.

[7] The GreatSPN tool `http://www.di.unitorino.it/~greatspn`, 2002. University of Torino.

[8] T. HoneyNet Project, editor. *Know Your Enemy: Learning about Security Threats*. Addison Wesley Publishing, 2nd edition, 2004.

[9] J. Jürjens. UMLsec: Extending UML for Secure Systems Development. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 412–425, London, UK, 2002. Springer-Verlag.

[10] M. Kaâniche, P. Lollini, A. Bondavalli, and K. Kanoun. Modeling the Resilience of Large and Evolving Systems. *International Journal of Performability Engineering*, 4(2):153–168, April 2008.

[11] R. Khan and K. Mustafa. From threat to security indexing: a causal chain. *Computer Fraud & Security*, 2009:9–12, 2009.

[12] F. Lagarde, H. Espinoza, F. Terrier, and S. Gérard. Improving UML Profile Design Practices by Leveraging Conceptual Domain Models. In *ASE '07: Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, pages 445–448, New York, NY, USA, November 2007. ACM.

[13] T. Lodderstedt, D. A. Basin, and J. Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 426–441, London, UK, 2002. Springer-Verlag.

[14] I. Majzik, A. Pataricza, and A. Bondavalli. Stochastic Dependability Analysis of System Architecture Based on UML Models. In R. De Lemos, C. Gacek, and A. Romanovsky, editors, *Architecting Dependable Systems, LNCS 2677*, Lecture Notes in Computer Science, pages 219–244. Springer-Verlag, Berlin, Heidelberg, New York, 2003.

[15] J. Merseguer, S. Bernardi, J. Campos, and S. Donatelli. A Compositional Semantics for UML State Machines Aimed at Performance Evaluation. In A. Giua and M. Silva, editors, *Proceedings of the 6th International Workshop on Discrete Event Systems*, pages 295–302, Zaragoza, Spain, October 2002. IEEE Computer Society Press.

[16] Object Management Group. *Unified Modelling Language: Superstructure*, July 2005. Version 2.0, formal/05-07-04.

[17] Object Management Group. *A UML profile for Modeling and Analysis of Real Time Embedded Systems (MARTE)*, November 2009. Version 1.0, formal/2009-11-02.

[18] OMG. UML Profile for Schedulability, Performance, and Time. Version 1.1, formal/05-01-02, January 2005.

[19] D. C. Petriu, C. M. Woodside, D. B. Petriu, J. Xu, T. Israr, G. Georg, R. France, J. M. Bieman, S. H. Houmb, and J. Jürjens. Performance analysis of security aspects in UML models. In *WOSP '07: Proceedings of the 6th International Workshop on Software and Performance*, pages 91–102, New York, NY, USA, 2007. ACM.

[20] A. E. Rugina, K. Kanoun, and M. Kaâniche. A System Dependability Modeling Framework Using AADL and GSPNs. In R. de Lemos, C. Gacek, and A. B. Romanovsky, editors, *WADS*, volume 4615 of *Lecture Notes in Computer Science*, pages 14–38. Springer, 2006.

[21] B. Selic. A Systematic Approach to Domain-Specific Language Design Using UML. In *10th IEEE Int.l Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*, pages 2–9. IEEE Computer Society, 2007.

[22] P. Veríssimo, N. Neves, and M. Correia. Intrusion-Tolerant Architectures: Concepts and Design. *Lecture Notes in Computer Science*, 11583:3–36, 2003.

[23] P. Veríssimo, N. F. Neves, M. Correia, Y. Deswarte, A. A. E. Kalam, A. Bondavalli, and A. Daidone. The CRUTIAL Architecture for Critical Information Infrastructures. In R. de Lemos, F. D. Giandomenico, C. Gacek, H. Muccini, and M. Vieira, editors, *WADS*, volume 5135 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2007.

[24] F. Wang, F. Jou, F. Gong, C. Sargor, K. Goseva-Popstojanova, and K. Trivedi. SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services. In *Foundations of Intrusion Tolerant Systems, 2003*, pages 359–367, 2003.