



**Project no.:** IST-FP6-STREP - 027513  
**Project full title:** Critical Utility InfrastructurAL Resilience  
**Project Acronym:** CRUTIAL  
**Start date of the project:** 01/01/2006      **Duration:** 36 months

**Deliverable no.:** D11

**Title of the deliverable:** List of requirements on formalisms and selection of appropriate tools

**Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)**

<b>Contractual Date of Delivery to the CEC:</b>	<b>18/01/2008</b>
<b>Actual Date of Delivery to the CEC:</b>	<b>18/01/2008</b>
<b>Organisation name of lead contractor for this deliverable:</b> CNIT	
<b>Author(s):</b> Susanna Donatelli <sup>6</sup> (editor), Silvano Chiaradonna <sup>3</sup> , Daniele Codetta <sup>6</sup> , Felicita Di Giandomenico <sup>3</sup> , Giuliana Franceschinis <sup>6</sup> , Marco Gribaudo <sup>6</sup> , Mohamed Kaaniche <sup>4</sup> , Paolo Lollini <sup>3</sup> , Francesco Romani <sup>3</sup> , Jeremy Sproston <sup>6</sup> ,	
<b>Participant(s):</b> <sup>6</sup> CNIT, <sup>3</sup> CNR-ISTI, <sup>4</sup> LAAS-CNRS,	
<b>Work package contributing to the deliverable:</b> WP5	
<b>Nature:</b>	<b>R</b>
<b>Dissemination level:</b>	<b>PU</b>
<b>Version:</b>	<b>3.0</b>
<b>Total number of pages:</b>	

### Abstract

This deliverable reports on the activities for the set-up of the modelling environments for the evaluation activities of WP5. To this objective, it reports on the identified modelling peculiarities of the electric power infrastructure and the information infrastructures and of their interdependencies, recalls the tools that have been considered and concentrates on the tools that are, and will be, used in the project: DrawNET, DEEM and EPSys which have been developed before and during the project by the partners, and Möbius and PRISM, developed respectively at the University of Illinois at Urbana Champaign and at the University of Birmingham (and recently at the University of Oxford).

**Keyword list:** modelling, performance evaluation tools, dependability tools, statistical analysis

**DOCUMENT HISTORY**

Date	Version	Status	Comments
29/10/2007	000	Draft	First version of table of contents and preliminary description of tools
14/12/2007	001	draft	Contributions from partners included
18/12/2007	001	draft	document circulated
16/01/2008	002	draft	comments from partners included, introductory and conclusive parts included
17/01/2008	003	submitted	final comments from partner included

## Table of Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>4</b>
1.1	Modelling peculiarities of information infrastructure II and electrical infrastructure EI. ....	4
1.1.1	<i>Hierarchical nature of the II structure .....</i>	4
1.1.2	<i>Complexity of the EI structure .....</i>	5
1.1.3	<i>EI failures and internal propagations.....</i>	5
1.1.4	<i>II failures and internal propagations .....</i>	5
1.1.5	<i>II <math>\leftrightarrow</math> EI (interdependencies).....</i>	6
1.2	Hierarchies, compositionality and multiformalism. ....	6
1.2.1	<i>Modelling power.....</i>	7
1.2.2	<i>Modelling efficiency .....</i>	7
1.2.3	<i>Solution power.....</i>	7
1.3	Parameters estimation, validation of assumptions.....	8
<b>2</b>	<b>TOOLS THAT SUPPORT CRUTIAL EVALUATION ACTIVITIES.....</b>	<b>8</b>
2.1	Overview .....	8
2.2	Möbius .....	9
2.2.1	<i>Modelling formalisms.....</i>	10
2.2.1.1	The Stochastic Activity Network formalism .....	10
2.2.1.1.1	Completion rules .....	10
2.2.1.1.2	Measures definition .....	11
2.2.1.2	The “connection model” formalism.....	11
2.2.2	<i>Solution methods and experiment definition .....</i>	11
2.3	The Draw-NET modelling system.....	11
2.3.1	<i>Defining formalisms in the DMS .....</i>	12
2.3.2	<i>Building models in the DMS .....</i>	14
2.4	PRISM – .....	16
2.4.1	<i>System description language .....</i>	16
2.4.2	<i>System evaluation and temporal logic properties.....</i>	17
2.5	DEEM .....	18
2.5.1	<i>Modelling formalism and approach.....</i>	18
2.5.2	<i>Dependability measures and transient analysis.....</i>	19
2.5.3	<i>Solution methods.....</i>	20
2.6	EPSyS .....	20
2.6.1	<i>Architecture of EPSyS.....</i>	20
2.6.2	<i>The coordination: EQ.....</i>	21
2.6.3	<i>The EI Part: ECM, PM, TM.....</i>	22
2.6.4	<i>The II part: IIM1, IIM2 and the state change functions.....</i>	23
2.6.5	<i>Evaluating measures: MOIM .....</i>	25
2.6.6	<i>The simulation procedure .....</i>	25
2.6.7	<i>Availability of the tool.....</i>	26
<b>3</b>	<b>DISTRIBUTION FITTING AND STATISTICAL ANALYSIS OF EVENTS.....</b>	<b>26</b>
3.1	Data pre-processing and categorization.....	26
3.2	Statistical inference and model validation .....	27
<b>4</b>	<b>CONCLUSIONS .....</b>	<b>29</b>
	<b>REFERENCES.....</b>	<b>30</b>

# 1 INTRODUCTION

This deliverable reports on the activities carried on by the consortium to select a set of appropriate tools to be used (and being currently used) for the evaluation activities of WP5. This deliverable is organized as follows: Section 1 discusses the modelling peculiarities of the systems evaluated in CRUTIAL, Section 2 reports on tools used for model based analysis, while Section 3 reports on techniques and tools for statistical analysis of collected data, which allow to provide statistically realistic data to the models used for the evaluation.

This introduction identifies the modelling peculiarities of the considered infrastructures, in terms of structure, complexity, propagations, and proceeds in translating these peculiarities into tools and formalisms requirements. The deliverable uses the following abbreviations: EI for Electrical Infrastructure, II for the Information Infrastructure (monitor, control and communication support) and EPS for Electrical Power System, the composition of the two infrastructures.

## 1.1 Modelling peculiarities of information infrastructure II and electrical infrastructure EI.

We have found a number of interesting features in our systems and in the corresponding models, that we discuss one by one.

### 1.1.1 Hierarchical nature of the II structure

The II system has a natural hierarchical structure. We can distinguish three main levels of control corresponding to different levels of criticality and locality of the decisions.

The local II components (Local Control System - LCS) control and guarantee the correct operation of a substation and reconfigure the substation in case of breakdown of some apparatus. Therefore, their decisions can affect the single substations only (local decisions, with low criticality).

The regional II components (Regional Telecontrol System – RTS) monitor the set of substations in their region, in order to diagnose faults on the power lines. In case of breakdowns, they choose the more suitable corrective actions to restore the functionality of the grid (regional decisions, with high criticality). Since these regional components are not directly connected to the substations, the corrective actions to adopt are communicated to the local components of reference.

Finally, the national II component (National Telecontrol System – NTS) has the main function of supervising the entire grid and handling the planning of medium and long term operations. It also assists the regional components to localize breakdowns on the power lines situated between two regions (or between two areas).

A complete description of the II structure has been provided in D2 [Donatelli 2008a], Donatelli et al, "Model-based evaluation of the middleware services and protocols & architectural patterns ", CRUTIAL project, Work Package 5, Deliverable D25, January 2008.

[Daly *et al.* 2000] D. Daly, D. D. Deavours, P. G. Webster and W. H. Sanders, "Möbius: An extensible tool for performance and dependability modeling", in *11th International Conference, TOOLS 2000*, (H. C. B. B. R. Haverkort, and C. U. Smith, Ed.), (Schaumburg, IL, USA), pp.332-336, Lecture Notes in Computer Science, 2000.

[Franceschinis *et al.* 2002] G. Franceschinis, M. Gribaudo, M. Iacono, N. Mazzocca and V. Vittorini, "Drawnet++: Model objects to support performance analysis and simulation of complex systems", in *Lecture Notes in Computer Science LNCS 2324*, pp.233-238, Springer-Verlag, 2002.

[Fricks *et al.* 1997] R. Fricks, C. Hirel, S. Wells and K. Trivedi, "The development of an integrated modeling environment", in *World Congress on Systems Simulation (WCSS '97)*, (Singapore), pp.471-476, 1997.

[German *et al.* 1995a] R. German, C. Kelling, A. Zimmermann and G. Hommel, "Timenet: A toolkit for evaluating non-markovian stochastic petri-nets", *Performance Evaluation*, 24, pp.69-87, 1995a.

[Garrone *et al.* 2007].

### 1.1.2 Complexity of the EI structure

The EI infrastructure is composed by a large number of components (electrical equipments), whose behaviours are strictly correlated each other. From a high-level point of view, EI is logically structured in different components: the transmission grid (TG), the distribution grid (DG), the high voltage generation plants (HG), the medium and low voltage generation plants (LG), the high voltage loads (HL) and the medium and low voltage loads (LL).

The transmission grid and the distribution grid can be considered like a network, or a graph: the nodes of the graph represent the substations, while the arcs represent the power lines. The generators and the loads are nodes connected by arcs (power lines) to the nodes of the grid (the substations). Some nodes of the grid can be connected to nodes of the contiguous grid.

In turn, each node is composed by a set of electrical equipments like transformers, bus-bars, breakers, switches, power lines and protections (see D3 [Kaâniche *et al.* 2007] for more details).

### 1.1.3 EI failures and internal propagations

Concerning the EI failures and their propagation inside EI, we identify the following peculiarities:

- The time to failures of the electrical equipments composing the EI depends on the component (e.g., the length of the line, the position of the station, etc.), on the value of the electrical parameters associated to the components (e.g., an overloaded power line has a lower time to failure than a non-overloaded one) and on the type of disruption it can be affected (lightning, tree fall, etc.).
- An EI failure of a component can propagate to contiguous components (e.g., a lightning on a power line propagates through the connected substations if the respective protections do not open), and the propagation time should not be considered instantaneous.
- Protections can stop the propagation of an EI failure by isolating the affected components from the grid. The activation time of a protection should not be considered instantaneous. The correct activation of a protection depends on its state (working or failed), on the strength of the disruption (e.g., the power of the lightning) and on the value of the electrical parameters associated to the protection component.

### 1.1.4 II failures and internal propagations

Concerning the II failures and their propagation inside II, we identify the following peculiarities:

- The failures of the II components can be summarized in (transient and permanent) omission failure, time failure, value failure and byzantine failure. In general, the failure of the components LCS, RTS and NTS may depend on the failures of the components connected to them through a network.
- Failures of some LCS components can impact on the input values that the

component RTS receives from LCSs. These values can be omitted, delayed (or anticipated) or erroneous. Since reconfigurations required by RTS (or NTS) are actuated by the associated component LCS, a failure of a component LCS can also impact on the reconfigurations required by RTS (or NTS).

- The failure of the component RTS (or NTS) corresponds to an erroneous (request of) reconfiguration of the state of EI (including an unneeded reconfiguration) affecting one or more components of the controlled region.
- The reaction time (with respect to the occurrence of an EI failure), the failure time and the erroneous activation time (when no EI failures have occurred) of an II component (e.g., LTS, RTS and NTS) should be considered.
- The functions that implement the reconfiguration and regulation algorithms should be considered. These functions activate when EI is not in equilibrium; in such conditions, they receive as input the current topology and the electrical values associated to each electrical equipment, and produce as outputs a new topology with a new setting of electrical parameters which allows the equilibrium condition to be restored (that is, EI is back to the NORMAL state), if possible.

### 1.1.5 II ↔ EI (interdependencies)

II→EI - cyber interdependency: when the state of the EI infrastructure depends on information transmitted through the II infrastructure.

- Failures in II impact on the state of EI depending on the components affected by the failures, and obviously by the type of the failures.
- The consequences of a failure of the component LCS associated to a substation can be, for example:
  - Omission failure of LCS, fail silent LCS → No (reconfiguration) actions are performed on EI, or undesired reconfiguration actions are erroneously performed.
  - Time failure of LCS → The reconfiguration actions on EI are performed after a certain delay (or before the instant of time they are required).
  - Value failure of LCS → Incorrect closing or opening of the protections directly connected to the failed component is performed.
- The effect of the failure of RTS (or NTS) on a substation is the same as the failure of the component LCS associated to the substation. In the case of Byzantine failure these effects can be different for each substation.
- The effect of a RTS failure on the global EPS may be catastrophic because potentially it may affect the all substations located within the region, therefore causing disservices to a huge number of loads and customers.

EI→II - physical interdependency: when the state of II infrastructure is dependent on the material outputs of the EI (the electricity).

- EI failures can impact on (parts of) the II system by lessening its functionalities (till complete failure in the extreme case the disruption is a total blackout of the power grid).

## 1.2 Hierarchies, compositionality and multiformalism.

To capture the EI and II peculiarities discussed in Section 1.1, the modelling and evaluation framework should possess the following major characteristics, grouped into three categories: *modelling power* aspects (the basic modelling mechanisms required to build the EPS model),

the *modelling efficiency* aspects (the advanced modelling mechanisms required to build the EPS model more efficiently), and the *solution power* aspects.

### 1.2.1 Modelling power.

- EPS is a very complex system having subsystems (or subcomponents) with very different characteristics. This heterogeneity must also be addressed inside the framework that, therefore, should support the definition of different models using different formalisms, each one capable to properly capture the behaviour of a specific subsystem.
- The framework should support the modelling of behaviours and interdependencies in a context where the modelled infrastructures have different operation phases and regimes with different configurations and characteristics.
- The framework should support the representation of both continuous and discrete states. The continuous states are needed to represent the real values associated to the electrical parameters.
- The framework should support time and probability distributions, and conditions enabling the time consuming events (e.g., for the activation of a local protection) that can depend both on the discrete and on the continuous state.
- The framework should support the call to the functions that implement both the II reconfiguration and control algorithms and the EI automatic evolution in case of events changing the electrical equilibrium.
- Risk analysis of EPS based on a stochastic approach requires the definition of measures of performability, which is a unified measure proposed to deal simultaneously with performance and dependability. To this purpose, a reward structure can be set-up by associating proper costs/benefits to generators/loads and interruption of service supply.

### 1.2.2 Modelling efficiency

- As detailed in Section 1.1.1, the system has a hierarchical structure; the modelling framework should support hierarchical composition of different sub-models.
- The model for the overall EPS could be facilitated considering replication of sub-models, with and without identification of each single replica. The replicated and composed models should share part of the state (common state).
- Compact representation for the topology of the grid, for example, describing a part of the state of the system in terms of a matrix (incidence matrix [nodes x arcs]).
- Compact representation of continuous state (for the electrical parameters), for example, describing a part of the state of the system in term of arrays, associating to each component of the EI grid (nodes and arcs) the values of the electrical parameters.

### 1.2.3 Solution power

- The framework should support analytical solution of the overall model (if possible). Actually, the huge system complexity and its heterogeneity may inhibit the computation of the analytic solution due to the explosion of the states of the model and stiffness. Moreover, an analytical solution method could not exist for the class of models considered, depending on the considered time distributions. Analytical solutions may be applied for simpler sub-models.
- The framework should support simulation, either using general purpose simulation

tools or ad hoc simulation software.

- The framework should support separate evaluation of different sub-models (with analytic or simulation techniques) and combination of the results.

### 1.3 Parameters estimation, validation of assumptions

Analytical and simulation based evaluation models incorporate parameters and assumptions that need to be estimated and validated based on real data. Such parameters correspond for example to the occurrence rates and the distributions associated to components failures and recoveries, or probabilities characterizing the efficiency of error detection and processing mechanisms. Data can be collected from the field or during controlled experiments. Statistical processing methodologies are generally used to process the collected data and estimate the needed parameters. Both accidental and malicious faults need to be addressed. Section 3 of this deliverable outlines the general methodology that is usually applied to process the collected data and to estimate the statistical properties of the parameters under investigation.

## 2 TOOLS THAT SUPPORT CRUCIAL EVALUATION ACTIVITIES

### 2.1 Overview .

We have already analyzed a number of tools that support some of the feature listed above as part of the WP2 activities of the first year. We provide here a summary of those tools before proceeding to the detailed description of the tools used in WP5.

We can group the tools considering to whether they are meant and designed for a single (or multiple) formalism, and whether they provide a single or a multiple set of solution methods, considering that most tools provide more than one solution method (typically some analytical solution and simulation), and that it is considered an added value to be able to mix various solution methods in a single solution process, so that each component can be solved with the most adequate technique.

Single-formalism/multi-solution tools are built around a single formalism and one or more solution techniques. They are very useful inside a specific domain, but their major limitation is that all parts of a model must be built in the single formalism supported by the tool.

DSPNexpress [Lindemann *et al.* 1999] , GreatSPN Chiola *et al.* 1995] , [Howard1971] R. A. Howard. Dynamic Probabilistic Systems, volume II: Semi-Markov and Decision Processes. Wiley, New York, 1971

[Illié *et al.* 2004] , SURF-2 [Béounes *et al.* 1993], DEEM [Bondavalli *et al.* 2000], TimeNET [German *et al.* 1995a], UltraSAN [Sanders & Meyer 2001], and HiQPN [HiQPN] are only some examples of tools based on Stochastic Petri Nets formalism and its extensions. They all provide analytic/numerical solution of a generated state-level representation and, in some cases, support simulation-based solution as well. In particular, GreatSPN tool includes the *algebra* software package that implements the composition of GSPN models as well as of their colored extension (Stochastic Well Formed Net - SWN) [Bernardi *et al.* 2001].

Multi-formalism/multi-solution tools can be further classified according to the integration support they provide.

A first set includes tools that try to unify several different single-formalism modelling tools into a unique software environment (loose integration). Examples are IMSE (Integrated Modelling Support Environment) **Errore. L'origine riferimento non è stata trovata.**, that is a support environment that contains tools for modelling, workload analysis, and system specification, IDEAS (Integrated Design Environment for Assessment of Computer Systems and



Communication Networks) [Fricks *et al.* 1997] , that provides a broad range of modelling capabilities without the need to learn multiple interface languages and output formats, and FREUD [van Moorsel & Huang 1998], that focuses on providing a uniform interface to a variety of web-enabled tools. Another tool that includes different formalisms is PRISM [Hinton *et al.* 2006] which offer a support for compositionality, but does not allow to specify a model using multiple formalisms. A more recent model design framework is the DrawNet Modelling System (DMS) [Franceschinis *et al.* 2002], [Codetta-Raiteri *et al.* 2006] that supports the definition of new formalisms and provides a support for multiformalism and multisolution.

A second set of multi-formalism/multi-solution includes tools in which the various formalisms, composition operators and solvers are actually implemented within a unique comprehensive tool (tight integration). Among the existing tools having these characteristics, we cite SHARPE [Schneeweiss 1999] W. G. Schneeweiss, "The Fault Tree Method", LiLoLe Verlag, 1999.

[Trivedi 2002], that is a tool for specifying and analyzing performance, reliability and performability models, SMART [Ciardo & Miner 1996], that is a multi-formalism modelling tool to study complex discrete-state systems, DEDS [Alur *et al.* 1999] R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7-48, 1999.

[Bause *et al.* 1998] , that is a toolbox for the construction of modular tools for functional and quantitative analysis of discrete event dynamic systems, and POEMS [Adve *et al.* 2000] , that is a tool for modelling complex parallel and distributed systems. Möbius [Daly 2001] is another important multi-formalism/multi-solution tool. It supports different formalisms like SAN, PEPA, Buckets and Balls, and Fault Tree, as well as the composition of models described using different formalisms. Concerning the solution process, it supports both state-based, analytical/numerical techniques (when applicable) and discrete event simulation (both transient and steady-state, applicable to any model).

Of the above mentioned tools we have currently using in the project Möbius, DEEM, DrawNET, and PRISM, that we shall describe in more detail in the following subsection, and an ad-hoc simulator of EPS that has been built specifically for the CRUTIAL needs, EPSys.

The criteria that have driven this choice are a mix of the adequacy of the tools with respect to the listed requirements, of their availability, reliability and cost, and of the personal preferences of the team involved. Although this may not sound as a scientific motivation, actually it is: indeed the learning curve of rather complex and articulated tools as the one listed above is quite steep and even if you are willing to learn it, the time it takes to master the tool to a level of expertise that allows to obtain the best out of it (especially in terms of solution efficiency) is not only a matter of time and cost but of expertise (for example it strongly depends upon the number of large case studies you have conducted using that tool). It should therefore not surprise if the various partners are using the tools they have more expertise upon in all those situations in which this choice was feasible.

The partners (in particular CNR-ISTI) has also developed the new tool EPSys, not because EPS simulators do not exist, but because of their cost and complexity. Actually, in addition to the difficulties in finding a mature, well assisted and easy to customize simulator for EPS systems, we decided to build our EPSyS simulator since our aim was primarily to use it in conjunction with the modelling framework under investigation in WP2, for reciprocal benefits. Therefore, we needed to have the two evaluation methods sharing the same common knowledge and representation of the stochastic models capturing the behaviours and dynamics of the entities composing the EPS system.

## 2.2 Möbius

Möbius [Clark *et al.* 2001] is a multi-formalism multi-solution tool that allows us not only to

build models using different formalisms (such as SAN, PEPA, Buckets and Balls, and Fault Tree), but also to compose them obtaining a single multi-formalism model, that is a model composed by submodels described using different formalisms.

### 2.2.1 Modelling formalisms

Among the available modelling formalisms, here we provide further details on two formalisms implemented in Möbius that can support some of the tool requirements listed in Section 1.1: the SAN formalism and the “connection model” formalism.

#### 2.2.1.1 The Stochastic Activity Network formalism

Stochastic Activity Networks (SANs) were first introduced in [Movaghar & Meyer 1984] and then formally defined in [Sanders & Meyer 2001]. The formalism is a generalization of Stochastic Petri Nets, and has some similarities to the GSPN formalism. The building blocks composing a SAN are *places*, *activities*, *arcs*, *input gates* and *output gates*.

*Places* in SANs have the same interpretation as in Petri Nets: they hold tokens, the number of tokens in a place is called the marking of that place, and the marking of the SAN is the vector containing the marking of all the places.

There are two types of *activities*, timed and instantaneous. Timed activities are used to represent delays in the system that affect the measure of interest, while instantaneous activities are used to abstract delays deemed insignificant relative to the measures of interest. Uncertainty about the length of the delay represented by a timed activity is described by a continuous probability distribution function, called the “activity time distribution function”, that can be a generally distributed random variables, and each distribution can depend on the marking of the network. Activities can have cases. Cases are used to represent uncertainty about the action taken upon completion of an activity.

Gates connect activities and places. *Input gates* are connected to one or more places and one single activity. They have a predicate, a boolean function of the markings of the connected places, and an output function. When the predicate is true, the gate holds. *Output gates* are connected to one or more places, and the output side of an activity. If the activity has more than one case, output gates are connected to a single case. Output gates have only an output function. Gate functions (both for input and output gates) provide flexibility in defining how the markings of connected places change when the delay represented by an activity expires.

*Arcs* in SANs are default gates, defined to duplicate the behaviour of arcs in Petri nets. Thus, arcs are directed. Each arc connects a single place and a single activity. The arc is an input arc if it is drawn from a place to an activity. An output arc is drawn from an activity to a place. An input arc holds if there is at least one token in the connected place. The function of an input arc removes a token from the connected place, while the function of an output arc adds a token to the connected place. An activity is thus enabled only when i) all of its input gates hold, and ii) all of its input arcs hold.

In the following we give some further details on two particularly important aspects: the execution of a timed activity, and the measure specifications.

##### 2.2.1.1.1 Completion rules

When an activity becomes enabled, it is activated, and the time between activation and the scheduled completion of an activity, called the activity time, is sampled from the activity time distribution. Upon completion of an activity, the following events take place: i) if the activity has cases, a case is (probabilistically) chosen; ii) the functions of all the connected input gates are executed; iii) tokens are removed from places connected by input arcs; iv) the functions of all the output gates connected to the chosen case are executed; v) tokens are added to places that are connected by output arcs to the chosen case. An activity is aborted

when the SAN moves into a new stable marking in which at least one input gate no longer holds.

### 2.2.1.1.2 Measures definition

Upon completing the model of the system, a modeller has to specify the measures in terms of the model. In the SAN modelling framework, the measures are specified in terms of reward variables [Sanders & Meyer 1991]. Let  $R(m)$  be the rate at which the reward accumulated in state  $m$ , and let  $C(a)$  be the reward earned upon completion of transition  $a$ . If  $\{X_t, t > 0\}$  is the modelled stochastic process and  $M$  the set of all possible states, a reward variable collected at an instant of time conventionally denoted by  $V_t$  is informally defined as

$$V_t = \sum_m R(m)P(X_t = m) + \sum_a C(a)I_t^a,$$

where  $I_t^a$  is the indicator of the event that  $a$  was the activity that completed to bring the SAN into the marking observed at time  $t$ . The steady-state reward can be obtained as  $V_{t \rightarrow \infty}$ . In the case in which the reward variable is evaluated considering an interval of time  $[t, t+l]$ , then the accumulated reward is related both to the number of times each activity completes and to the time spent in a particular marking during the interval. More precisely,

$$Y_{[t, t+l]} = \sum_m R(m)J_{[t, t+l]}^m + \sum_a C(a)N_{[t, t+l]}^a,$$

where  $J_{[t, t+l]}^m$  is a random variable representing the total time that the SAN is in the marking  $m$  during  $[t, t+l]$  and  $N_{[t, t+l]}^a$  is a random variable representing the number of completions of activity  $a$  during  $[t, t+l]$ .

### 2.2.1.2 The “connection model” formalism

The connection model formalism ([Christensen 2001, Daly 2001]) enables the passing of results between models, using a database to collect and share the produced results.

A connected model is a graph consisting of a set of model nodes (called “solvable models”), arcs (called “conduits”), and transformation nodes (called “connection functions”). A solvable model is a model with a specified reward structure that can be solved, a conduit transfers results, while a connection function takes a set of results to generate an input parameter for another model. The connection graph is traversed in a certain order. First a solvable model is solved, and it writes the results to a results database. The output conduits for that solvable model transfer specific results from the model to a connection function. Then the connection function performs some mathematical transformation on all the results passed to it, and a conduit provides this value to another solvable model. The value is passed into the solvable model via a global variable.

## 2.2.2 Solution methods and experiment definition

Möbius currently supports two classes of solution techniques: discrete event simulation (both transient and steady-state, applicable to any model), and state-based, analytical/numerical techniques. The simulator may be executed on a single workstation, or distributed on a network of workstations.

## 2.3 The Draw-NET modelling system

The *Draw-Net Modelling System* (DMS) [Gribaudo et al. 2005] is a framework offering multiformalism and multisolution and it is characterized by an open architecture such that the DMS can be customized for the design and the solution of models conforming to new graph based formalisms. Further details about the DMS architectural and formal aspects can be found in [Codetta-Raiteri et al. 2006].

The *multi-formalism* support in DMS allow the use, in a single model, of different modelling formalisms that are used to represent in the most suitable way different aspects of the system. DMS also support *multi-solution* in the sense of a combined use of different solvers

to perform the analysis of the model.

Moreover DMS allows for the definition of new formalisms, and new composed formalisms in an easy manner, a feature that we shall exploit for providing a tool support for the Dependent Automata formalism defined in WP2

### 2.3.1 Defining formalisms in the DMS

In the DMS, formalisms can be classified as *Simple formalisms*, *Derived formalisms* and *Composed formalisms*.

A simple formalism defines the primitives of a graph based model whose elements can only be nodes and edges, with no submodels. At the same time, a simple formalism does not inherit any element from a parent formalism (inter-formalism inheritance), except from *GraphBased* which provides the basic elements of a graph. Figure 1 shows the elements of the Fault Tree (FT) [Schneeweiss 1999] simple formalism, using a UML-like graphic language where each box indicates the name of an element, its properties and the results computable on it.

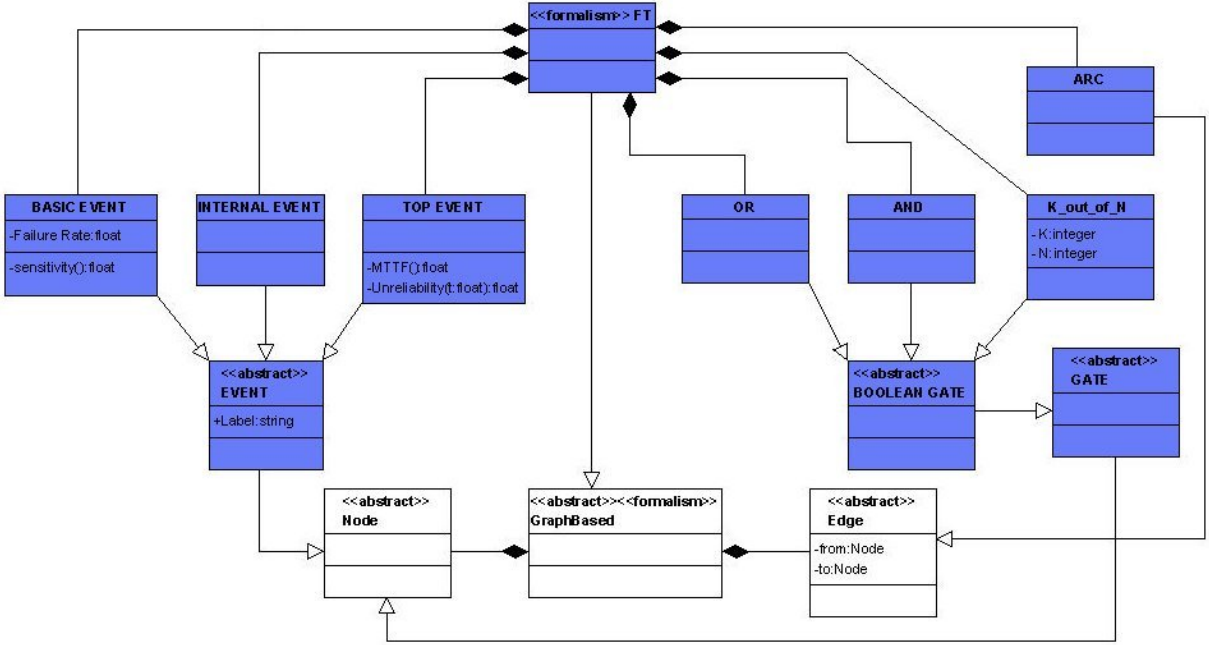
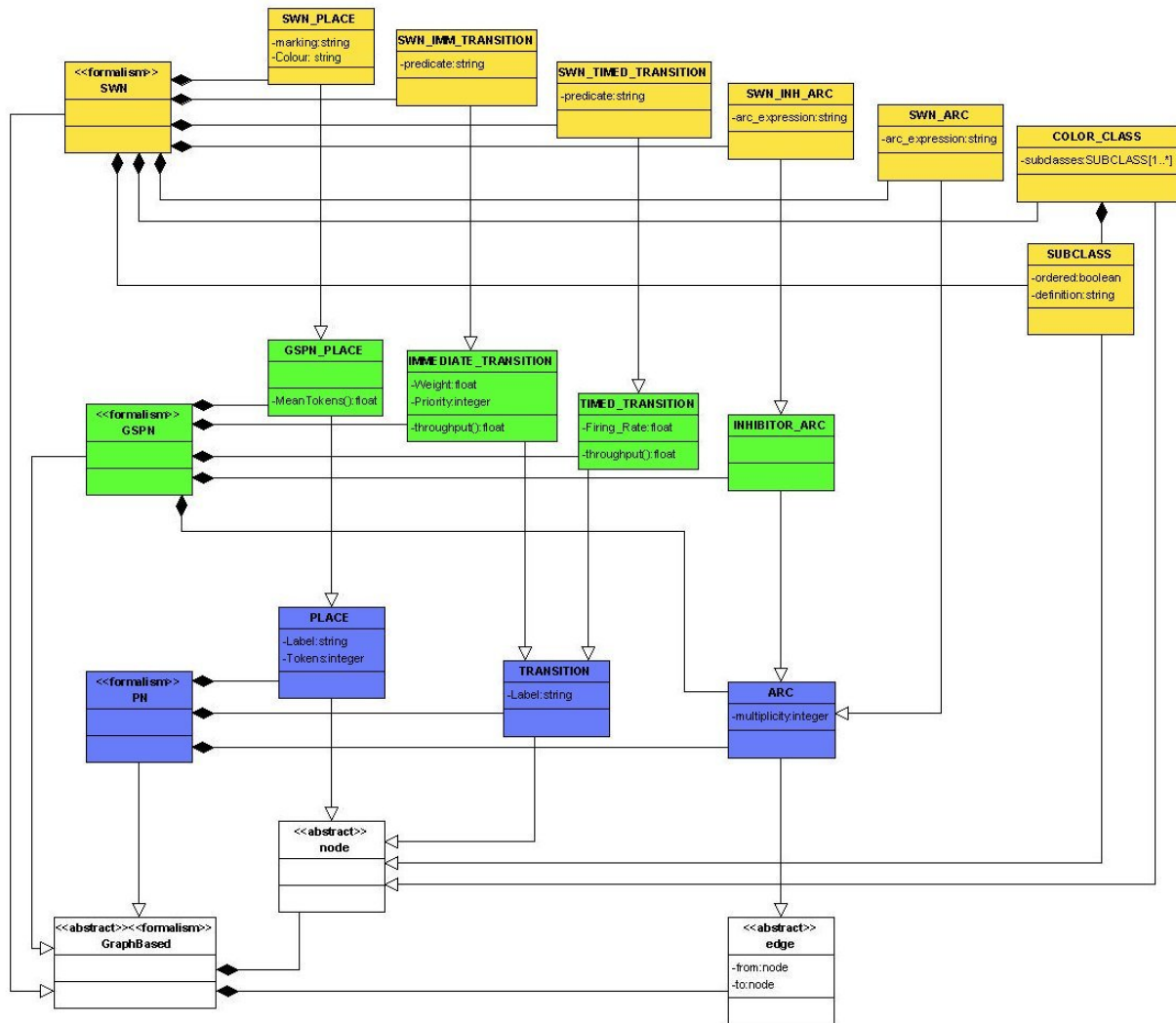


Figure 1: UML-like diagram of the abstract representation of the FT formalism

Derived formalisms are the result of the application of inter-formalism inheritance; this means that some of the elements of a derived formalism are inherited from another formalism (simple or derived), or that some of the elements of a derived formalism are defined by extending the elements of another formalism. A case of derived formalism is SWN (Stochastic coloured Well-formed Nets [Chiola et al. 1993]) derived from the GSPN (Generalized Stochastic Petri Nets [Ajmone-Marsan et al. 1995]) formalism which is in turn derived from the PN (Petri Nets) formalism (Figure 2).



**Figure 2: UML-like diagram of the abstract representation of the SWN formalism**

In a multi-formalism model, we usually have a container model and a set of submodels; the container model is an higher level model whose aim consists of containing several submodels, and defining how each submodel interacts with the others. The elements to build a container model have to be defined in a *Composed formalism*. A composed formalism includes several simple or derived formalisms, together with a set of elements necessary to connect submodels. Figure 3.a shows an example of composed formalism called FT+SWN and including the FT and the SWN formalism.

*DNForGe*, the DMS formalism editor, allows the user to define formalisms of any kind (simple, derived, composed). Figure 3.b shows a screenshot of *DNForGe*.

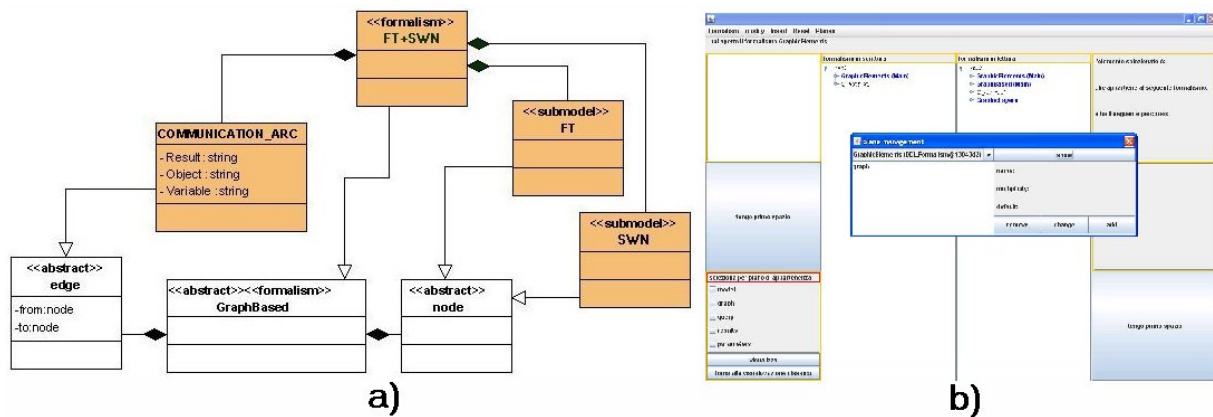


Figure 3: a) UML-like diagram of FT+SWN b) Screenshot of DNFForGe

### 2.3.2 Building models in the DMS

In this section, the system under study refers to the control system scenario n. 1 described in Deliverable D2 [Garrone et al. 2007] and concerning the Teleoperation function performed between a DSO control centre and a substation, by means of a communication network. In this situation, an attacker may attempt a denial of service (DoS) attack to the communication network between the control centre and the substation. A DoS attack may last for a random period of time, and it may be blocked by the success of a possible countermeasure (firewalling, traffic monitoring, etc.).

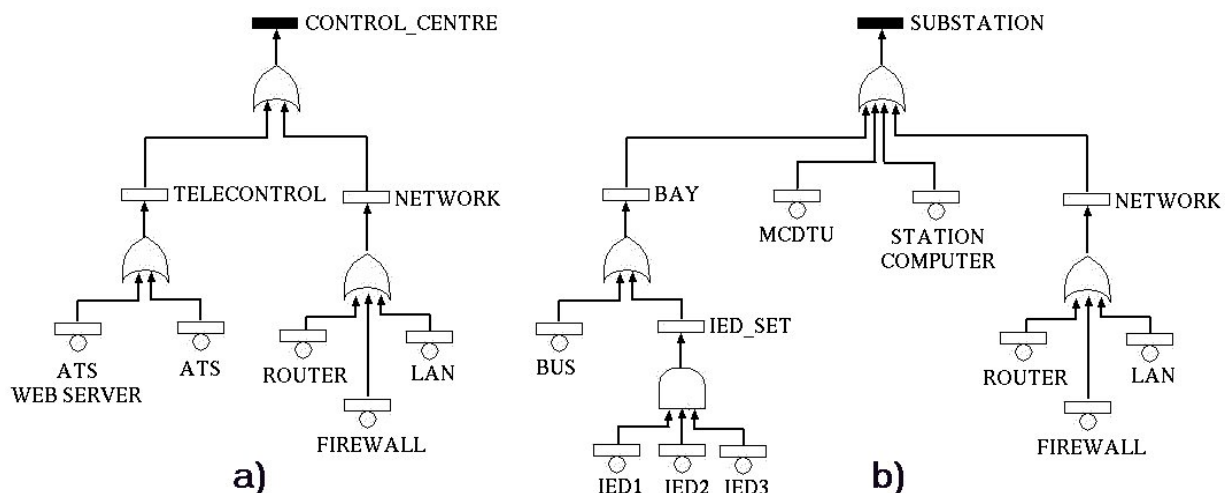


Figure 4: a) Submodel FT1 b) Submodel FT2

In this section, we provide the multi-formalism model representing the behaviour and the failure mode of such system. The model is composed by a container model (Figure 6) conforming to the FT+SWN formalism, and three submodels: FT1 (Figure 4.a) and FT2 (Figure 4.b) are conforming to the FT formalism and they model the failure mode of the control centre and the failure mode of the substation respectively; the submodel SWN0 (Figure 5) is conforming to the SWN formalism and models the generation of requests by the control centre, their transmission on the communication network, their processing by the substation, the dispatch of the replies from the substation to the control centre, and the possible DoS attack to the communication network. The container model (Figure 6) indicates the exchange of results among the submodels. Such a multi-formalism model has been built by means of the *Draw-Net tool*, the DMS model editor

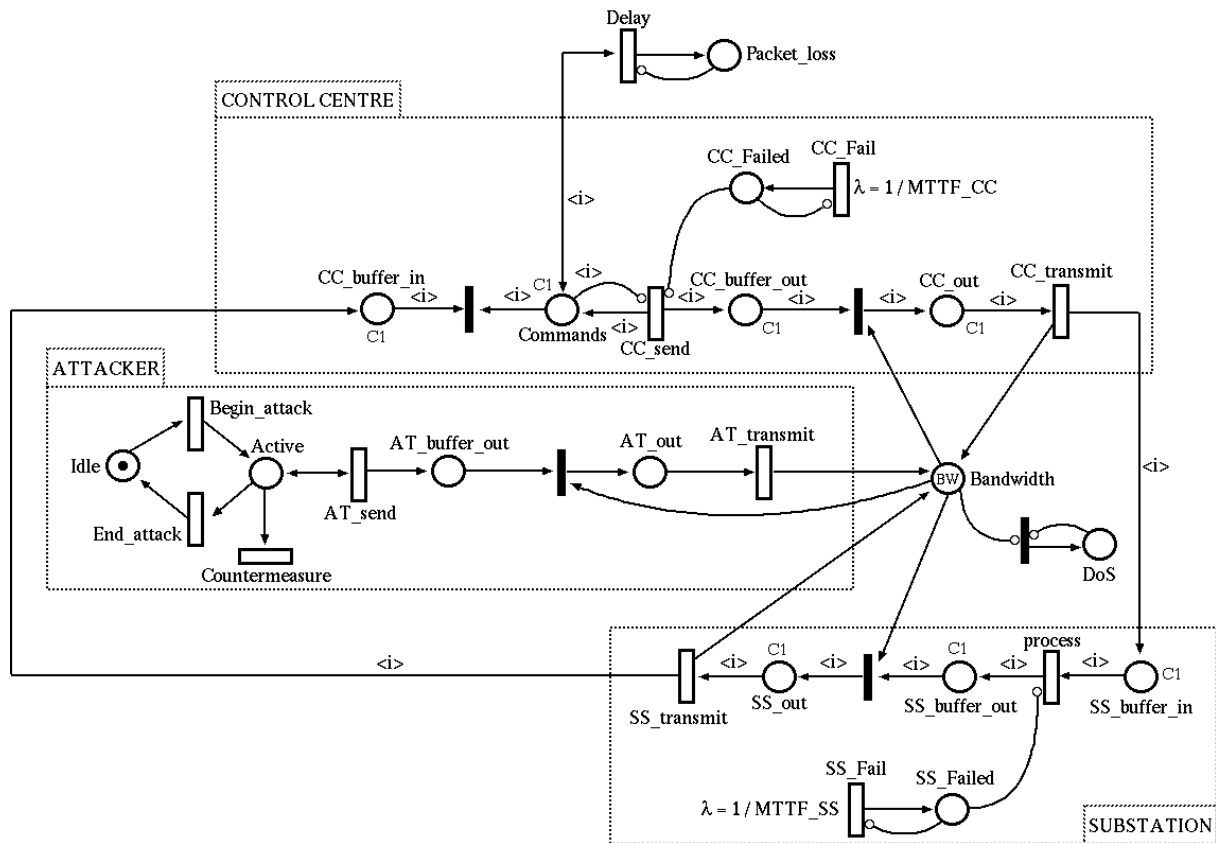
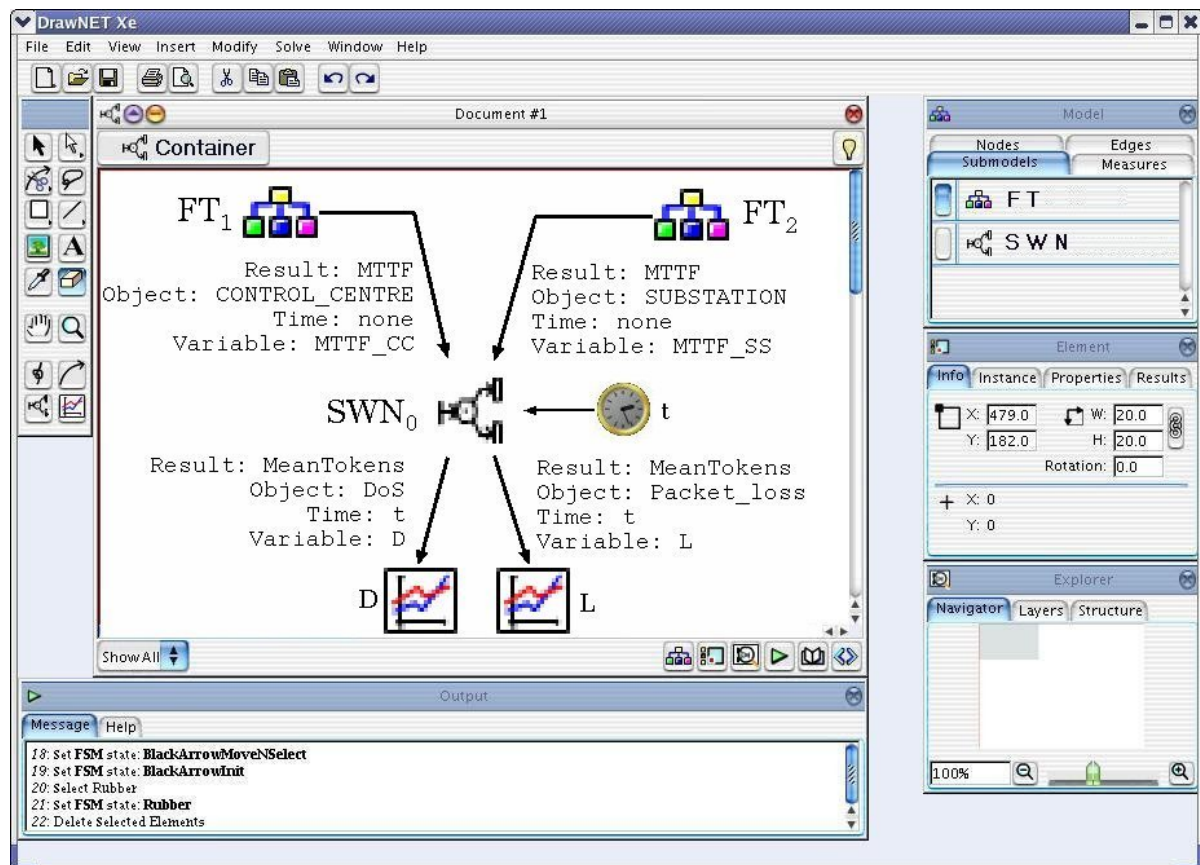


Figure 5: Submodel SWN0



## Figure 6: Container model

### 2.4 PRISM –

PRISM is a tool for the automatic verification of probabilistic systems using model-checking techniques [Hinton et al. 2006]. Model checking is a well-established method for the verification of a formally-defined system against a formally-defined set of properties. Typically in model checking, finite-state systems with nondeterministic transitions are considered (that is, without probabilistic information regarding the relative likelihood of transitions), which are modelled in high-level languages such as Petri nets, process algebras, or Reactive Modules. Properties are usually represented by temporal logic formulae. The aim of a model-checking algorithm is to determine automatically whether the model of the system satisfies the model of the property.

Model-checking analyses depend critically on the size of the state space of the system, and therefore suffer from a similar “curse of dimensionality” problem as that seen in performance evaluation. In the field of model checking, the exponential blow-up in the size of the underlying transition system compared to the size of the more concise high-level description used by the system modeller is called the state-space explosion problem. Techniques have been developed to reduce the effect of the state-space explosion problem on the resources (time and space) used by model-checking algorithms: one such technique is the use of data structures based on Binary Decision Diagrams for the implicit representation of the system’s transition relation and the state sets computed during the execution of the algorithm. Such implicit representations give rise to symbolic model-checking algorithms. The underlying verification engine of PRISM is based on an extension of Binary Decision Diagrams, thereby permitting symbolic model checking of probabilistic systems.

PRISM can be considered as a multiformalism tool since it allows the verification of systems whose underlying stochastic process is either a discrete-time Markov chain, a continuous-time Markov chain, or a Markov decision process (a process in which nondeterministic and probabilistic choice can coexist). Note that the input language for the three cases is very similar, but, although compositionality is at the heart of the specification language, components have to be specified in the same formalism.

#### 2.4.1 System description language

The system description language of PRISM is based on the Reactive Modules formalism

[GMP] The GNU Multi Precision arithmetic library: <http://gmplib>. The language consists of a finite number of interacting modules. Each module consists of a set of variables with a finite domain, and the state of the module is given by a valuation of the variables. The module also comprises a number of guarded commands defining the behaviour of the module. Each such command comprises the following elements:

- A predicate over the system’s variables (the *guard* of the command), which describes the set of states from which the command can be taken.
- A set of *updates*, which determine the modifications made to the values of the module variables in order to result in the target state after execution of the command, and which have each an associated probability or rate of occurrence.

Furthermore, guarded commands can be associated with actions, which can be used to synchronize with other modules, as in process algebras. The overall system is then obtained as a set of modules, with the global state of the overall system comprising the current local state of each of the individual modules. Transitions of the overall system are obtained from transitions of individual modules, or, if modules can synchronize on shared actions, by transitions of more than one module.



As an example of the manner in which the PRISM modelling language can be used, we can consider the example of a simple communication protocol: the model could consist of sender and receiver modules, and a single module representing the medium. The sender module may interact with the medium module by synchronizing on the shared action *send*, whereas the receiver module may interact with the medium module synchronizing on the shared action *receive*. The medium module may consist of one Boolean valued variable set to 1 when a message is being sent on the medium, and to 0 when the medium is free. The sender and receiver modules may comprise a number of variables, such as those representing the current control state, others representing the number of messages in a buffer, and so on.

We note that PRISM models may take the form of discrete-time Markov chains, continuous-time Markov chains, or Markov decision processes (in which nondeterministic and probabilistic choice can coexist).

#### 2.4.2 System evaluation and temporal logic properties

In this section, for simplicity, we consider the case in which the model is represented as a continuous-time Markov chain, noting that most of the functionality of PRISM described here is also available for discrete-time Markov chain and Markov decision process models. PRISM offers a number of methods for the analysis of the model:

- PRISM automatically tests for the presence of reachable deadlock states during the construction of the BDD-based internal representation of the model;
- the user has the possibility to generate and visualize graphically sample paths of a PRISM model, which can be useful for understanding and debugging a model during its development;
- steady-state and transient probabilities can be computed;
- model checking of temporal logic properties can be employed.

In particular we consider the final item of this list. The advantage of temporal logics in the context of formal verification is that they provide a precise and unambiguous method for the specification of properties of systems. The language employed by PRISM for continuous-time Markov chains is the temporal logic CSL (Continuous Stochastic Logic). A typical property that can be expressed in CSL is the following: “the system reaches a goal state, while avoiding error states, within 30 minutes with probability greater than 0.99”, expressed as  $P_{>0.99}(\text{“error” } U^{<30} \text{ “goal”})$ . We note that this property not only is concerned with reachability of goal states, but also with the intermediate states visited on an execution path of the system before reaching a goal state. Another type of property refers to the long-run probability of residing in a given set of states: for example  $S_{<0.1}(\text{“error”})$  expresses the property that the long-run probability of residing in error states is less than 0.1.

We note that the two examples of CSL properties given above are assertions; that is, they are true or false for a given state of the model. PRISM allows us also to compute the probability of satisfying a CSL: for example, for each state of the system, the property  $P_{=?}(\text{“error” } U^{<30} \text{ “goal”})$  evaluates to the probability of reaching a goal state, while avoiding error states, within 30 minutes.

In the examples above, states sets corresponding to errors or the satisfaction of a goal are denoted by the propositions “error” and “goal”; for each such proposition, the user is required to express the state sets to which these propositions correspond, using predicates over the variables of the system model (for example, “error” =  $\text{proc1}=0 \ \& \ \text{proc2}=0$ , where  $\text{proc1}$  and  $\text{proc2}$  are system variables). Alternatively, the user may write predicates over system variables directly in the CSL formula instead of the propositions, for example:

$$P_{>0.99}((\text{proc1}=0 \ \& \ \text{proc2}=0) \ U^{<30} (\text{finished1}=1 \ | \ \text{finished2}=1)).$$

Another way of expressing the state sets used within a CSL formula is through the use of CSL itself: that is, we can nest CSL formulae within another CSL formula. For example, we may use CSL formulae, rather than expressions over system variables, to express the error or goal states in the above example.

Given an appropriate reward structure applied to the continuous-time Markov chain model, PRISM also permits the verification of reward-based properties. For example, the property  $R_{>3}(F \text{ "goal"})$  specifies that the expected cumulative reward before reaching a goal state is greater than 3, the property  $R_{\leq 4.5}(C_{\leq 10})$  specifies that the expected cumulative reward within the next 10 time units of system operation is not greater than 4.5, the property  $R_{>2.86}(I_{=10})$  specifies that the expected reward being gained at time instant 10 is greater than 2.86, and the property  $R_{<30}(S)$  specifies that the long-run average reward is less than 30.

## 2.5 DEEM

DEEM [Bondavalli et al.2000, Bondavalli et al.2004] is a dependability modelling and evaluation tool specifically tailored for Multiple Phased Systems (MPS). MPS are systems whose operational life can be partitioned in a set of non-overlapping periods, called "phases". Examples of MPS can be found in various application domains (nuclear, aerospace, transportation, electronics, and many other industrial fields). They include systems for the aided-guide of aircraft, whose mission-time is divided into several phases such as take-off, cruise, landing, with completely different requirements. A very important sub-class of MPS is represented by the so-called Scheduled Maintenance Systems (SMS) encountered in almost all the application domains where an artifact is to be used for long time and is periodically subject to maintenance actions. An SMS is easily formulated as an MPS, for which operational and maintenance phases alternate according to a prefixed schedule. CRUTIAL offers aspects whose organization is in the form of an MPS, which make DEEM appropriate in the context of the project. In particular, DEEM is currently applied to evaluate Proactive-Reactive Recovery strategies proposed to reinforce the intrusion tolerance of the CIS in the scope of the protection service (see Deliverable D25 [Donatelli 2008a]).

DEEM supports the methodology proposed in [Mura et al.1999, Bondavalli et al.1999, Mura and Bondavalli2001] based upon Deterministic and Stochastic Petri Nets (DSPN) as the modelling formalism and on Markov Regenerative Processes (MRGP) for the analytical model solution. When compared to existing general-purpose tools based on similar formalisms, DEEM offers advantages on both the modelling side (sub-models neatly model the phase-dependent behaviours of MPS), and on the evaluation side (a specialized algorithm allows a considerable reduction of the solution cost and time).

### 2.5.1 Modelling formalism and approach

The class of the DSPN [Ajmone Marsan and Chiola1987, Lindemann1993] has been chosen as the modelling formalism for DEEM. DSPN extend Generalized Stochastic Petri Nets and Stochastic Reward Nets, allowing for the modelling of events having deterministic occurrence times. Due to their high representative power, DSPN models are able to cope with the dynamic structure of MPS, and allow defining very concise models of even quite complex systems, through the use of guards on transitions, immediate transition priorities, multiplicity functions, rate and impulse rewards, etc.

The graphical representation of a DSPN model consists of places (drawn as circles), immediate transitions (drawn as thin bars), exponential timed transitions (drawn as white rectangular boxes), deterministic timed transitions (drawn as black rectangular boxes) and arcs. Places and transitions are connected by directed arcs (drawn as lines terminating with an arrowhead). An input arc connects an input place to a transition. An output arc connects a transition to an output place. Arcs have associated a positive integer weight, called multiplicity. Places may contain tokens, entities graphically represented as black dots or positive integers. The marking of the DSPN, representing the state of the net, is a vector

defined by the number of tokens in each place. A weight (no negative real number) is associated to each immediate transition. A priority is associated to each transition. Priority of timed transitions is 0. Priority of the immediate transitions is a real number greater than 0 (default is 1). A firing rate (a positive real number) is associated to each exponential transition, representing the random firing delay of the transition whose probability density function is a negative exponential. A constant time (a positive real number) is associated to each deterministic transition, representing the firing time. Moreover, for each transition, enabling conditions (named guards) can be defined in terms of complex expressions. A transition is said to have concession in marking if each enabling guard is true and each of its input places contains at least as many tokens as the multiplicity of the corresponding input arc. A transition is enabled if no higher priority transitions have concession. Immediate transitions once enabled fire in zero time. As soon as a timed transition gets enabled, a random firing time is sampled from the distribution associated to it, and a timer starts counting from that time down to zero. The timed transition fires if and only if it remains continuously enabled until the timer reaches zero. When the transition fires, from each input (output) place as many tokens as the multiplicity of the corresponding input (output) arc are removed (are added) in a single atomic and immediate operation (atomic firing rule). Moreover, an inhibitor arc (drawn as a line terminating with a circle at the transition) can be used to connect a (inhibitor) place to a transition. They inhibit the firing of a transition when a token is present in the input place (or in general when the number of token of the inhibitor place is equal or greater than the multiplicity of the arc). Weights are used to represent probabilistic choice when more immediate transitions are enabled on the same marking. Some attributes of the primitives, such as, guards, rates, firing times, transition weights and multiplicities may specified through arbitrary functions of the marking.

Models consist of two logically separate parts:

- Phase Net (PhN), which represents the behaviour of the system through the alternation of the various phases. PhN contains all the deterministic transitions of the overall DSPN model and may as well contain immediate transitions. At most one deterministic transition is enabled in each marking of the net. A marking of the PhN model represents a phase being executed, and the firing of a deterministic transition models a phase change. Notice that quite general structures of the PhN sub-model are allowed. In particular, the PhN is not limited to have a linear structure, but it may take a tree or cyclic structure [Bondavalli et al.1997, Bondavalli et al.1999, Mura and Bondavalli2001, Mura et al.1999].
- System Net (SN), represents the intra-phase evolution of an MPS. For instance, depending on the specific MPS, the SN may include the failure/repair behaviour of system components, the operations to be carried out within the operational phase, the maintenance activities, and so on. SN-type subnets are only allowed to include exponentially distributed and immediate transitions. This constraint is imposed to ensure the existence of a simple and computationally efficient time-dependent analytical solution. Besides that, any structure of the SN sub-model can be considered.

Each net can be made dependent on the other one by marking-dependent functions which modify attributes of the primitives, such as, transition rates, enabling conditions, transition weights and multiplicity functions to model the specific MPS features. A restriction is only imposed on the firing times of the deterministic transitions of the PhN, which are not allowed to be dependent on the marking of the SN. The attributes of objects like times, rates, weights and multiplicities, can be expressed as a function of parameters rather than numerical values directly.

### 2.5.2 Dependability measures and transient analysis

In the DEEM modeling framework, the measures of system behavior that are of interest to a user are specified in terms of performance variable based on “reward models” [Howard1971, Sanders and Meyer1991]. A reward model consists of a stochastic process and a “reward

structure". Following the same approach adopted in [Sanders and Meyer1991], the measures are defined through a reward structure that quantifies behaviours at the DSPN level. They are based on a general mechanism of marking-dependent reward functions. Informally, a reward structure consists of a rate reward and an impulse reward that are associated with the time spent in a state or with some event of the process, respectively. Rate rewards can be defined as arbitrary function of the model marking. Measures are further distinguished in three categories, in accordance with the time interval they depend on: instant-of-time, interval-of-time and time-averaged interval-of-time. The instant-of-time measure represents the reward that is associated with the status of the modelled system at a particular time. The interval-of-time and time-averaged interval-of-time measures represent the total and time-averaged reward accumulated during some interval of time, respectively. In DEEM, composed measures can be also defined, as a function of the evaluated reward-based measures.

Once measures are defined, the transient evaluation can be launched on the selected study. Results are collected in two output files, in formats compatible with spreadsheet programs and gnuplot respectively, so as to produce plots or tables of the dependability measures.

### 2.5.3 Solution methods

DEEM currently supports two classes of solution techniques: transient analytical solution based on the specialized solution methods proposed in [Mura et al.1999, Bondavalli et al.1999, Mura and Bondavalli2001] and discrete event simulation.

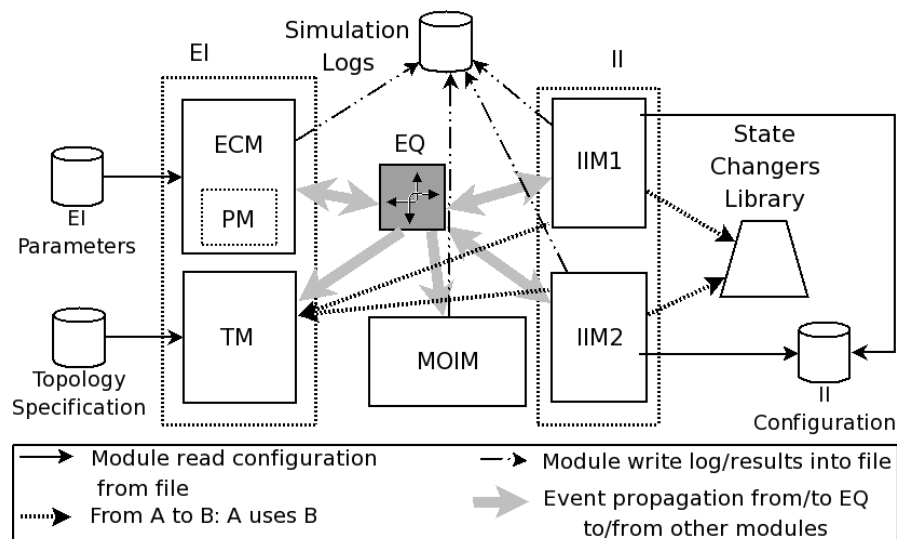
In its present version, the duration of the phases is deterministic and the phase model is restricted to contain only exponential and immediate transitions. Moreover, in every non-absorbing marking of the DSPN there is always one deterministic transition enabled, which corresponds to the phase being currently executed. The specialized analytical solution finds its ground by observing that the only deterministic transitions in a DSPN model of a MPS are the phase durations, and that these transitions are enabled one at the time. Thus, the marking process of the DSPN is a Markov Regenerative Process (MRGP) [Choi et al.1993, Kulkarni1995] for which the firing times of the deterministic transitions are indeed regeneration points. The simulator also supports models where the above assumptions are released.

## 2.6 EPSyS

The simulator EPSyS is under development at CNR-ISTI in the context of the CRUTIAL project. It is meant to assist the development of the generic, reusable model templates capturing the internal behaviour as well as external interactions of EPS entities. Therefore, it basically shares the same stochastic models of the two major infrastructures composing the electric power system (namely, the electric infrastructure EI and the information control infrastructure II), as already described in Deliverable D8 [Kaâniche et al. 2008] of this project. Here, the salient implementation modules the simulator is composed of are presented, as well as the simulation procedures and some information on its current availability stage.

### 2.6.1 Architecture of EPSyS

EPSyS is an ad-hoc discrete event simulator, described in [Romani et al.2007], designed for performability analysis of Electrical Power Systems. EPSyS has been structured in six main modules, as shown in Figure 7.



**Figure 7: Modules composing the EPSyS architecture**

Two of them (ECM and TM) reproduce the EI behaviour. The module ECM implements the (electrical) state of the elements of the transmission grid and their failure model. ECM also includes the protection system (PM) embedded into EI, which triggers the automatic defensive actions after a disruption and the hidden failure model. The module TM implements the topology of the transmission grid represented by an oriented graph. The modules IIM1 and IIM2 implement the behaviour of II at the two different abstraction levels considered. Both these components can be considered like an instance of a generic component IIM that supports the activation condition, the reaction delay, the reconfiguration strategy (*RS*) and the failure model of II. If the activation conditions are met, the corresponding corrective actions are performed immediately by IIM1 and within a reaction delay by IIM2. The component EQ implements the events queue, which allows the whole power system model to evolve reproducing the dynamics of EPS. Finally, there is the module MOIM that supports the measures of interest. The current EPSyS implementation has been developed in Python language, using C libraries for performance critical code. The following sections will describe the various EPSyS modules and provides some design notes.

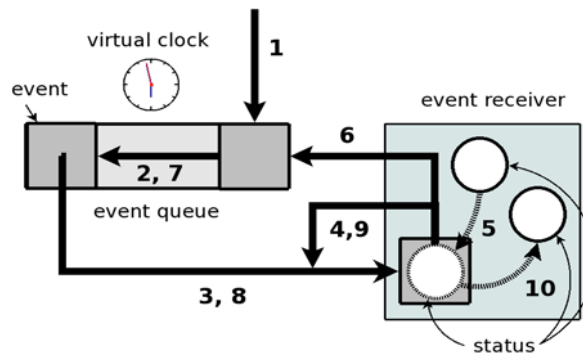
### 2.6.2 The coordination: EQ

EPSyS is a discrete event simulator. The core module is the Event Queue (**EQ**), which also contains the (virtual) clock associated with the queue itself. The EQ module takes care to properly serialize the simulation events according to their timing relations. An event is implemented with a simple data structure carrying all the information (time, destination, type of event, ...) needed by simulation to proceed. A peculiarity of EPSyS is that since events cause state transitions of elements, the new state is piggybacked in the event itself. Events can be injected into event queue or can be created by modelled element. During its lifespan, an event will traverse two phases, *sensing* or *handling* before being considered elapsed. Each phase is characterized by a specific processing delay, and one of two phases can be absent. The element receiving the event sees it twice, and it will handle the event according with its phase. The status of an event is stored in the event itself, and when a receiver processes an event, the event status is updated. An event is considered expired when the receiver has fully handled it, including the necessary receiver status updates. An expired event is thus removed from the event queue.

We can track the event lifecycle through Figure 8 as follows:

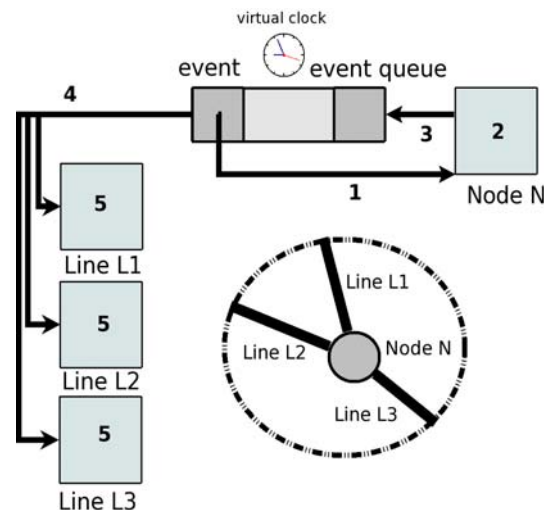
1. The event enters the event queue.
2. Events exit the event queue in the correct ordering thanks to the virtual clock associated to the event queue itself.
3. The event is delivered to right receiver by inspecting its *destination* field.
4. The receiver is notified, so it updates the event data and performs the event phases switch.
5. The receiver computes its next state and attaches it to event data.
6. The event processing must be completed later, so it is reinserted into the event queue.
7. Events exit again from the event queue after the appropriate virtual time is passed.
8. The event is passed to (former) receiver again.
9. The receiver is notified, it updates the event again, marking it as completed.
10. The receiver fetches the pre-computed state from event data, updates itself with such data, and finally discards the now elapsed event.

Figure 8: Event lifecycle in EPSyS



### 2.6.3 The EI Part: ECM, PM, TM

The **ECM** module contains the implementation of the simulated grid components. Each modelled physical grid component is represented by an *element*. Each element, in turn, is an instance of a given element class. EPSyS has model classes for *loads*, *power plants*, *transmission lines* and *substations*. Each element instance is characterized by model-dependent parameters (external temperature, maximum allowed load, priority) and by a behaviour dependent by its class. For example, a load can sustain overload for an indefinite amount of time, while transmission line has limited resistance, and power plants react to power oscillations. The modelling of the behaviour of protection embedded in the power system elements is implemented as part of the behaviour model of the element themselves. The status of EI elements is determined by events as outlined in the previous sections. This is also the way the elements interact with each other: they interact indirectly by generating or propagating events. For example, a substation will handle a fault affecting it by going offline and by generating fault events for all directly attached transmission lines.



**Figure 9: Communications between EI elements using events**

Figure 9 summarizes this example, as follows:

1. An event is fetched from the event queue and it is delivered to the receiver (*Node N*, a substation).
2. The notified node recognizes the received event meaning a fault, so it prepares for disabling itself. Before shutting down, the node checks if it must propagate the fault. Being a substation, it must propagate its fault to all the attached transmission lines. So the node discovers its neighbours using services provided by the TM module, and generates *disable* events directed to those lines.
3. The node inserts the created events into the event queue, then finally disables itself.
4. The new events are (instantaneously) extracted from the queue and they are propagated to the appropriate lines.
5. The elements are notified and they disable themselves accordingly.

The status of EI elements also drives the values of all internal status and performance variables that are used by MOIM module (see below). Different variables are attached to different element classes; for example, transmission lines never provide any gain, while loads provide both gain if satisfied and penalty cost if their requests are not fulfilled.

The **TM** module encapsulates all topology information of the modelled grid, being completely independent from the ECM module. It provides low-level adjacency information needed for flow redispatch decisions, for islands detection, and event propagations. Most of supporting topological functions, as the detection of all connected elements, as well as island split and join handling, are part of this module.

#### 2.6.4 The II part: IIM1, IIM2 and the state change functions


The **IIM1** and **IIM2** implement the two functions representing the local and global EI reconfigurations in charge of the control infrastructure (called  $RS_1()$  and  $RS_2()$ , respectively). Both use a state change procedure from the internal EPSyS library. A state change procedure takes as inputs the current state of the EI and the current *adjacency matrix* of the EI, representing the available interconnections, and produces a new stable state. An EI state consists of the values for the physical characteristics of the system, the most important value being the **power flow** through transmission lines or substations, or injected into grid by power plants, or received by loads. The implementation of the state change procedure is bounded only by the EPSyS representation of the EI elements and physical parameters. In accordance with the literature, the current implementation formulates the new state

computation as Linear Programming (LP) problems, by minimizing the differences between old and new states, since such differences are perceived as a *cost*. Therefore, we say that state changes are implemented using different *cost functions*. The implementation of the II control functions is based on the following LP problem:

$$\min(W_0 Z + W_1(\sum_i |P_i - P'_i|) + W_2(\sum_j |R_j - R'_j|))$$

subject to the following constraints:


$$[d2] \begin{cases} \sum_i P_i = \sum_j R_j \\ P - R = B\theta \\ F = bA\theta \\ F \leq Z \end{cases}$$

This problem has different tunable parameters that determine the cost to be minimized, allowing this basic function to be ctively reused in different situations. The parameters



- $Z$  is a constant that bounds the *difference* between any pair of power flows in the grid under a given threshold. This allows controlling the spreading of flow through lines.
- $W_0$  is the weight (constant) associated to the power flow. The higher the value, the higher is the chance of line overload.
- $W_1$  is the weight (constant) associated to the power production. The higher the value, the less a power plant will change its power generation. On the other hand, low values let the power production to have great oscillations.
- $W_2$  is the weight (constant) associated to the power delivering. The higher the value, the less is the chance of a load shedding. On the other hand, low values let the load to be shed.

The other symbols used in the problem are :

-  is the vector of power flow *injected* in the grid by each node.
- $P'$  is the former value of  $P$  in the context of state change computation.
- $R$  is the vector of power flow *drained* from the grid by each node.
- $R'$  is the former value of  $R$  in the context of state change computation.
- $N$  is the cardinality of vectors  $P$  and  $R$ , corresponding to the number of nodes into the grid.
- $j$  is the number of *loads* in the grid.  $J < N$  always holds (also  $I+J=N$  holds).  $j$  is the generic load.
- $i$  is the number of *plants* in the grid.  $I < N$  always holds (also  $I+J=N$  holds).  $i$  is the generic plant.
- $F$  is the vector of power flow through transmission lines.
- $M$  is the cardinality of vector  $F$ , corresponding to the number of lines into the grid.  $m$  is the generic line.
- $b$  is the *weight* vector of lines, having cardinality  $M$ . High weight means that a line will likely carry high amount of flow.
- $A$  is the *adjacency matrix* of the grid.
- $B$  is a matrix defined as  $A'bA$ .

Any status of EI is represented by combinations of values of  $P$  and  $F$ . There are some major behavioural differences between IIM submodules: IIM1 delivers immediately the new state while IIM2 stores it internally, simulating a complex computation, and delivers it after the delay is expired. This reflects the assumption that the local  $RS_1()$  reconfiguration



implemented through IIM1 is performed instantaneously, while the  $RS_2()$  global one (implemented through IIM2) requires an amount of time to be computed and applied.

### 2.6.5 Evaluating measures: MOIM

The **MOIM** module collects various data during the simulation runtime and computes the value of measures of interests; data is collected at every state change of the system. EPSyS has a fixed set of measures of interest, and they can be computed per-element, system-wide or both. Each measure is expressed as a function of the value and the duration of the state that each element traverses during the simulation. EPSyS produces a full report of evaluated measures, including the final value of each measure, the value of each measure as a function of the simulation time, and the time instants corresponding to system state changes. EPSyS produces very detailed input to allow to inspect in detail the evolution of the scenario and to allow understanding the grid dynamics. Log files are produced, which contain detailed state transition for each event and for each modelled element, and can track internal interaction between elements. Log files can be transparently compressed.

### 2.6.6 The simulation procedure

The states of the EI are represented by the tuple  $(E_{i,h}, T_i)$ , with  $i=1,2,\dots$ , and  $h=1,2$ .  $T_i$  is the topology and  $E_{i,h}$  represents the electrical state resulting from the application of the reconfiguration strategy  $RS_h()$  on the topology  $T_i$ . The main steps of a simulation run are:

- 1) For a given topology  $A$  and a given susceptance matrix  $B$ , the initial setting for  $P$  and  $F$  can be derived manually or solving an optimization problem using the same approach adopted to evaluate the functions  $RS$ . The initial state of EI is represented by  $(E_{0,2}, T_0)$ .
- 2) Values of the occurrence times of random disruptions (of EI and failures of II) are sampled from the distributions considered for the failure model.
- 3) When, in the state  $(E_{i,2}, T_i)$ , a random (or cascading) disruption is selected from the event queue, the following steps are performed with  $j=i+1$ :
- 4) The disruption immediately propagates to the neighbour components, based on the hidden failure model of the protections, generating a new topology  $T_j$  for EI.
- 5) The function  $RS_2()$  is evaluated, based on the electrical state  $E_{i,2}$  and on the topology  $T_j$ .
- 6) The function  $RS_1()$  is evaluated, based on the electrical state  $E_{i,2}$  and on the topology  $T_j$ . If the definition of  $RS_1()$  is based on the result of the optimization problem defined for  $RS_2()$ , the function  $RS_2()$  must be evaluated before  $RS_1()$ .
- 7) The outputs  $P$  and  $F$  of  $RS_1()$  are immediately applied to EI (i.e., at the system time of the disruption occurrence), generating a new  $E_{j,1}$  for EI.
- 8) The values of the occurrence times of cascading disruptions for the new state  $(E_{i,1}, T_j)$ , are sampled from the overloading failure model.
- 9) The value of the delay  $t_2^{RS}$  to the occurrence of the reconfiguration  $P$  and  $F$ , obtained through the solution of  $RS_2()$ , is sampled from the distributions considered for the II model.
- 10) When cascading or random disruptions occur (being selected from the event queue) before time  $t_2^{RS}$ , the simulation goes to step 4. With  $j=j+1$  (where the evaluation of  $RS_2()$  is restarted based on the new topology generated by the new disruption).
- 11) Otherwise, if no disruption occurred before  $t_2^{RS}$ , at time  $t_2^{RS}$ , the outputs  $P$  and  $F$  of  $RS_2()$  are applied, generating a new state  $(E_{j,2}, T_j)$  for EI.
- 12) The values of the occurrence times of cascading disruptions for the new state  $(E_{j,2}, T_j)$  are sampled from the overloading failure model.
- 13) If an halting condition is satisfied (e.g, a total blackout, or a state without cascading and random disruptions, or when a pre-determined instant of time is reached), the simulation stops, otherwise the simulation goes to step 3.

Note that the functions  $RS_1()$  and  $RS_2()$  are evaluated on the states  $(E_{i,2}, T_j)$ , although

alternative choices could be considered. Moreover, when cascading or random disruptions occur before time  $t^{RS_2}$  (step 10.), instead of restarting the evaluation of  $RS_2()$ , different alternatives can be considered, such as performing immediate but less effective actions, or even do nothing.

### 2.6.7 Availability of the tool

EPSyS is implemented in a portable way and has no platform constraints. The EPSyS core runs in every platform supported by the Python runtime [PYTHON]. The C libraries currently needed by EPSyS, GLPK [GLPK] and GMP [GMP] are available as well on all the major platforms. EPSyS, as well as the needed libraries, is available for usage after a simple free registration request. The sources are available as well on demand.

## 3 DISTRIBUTION FITTING AND STATISTICAL ANALYSIS OF EVENTS

The stochastic models used to assess quantitative dependability or security measures rely on assumptions and parameters that need to be estimated based on real data. Such data could be collected from the field or during controlled experiments based for example on fault or attack injection. Statistical analysis methodologies are needed to process the collected data and extract relevant information to estimate the values of the parameters and the forms of distributions characterizing the random variables involved in the models discussed in the previous sections. For example, we need to characterize the distributions of the times between failures, the times between attacks or repair durations, and to estimate the coverage probabilities reflecting the error detection and recovery mechanisms efficiency. These quantities correspond to random variables for which we need to estimate representative statistical distributions that faithfully reflect the behaviour of the collected data.

The activities carried out in CRUTIAL in this context mainly concern the characterization of attack processes based on data collected from the Internet (See Deliverable D25 [Donatelli 2008a]).

Starting from the raw data collected from the field or the controlled experiments, we can distinguish three main steps to estimate such statistical distributions and models:

- 1) data pre-processing and categorization
- 2) statistical inference to estimate parameters values and confidence intervals
- 3) model validation through the use of hypothesis testing to accept or reject assumptions or the form of the distribution

### 3.1 Data pre-processing and categorization

Generally, the data collected contain a large set of heterogeneous information with a mixture of events characterizing the system behaviour under normal operation conditions and also in the presence of errors and failures and recovery actions. Data processing and clustering algorithms are generally needed in order to extract the relevant information that characterize the observed error patterns, their causes and their effects, and correspond to the occurrence of the dependability and security related random variables under interest (times between failures, times between attacks, etc.). A review of the state of the art and examples of common algorithms used in this field can be found in [Siewioreck 2004, Iyer & Kalbarczyk 2002].

The application of such algorithms to the collected data will produce a sample of observed values of the random variables and events under interest to which statistical analysis techniques can be applied to identify the corresponding distributions and estimate their associated parameters.

One of the concerns that need to be addressed at this stage is the identification of atypical or infrequent values among the observed sample that are significantly separated in value from the rest of the other observations. Such values, generally called “outliers”, likely correspond to unrealistic values resulting e.g., from errors or mis-reporting in the measurement or the data collection process. Thus, they need to be considered carefully and possibly filtered out, otherwise they might have a significant impact on the statistical analysis and modelling results.

Various graphical techniques and statistical tests exist for the identification of outliers, e.g., boxplot or Dixon and Grubbs tests [Hodge & Austin 2004].

For example, the boxplot is a popular visual technique that uses the first and third quartiles of the observed sample for the identification of outliers. If the first quartile is  $Q1$  and the third quartile is  $Q3$ , then the difference ( $Q3 - Q1$ ) is called the interquartile range or IQR. Any data observation which lies more than  $1.5 \cdot \text{IQR}$  lower than the first quartile or  $1.5 \cdot \text{IQR}$  higher than the third quartile is considered an outlier.

A modified version of the boxplot test has been defined in [Vanderviere & Hubert 2004], which is well suited when the distribution of the data is skewed. This test proceeds in two steps.

At a first step, this test computes for the considered data set  $D$  a metric denoted as  $MC(D)$ , taking values in the interval  $[-1, 1]$ , that measures the skewness of the distribution. Positive (respectively, negative) values correspond to positively (respectively, negatively) skewed distributions, and when  $MC(D)$  is null the distribution is not skewed.

At a second step, the test computes a critical interval that depends on the sign of the skewness metric  $MC(D)$ , such that any value outside this interval is considered as an outlier.

The test identifies outliers as follows:

- *If  $MC(D) \geq 0$ ,  $x$  is an outlier  $\Leftrightarrow$*   
 $x \notin [Q1 - 1.5 e^{-4MC(D)} \text{IQR}; Q3 + 1.5 e^{3MC(D)} \text{IQR}]$
- *If  $MC(D) < 0$ ,  $x$  is an outlier  $\Leftrightarrow$*   
 $x \notin [Q1 - 1.5 e^{-3MC(D)} \text{IQR}; Q3 + 1.5 e^{4MC(D)} \text{IQR}]$

Once outliers are identified, then, the question is what should we do with these outliers? Different solutions could be investigated:

- 1) Remove the outliers from the data set
- 2) Substitute them by synthetic values generated based on the general characteristics of the sample distribution.
- 3) Accommodate the presence of the outliers through the use of appropriate techniques that reduce their potential impact on the statistical results, e.g., i) data transformation, ii) use of robust estimation techniques or iii) selection of a subset of the data such that the impact of the outliers is reduced.

The selection of the most appropriate solution depends on the application context.

### 3.2 Statistical inference and model validation

Using the data selected from the data pre-processing step, statistical inference techniques are aimed at estimating the characteristics of the probability distributions that faithfully reflect the behaviour of the considered data. Different candidate distributions can be investigated. Two main steps can be distinguished. The first one is dedicated to the estimation of the parameters defining the considered distribution and the second one is aimed at the selection of the most appropriate distribution based on statistical hypothesis testing.

Various methods exist to support parameter estimation. The most commonly used ones are least-squares estimation (LSE), maximum likelihood estimation (MLE) and the method of moments. Unlike MLE, LSE requires no or minimal distributional assumptions. The MLE method requires the knowledge of the probability density function (pdf) under investigation. It is based on the principle that the desired probability distribution is the one that makes the observed data “most likely”. MLE method have interesting statistical properties, however it is not always easily applicable, especially when it is difficult to find maximum likelihood estimators in closed form. The method of moments consists of equating the first few moments estimated from the observed sample with the corresponding theoretical ones, and solving equations for the parameters.

The application of these methods generally requires the use of iterative numerical solving techniques to find the optimal parameters, e.g., the Newton Raphson method, Expectation-Maximization algorithm, etc. [Kay 1993].

For each considered distribution, the estimated parameters provide the best fit of the distribution to the observed data. However, this does not mean that this distribution will be appropriate to describe the data. Statistical hypothesis tests are generally used to assess the quality of the distribution fitting and to decide whether to reject or accept the assumption that observed data could have been derived from the estimated distribution. Examples of commonly used statistical tests include the chi-square test, and the Kolmogorov-Smirnov test. The quality of fit can also be appreciated by evaluating the residue, i.e., the mean error between the estimated values and the observed values of the random variable under investigation.

Several statistical packages that support statistical inference and hypothesis testing are available. These include commercial tools, e.g., SAS, SPSS, S-PLUS and also open source tools, e.g., R. Indeed, R is a programming language and a statistical computing environment distributed under the GNU GPL license. It is widely used for being free and for supporting a large variety of statistical and numerical techniques. Also, it is highly extensible through the use of packages, which are user-submitted libraries for specific functions or specific areas of study.

In the context of CRUTIAL, the R tool is used to support the characterization of attack processes based on real data collected from the Internet. The technical details and the initial results obtained are presented in deliverable D25 [Donatelli 2008a].

## 4 CONCLUSIONS

This deliverable has reported on the tools that have been selected by the consortium to be used for the evaluation of CRUTIAL architectural solutions.

The tools identified will allow the analysis of the architectural solutions in isolation to study their correctness and to provide input for tuning their parameters: PRISM and DEEM have already been used for this goal (see D25 [Donatelli 2008a]).

The use of Möbius and the ad-hoc simulator that has been built in the project will also allow the evaluation of the added value of the use of the CRUTIAL architecture in terms of reliability, availability, etc. of the service delivered by the EPS.

All models considering the impact of malicious faults will take advantage of the availability of real data on attackers behaviour (collected through honeypot as explained in D25 [Donatelli 2008a]) and of the statistical analysis applied on these data, so that evaluation based on models will be able to use “statistically approved” input data for the models.

## REFERENCES

- [Adve *et al.* 2000] V. S. Adve, R. Bagrodia, J. C. Browne, E. Deelman, A. Dube, E. Houstis, J. Rice, R. Sakellariou, D. Sundaram-Stukel, P. J. Teller and M. K. Vernon, "Poems: End-to-end performance design of large parallel adaptive computational systems", *IEEE Transactions on Software Engineering, Special Section of invited papers from the WOSP '98 Workshop*, 26 (11), pp.1027-1048, 2000.
- [Ajmone Marsan and Chiola1987] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In G. Rozenberg, editor, *Advances in Petri Nets 1987*, volume 266 of LNCS, pages 132–145. Springer Verlag, 1987.
- [Ajmone-Marsan *et al.* 1995] M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, "Modelling with Generalized Stochastic Petri Nets", J. Wiley and Sons, 1995.
- [Alur *et al.* 1999] R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7-48, 1999.
- [Bause *et al.* 1998] F. Bause, P. Buchholz and P. Kemper, "A toolbox for functional and quantitative analysis of dedcs", in *Lecture Notes in Computer Science* (N. N. S. R. Puigjaner, and B. Serra, Ed.), LNCS 1469, pp.356-359, Springer-Verlag, 1998.
- [Béounes *et al.* 1993] C. Béounes, M. Aguéra, J. Arlat, C. Bourdeau, J.-E. Doucet, K. Kanoun, J.-C. Laprie, S. Metge, J. Moreira de Souza, D. Powell and P. Spiesser, "SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems", in *23rd IEEE Int. Symposium on Fault-Tolerant Computing (FTCS-23)*, (Toulouse, France), pp.668-673, IEEE Computer Society Press, 1993.
- [Bernardi *et al.* 2001] S. Bernardi, S. Donatelli and A. Horváth, "Special section on the practical use of high-level Petri nets: Implementing compositionality for stochastic Petri nets." *Journal of Software Tools for Technology Transfer (STTT)*. pp.417-430, 2001.
- [Chiola *et al.* 1995] G. Chiola, G. Franceschinis, R. Gaeta and M. Ribaud, "Greatspn 1.7: Graphical editor and analyzer for timed and stochastic Petri nets", *Performance Evaluation*, 24 (1-2), pp.47-68, 1995.
- [Chiola *et al.* 1993] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad, "Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications", *IEEE Transactions on Computers*, volume 42, pp. 1343-1360, 1993.
- [Ciardo & Miner 1996] G. Ciardo and A. S. Miner, "Smart: Simulation and markovian analyzer for reliability and timing", in *IEEE International Computer Performance and Dependability Symposium (IPDS'96)*, (Urbana-Champaign, IL, USA), p.60, IEEE Computer Society Society, 1996.
- [Bondavalli *et al.*1997] A. Bondavalli, I. Mura, and M. Nelli. Analytical modelling and evaluation of phased-mission systems for space applications. In *IEEE HASE'97, High Assurance System Engineering Workshop*, pages 85–91, Washington, DC, USA, August 11-12 1997.
- [Bondavalli *et al.*1999] A. Bondavalli, I. Mura, and K. S. Trivedi. Dependability modelling and sensitivity analysis of scheduled maintenance systems. In *EDCC-3 European Dependable Computing Conference* (also LNCS N. 1667), pages 7–23, Prague, Czech Republic, September 1999. Springer Verlag.
- [Bondavalli *et al.*2000] A. Bondavalli, I. Mura, S. Chiaradonna, R. Filippini, S. Poli, and F. Sandrini. DEEM: a tool for the dependability modeling and evaluation of multiple phased systems. In *DSN-2000 IEEE Int. Conference on Dependable Systems and Networks (FTCS-30 and DCCA-8)*, pages 231–236, New York, US, June 25-28 2000.

- [Bondavalli et al.2004] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and I. Mura. Dependability modeling and evaluation of multiple-phased systems, using DEEM. *IEEE Transactions on Reliability*, 53(4):509–522, 2004.
- [Buchholz 1995] P. Buchholz, “Hierarchical Markovian Models: Symmetries and Reduction”, *Performance Evaluation*, 22 (1), pp.93-110, 1995.
- [Ciardo & Trivedi 1993] G. Ciardo and K. S. Trivedi, “Decomposition Approach to Stochastic Reward Net Models”, *Performance Evaluation*, 18 (1), pp.37-59, 1993.
- [Christensen 2001] A. Christensen. Result specification and model connection in the Möbius modeling framework, Masters Thesis, 2001.
- [Choi et al.1993] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Transient analysis of deterministic and stochastic Petri nets. In 14th Int. Conference on Application and Theory of Petri Nets, pages 166–185, Chicago Illinois, USA, 1993.
- [Clark et al. 2001] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, P. Webster: The Möbius Modeling Tool. In pnpm, p. 0241, 9th international Workshop on Petri Nets and Performance Models (PNPM'01), 2001.
- [Codetta-Raiteri et al. 2006] D. Codetta-Raiteri, G. Franceschinis, M. Gribaudo, "Defining formalisms and models in the Draw-Net Modeling System", *Proceedings of the Fourth International Workshop on Modelling of Objects, Components and Agents (MOCA '06)*, pp. 123-144, Turku, Finland, June 2006.
- [Daly 2001] D. Daly. Analysis of connection as a decomposition technique. PhD thesis, University of Illinois, Illinois, 2001.
- [Donatelli 2008a], Donatelli et al, "Model-based evaluation of the middleware services and protocols & architectural patterns ", CRUTIAL project, Work Package 5, Deliverable D25, January 2008.
- [Daly et al. 2000] D. Daly, D. D. Deavours, P. G. Webster and W. H. Sanders, “Möbius: An extensible tool for performance and dependability modeling”, in *11th International Conference, TOOLS 2000*, (H. C. B. B. R. Haverkort, and C. U. Smith, Ed.), (Schaumburg, IL, USA), pp.332-336, Lecture Notes in Computer Science, 2000.
- [Franceschinis et al. 2002] G. Franceschinis, M. Gribaudo, M. Iacono, N. Mazzocca and V. Vittorini, “Drawnet++: Model objects to support performance analysis and simulation of complex systems”, in *Lecture Notes in Computer Science LNCS 2324*, pp.233-238, Springer-Verlag, 2002.
- [Fricks et al. 1997] R. Fricks, C. Hirel, S. Wells and K. Trivedi, “The development of an integrated modeling environment”, in *World Congress on Systems Simulation (WCSS '97)*, (Singapore), pp.471-476, 1997.
- [German et al. 1995a] R. German, C. Kelling, A. Zimmermann and G. Hommel, “Timenet: A toolkit for evaluating non-markovian stochastic petri-nets”, *Performance Evaluation*, 24, pp.69-87, 1995a.
- [Garrone et al. 2007] F. Garrone, C. Brasca, D. Cerotti, D. Codetta-Raiteri, A. Daidone, G. Deconinck, S. Donatelli, G. Dondossola, F. Grandoni, M. Kaâniche, T.Rigole, "Analysis of new control applications", CRUTIAL project, Work Package 1, Deliverable D2, January 2007.
- [Gribaudo et al. 2005] M. Gribaudo, D. Codetta-Raiteri, G. Franceschinis, "Draw-Net, a customizable multi-formalism multi-solution tool for the quantitative evaluation of systems", *Proceedings of the 2nd International Conference on Quantitative Evaluation of Systems*, pp.

257-258, Turin, Italy, September 2005.

[GLPK] The GNU Linear Programming Kit: <http://www.gnu.org/software/glpk/>

[GMP] The GNU Multi Precision arithmetic library: <http://gmplib.org>

[Hinton et al. 2006] A. Hinton, M. Kwiatkowska, G. Norman and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In H. Hermanns and J. Palsberg (editors) *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of LNCS, pages 441-444, Springer. March 2006.

[Henzinger et al. 1995] T. A. Henzinger, P.-H. Ho and H. Wong-Toi, "A user guide to HyTech", in *1st Workshop Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp.41-71, Springer Verlag, 1995.

[HiQPN] HiQPN, "The HiQPN Software.web link: <http://ls4-www.informatik.uni-dortmund.de/QPN.>"

[Howard1971] R. A. Howard. Dynamic Probabilistic Systems, volume II: Semi-Markov and Decision Processes. Wiley, New York, 1971

[Illié et al. 2004] J. M. Illié, S. Baarir, M. Beccuti, C. Delamare, S. Donatelli, C. Dutheillet, G. Franceschinis, R. Gaeta and P. Moreaux, "Extended SWN solvers in GreatSPN", in *1st International Conference on Quantitative Evaluation of Systems (QEST)*, (Enschede, The Netherlands), IEEE Computer Society Press, 2004.

[Kaâniche et al. 2007] M. Kaâniche et al., CRUTIAL D3 - Methodologies synthesis, Gen 2007.

[Kaâniche et al. 2008] M. Kaâniche et al., "Preliminary modelling framework", CRUTIAL project, Work Package 2, Deliverable D8, January 2008.

[Kulkarni1995] V. G. Kulkarni. Modeling and Analysis of Stochastic Systems. Chapman & Hall, London, 1995.

[Lindemann1993] C. Lindemann. An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models. *Performance Evaluation*, 18(1):79-95, 1993.

[Lindemann et al. 1999] C. Lindemann, A. Reuys and A. Thümmler, "The dspnexpress 2.000 performance and dependability modeling environment", in *29th Annual Int. Symp. on Fault-Tolerant Computing (FTCS-29)*, (Madison, Wisconsin, USA), pp.228-231, IEEE Computer Society, 1999.

[Muppala et al. 1992] J. K. Muppala, A. Sathaye, R. Howe and K. S. Trivedi, "Dependability Modeling of a Heterogeneous VAX-cluster System Using Stochastic Reward Nets", in *Hardware and Software Fault Tolerance in Parallel Computing Systems* (D. R. Avresky, Ed.), pp.33-59, 1992.

[Mura and Bondavalli2001] I. Mura and A. Bondavalli. Markov regenerative stochastic Petri nets to model and evaluate the dependability of phased missions. *IEEE Transactions on Computers*, 50(12):1337-1351, 2001.

[Mura et al.1999] I. Mura, A. Bondavalli, X. Zang, and K. S. Trivedi. Dependability modeling and evaluation of phased mission systems: a DSPN approach. In *IEEE DCCA-7, IFIP Int. Conference on Dependable Computing for Critical Applications*, pages 319-337, San Jose, CA, USA, January 6-8 1999.

[Movaghar & Meyer 1984] A. Movaghar, and J. F. Meyer. Performability modelling with stochastic activity networks. In *Proc. of the Real-Time Systems Symposium*, pages 215-224,



1984.

[Python] The Python programming language: <http://www.python.org>

[Pooley 1991] R. J. Pooley, "The integrated modelling support environment: A new generation of performance modelling tools", in *5th International Conference in Computer Performance Evaluation: Modelling Techniques and Tools*, (G. B. a. G. Serazzi, Ed.), (Torino, Italy), pp.1-15, 1991

[Romani et al.2007] F. Romani, S. Chiaradonna, F. Di Giandomenico, and L. Simoncini. Simulation models and implementation of a simulator for the performability analysis of electric power systems considering interdependencies. In *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*, pages 305-312, 2007.

[Sanders & Meyer 1991] W. H. Sanders, and J. F. Meyer. A unified approach for specifying measures of performance, dependability and performability. In *Dependable Computing for Critical Applications*, volume 4 of *Dependable Computing and Fault-Tolerant Systems*, pages 215-237. Springer Verlag, 1991.

[Sanders & Meyer 2001] W. H. Sanders, and J. F. Meyer. Stochastic activity networks: Formal definitions and concepts. In *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *Lecture Notes in Computer Science*, pages 315-343. Springer Verlag, 2001.

[Sanders and Meyer1991] W. H. Sanders and J. F. Meyer. A unified approach for specifying measures of performance, dependability and performability. In A. Avizienis and J. Laprie, editors, *Dependable Computing for Critical Applications*, Vol. 4 of *Dependable Computing and Fault-Tolerant Systems*, pages 215–237. Springer Verlag, 1991.

[Sanders & Meyer 2001] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts", in *Lectures on Formal Methods and Performance Analysis. Lecture Notes in Computer Science 2090*, pp.315-343, Springer-Verlag, 2001.

[Schneeweiss 1999] W. G. Schneeweiss, "The Fault Tree Method", LiLoLe Verlag, 1999.

[Trivedi 2002] K. Trivedi, "Sharpe 2002: symbolic hierarchical automated reliability and performance evaluator", in *IEEE Int. Conference on Dependable Systems and Networks (DSN 2002)*, (Washington, DC, USA), p.544, IEEE Computer Society, 2002.

[van Moorsel & Huang 1998] A. P. A. van Moorsel and Y. Huang, "Reusable software components for performability tools, and their utilization for web-based configuration tools", in *10th International Conference in Computer Performance Evaluation: Modelling Techniques and Tools*, (N. N. S. R. Puigjaner, and B. Serra, Ed.), (Palma de Mallorca, Spain), pp.37-50, 1998.