

DefPlans: Agent Modeling Techniques for Power System Emergency Control

E. Ciapessoni, E. Corsetti - ERSE

Abstract—The occurrence in recent years of catastrophic blackouts in worldwide power systems has raised on the need to endow specific protection schema to deal with emergency control. The paper presents DefPlans, an environment envisaged to specify and validate defense plans for bulk power systems. In DefPlans is possible to specify the emergency control functionality according to the multi-agents paradigm, and validate them by the Power System Dynamic simulator (PSD). DefPlans bridge the gap between the specification and the code, and offers a number of features tailored to design real-time, concurrent and distributed functionality.

Index Terms— Multi-agent systems, emergency control, real-time, timed operators

I. INTRODUCTION

Major blackouts raised fundamental questions about the appropriateness of the rules, regulations and system operating practices governing transmission system security. Despite the considerable efforts put on since 2000, to address the weaknesses exposed by the blackouts, it can still be argued that the development of these rules and operating practices have not kept pace with the fundamental changes resulting from electricity market reform. System operating practices need to give greater emphasis to system-wide preparation to support flexible, integrated real-time system management [8]. Real-time coordination, communication and information exchange, particularly within integrated transmission systems spanning multiple control areas, can and must be improved [11]. Effective real-time system operation requires accurate and timely information and state-of-the-art technology to facilitate effective contingency planning, system management and coordinated emergency responses. New and existing technology could be more fully employed to enhance effective system operation [9]. Modern control systems must meet increasingly demanding requirements to cope with significant degrees of uncertainty, as well as with more dynamic

environments, and to provide greater flexibility ([3], [4] and [5]). This implies that control systems software is highly complex, in that it has a large number of interacting parts. This complexity requires the use of state-of-the-art software engineering methodologies. In this paper, we will argue that analyzing, designing, and implementing such complex software systems as a collection of interacting, autonomous, flexible components (i.e., as agents) presents significant advantages over common used methods, [4], [5] and [7].

A multi-agent system contains multiple, interacting computing elements, called agents, which are capable of "rational" autonomous (i.e., independent) action — making independent decisions about what actions to perform in the furtherance of their goals. From the programming language point of view, the key issue is "what is the *right way* to program agents". DefPlans constructs are based on "*agent-oriented programming*" and provide methods to specify single and coordinated functionalities, communication between agents and inference rules to specify agent behavior. DefPlans also support time reasoning by means of a suitable set of time operators.

The multi-agents methodology we propose to specify and validate defense plans is coupled with an automatic translation process that, taking the specification of a defense plan, generates an executable program. Such a program can be linked as a module of the Power System Dynamic simulator (PSD) in order to validate and verify the specification.

The paper presents in chapter II the defense plans characteristics and the multi-agents paradigm. In chapter III the DefPlans specification abilities are discussed. Chapter IV presents the verification and validation support. Conclusion highlights future work.

II. POWER SYSTEM DEFENSE PLANS AND MULTI-AGENTS

A. Defense plans

To guarantee the security of an electric power system, as well as incorporating some overcapacity into the system, it is necessary to develop a global defense plan, able to cover the possible emergency situations. When looking at the most recent large-scale blackouts, critical phenomena such as:

- overloaded extra-high-voltage (EHV) lines, and possibly uncontrolled cascades of line tripping,
- voltage reduction, possibly leading to voltage collapse problems caused by reduced reactive power support,

Manuscript received July 1, 2009. This work has been financed by the Research Fund for the Italian Electrical System under the Contract Agreement between CESI RICERCA and the Ministry of Economic Development - General Directorate for Energy and Mining Resources stipulated on June 21, 2007 in compliance with the Decree n.73 of June 18, 2007.

Authors are with ERSE (ENEA-Research in the Power System, former CESIRICERCA), via Rubattino, 54; Milan - ITALY ({emanuele.ciapessoni, edoardo.corsetti}@erse-web.it).

- local and wide area transient stability problems, related to the increase of long distance interchanges and coupled with the risk of reduced voltage support,
- inter-area oscillations due to the interconnection,
- intervention of protecting device (*overcurrent protections, differential protections, ...*) causing cascading events

combine and interact in complex fashion, notably because of some internal *hidden failures* of the protection system. Defense plans are used to ensure that the overall power system is protected against major disturbances involving multiple contingency events [2], [4] and [8].

This objective can only be reached with coordinated automatic emergency measures, such as generation run back schemes, load or generation rejection, load shedding, reactive switching, bus or system splitting, able to manage extreme contingencies. These automatic measures must counter the post-disturbance dynamic phenomena so that the network remain in a viable operating state. In particular, they are intended to minimize and reduce the severity and consequence of low probability and unexpected - possibly multiple - events and to prevent a widespread collapse of power.

The main characteristics of a defense plan are:

- *reactivity*: defense plan must react to external stimuli in order to pursue the mission by means of a set of coordinated actions;
- *real-time*: coordination measures require the satisfaction of time constraints to be applied;
- *closed loop-cycle*: defense plans verify the efficacy of the actuated actions;
- *adaptive*: response must be tuned to the disturbances.

A defense plan acts according to three main phases:

- *identification* of a critical status;
- *decision* about the suitable action to deal with it;
- *application* of the action.

B. Emergency control by agents

The functional architecture of a defense system comprises a set of functionalities controlling and protecting the lines, the nodes, the corridors, the sections, the areas and the power system as a whole. A reference agent based architecture is depicted in Figure 1, where the design of each agent consists of two components:

- the *functional design* captures the “*what*”, it describes the system’s functions-actions, processes. It also includes the input and output flows of information between system components and between system and its environment;
- the *behavioral design* captures the “*when*”, it describes the system behavior over time, the dynamic of actions, their control, time constraints and the states and modes of the system. Behavioral design also deals with *causality*, *concurrency* and *synchronization*.

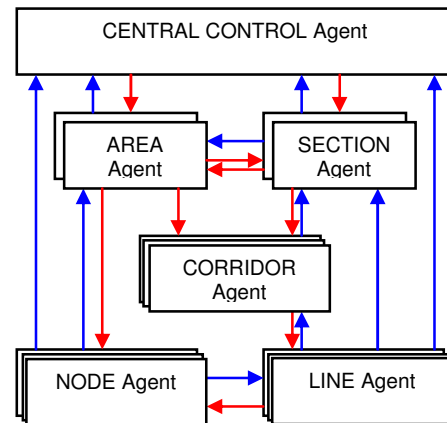


Figure 1. A functional architecture of an integrated defense system

An agent is a software abstraction providing a convenient and powerful way to describe a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks on behalf of its user. An agent is defined in terms of its behavior. In the context of Model Driven Engineering (MDE), agents are used as means for system structuring especially in the case of distributed, complex, evolving, and fast changing systems to ensure system flexibility and system robustness using predefined agent-oriented design pattern.

Until recently, the main focus of the multi-agent systems community has been on the development of concepts, architectures and interaction mechanisms, of multi-agent systems [3], [7]. In the following, we present how this approach can be turned into a formal methodology to design complex defense plan, composed of local and coordination rules, that specify agent behaviors and agent interactions.

III. DEFPLANS LANGUAGE FEATURES

Defense plan system spans over an entire national electric grid. It is composed of software and hardware functionality interacting by means of a suitable communication network. DefPlans language is devoted to specify and design defense plans for bulk power systems. It allows one to express the main characteristics of defense plans:

- strict real-time requirements deriving from the necessity of timely monitoring and controlling the electric system with limited communication delays and of managing the synchronization among all the involved components;
- strict reliability and availability requirements to guarantee an adequate level of communication also in case of partial or total line unavailability or in case of black out;
- need to support continuously the most critical functions, maintaining a predefined quality of service.

DefPlans is agent and rule based. An agent identifies a defense plan functionality which has a proper knowledge and is able to communicate with the other agents. The agent behavior is expressed by means of rules which set a system state when specific conditions are identified.

The rule based paradigm consists of: a *knowledge base*, a *working memory* and an *inference engine*. These notions are represented in DefPlans as follows: the knowledge base is constituted by the set of attributes and by the set of rules associated to agents. The working memory consists of the set of facts obtained from the evaluation of the rules. The inference engine is the logic according to the set of agents are cyclically processed. Here follows a description of the main DefPlans language keywords.

A. Agents and instances

An agent identifies a specific functionality, a component of a more complex control system. Such a component owns specific knowledge and inference abilities, and interacts with others in order to pursue a specific goal.

Control element

DefPlans allows one to distinguish between the class of functionality, representing the function general *schema*, from a specific *instance*. Let us consider the class of functions devoted to control transmission lines (hereinafter, language keywords are in bold face):

```
AGENT : class : Line_Controller
hierarchy_level : 1
END_AGENT_class
```

The Line_Controller agent class is associated to the first hierarchical level. Hierarchy identifies a precedence order for the evaluation of the agents. The lower the level the higher the precedence in the agent execution. The evaluation mechanism will be detailed in the next chapter. An instance of the Line_Controller agent identifies the functionality specifically operating on the line XXX of the grid, and it is defined as follows:

```
AGENT : instance : LC_XXX : class : Line_Controller
END_AGENT_instance
```

Grid Elements

To identify the components of the transmission network subject to the defense plan functionality DefPlans provides the notion of *grid element*. A grid element is a symbolic representation of the real grid component, DefPlans provides a set of parameters and a set of methods to acquire and to modify the grid element status, respectively. For example, the grid element (transmission) Line is modeled by the parameter instantaneous power P, and by methods open, and c_open to verify whether the line is open, and to command the breakers to put the line out of order, respectively.

B. Signature of attributes

An attribute specifies either a relevant characteristic of an agent or an agent state. The set of attributes constitute a knowledge base of the agent. An attribute is specified by the identifier, the type of the values it takes, a boolean flag to state the time dependence, or not, and a possible initial value. DefPlans provides the standard types (integer, double, boolean and string) and their temporal extensions (td_integer, td_double and td_boolean). Temporal extensions of integer, double and

boolean enable the attribute values to be referred to different time instants. This is especially suitable when the system is subject to real-time constraints. For instance, the agent dedicated to control of a tie line, owns two attributes to identify low and high thresholds of power P overflow: low_overload and high_overload, respectively, whose type is double and do not change the value once set. Two further attributes, P_low_over and P_high_over, are devoted to indicate whether the measured power P has exceeded the low or the high threshold, respectively. Their basic type is boolean. Though, as the control defense logic of the agent requires measuring the duration of the overloaded states, the type must be timed extended, that is td_boolean. The value of these two attributes change and their initial values are false (the value of P is under the thresholds). Formally:

```
AGENT : class : Line_Controller
hierarchy_level : 1
attribute : low_overload : double : false : - : .
high_overload : double : false : - : .
P_low_over : td_bool : true : false : - .
P_high_over : td_bool : true : false : - .
```

END_AGENT_class

C. Connection between agents

Connections are the way agents exchange information. A connection is a directional channel set between two agents, and allows referring the attributes in the connected agents.

A connection is specified by: the identifier of the agent class, the number of instances connected (either a natural number if fixed, or a variable denoting an unspecified number of instances, greater, or equals, to one) and a set of variable to identify the connected agent, if needed. Let's for example specify the agent to control a critical grid section (Section_Controller), connected to an agent controlling a grid tie (Line_Controller). In general, there is one Section_Controller and one, or more, Line_Controller.

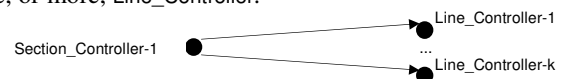


Figure 2. The instances of the agents belonging to the connection

Formally, the connection between Section_Controller and Line_Controller is specified as follows:

```
AGENT : class : Section_Controller
hierarchy_level : 2
connection : Line_Controller : Nline : lc1, lc2.
END_AGENT_class
```

The number of instances is indicated by the variable Nline, as it is not fixed, and two variables are declared to denote the set of connected Line_Controller.

Connections allow specifying the agent interaction with grid elements. In these cases a connection specification refers to the grid element identifier. In the example, the specification of the connection between Line_Controller and Line is given by:

```
AGENT : class : Line_Controller
hierarchy_level : 1
connection : Line : 1 : l1, l2.
END_AGENT_class
```

In the agent instance a connection is specified by the set of connected instances. Let us consider the instance of section controller SC1, and the instance of line controller LC_XXX, associated to the transmission line XXX, respectively. The connection specified into the section controller SC1 is constituted by one instance the agent instance LC_XXX.

```
AGENT : instance : SC1 : class : Section_Controller
      connection : Line_Controller : 1 : LC_XXX
END_AGENT_instance
```

The connection between the line controller and the grid element Line is constituted by one instance.

```
AGENT : instance : LC_XXX : class : Line_Controller
      connection : Line : 1 : XXX.
END_AGENT_instance
```

A connection can also refer to another connection. DefPlans allows specifying a chain of connections to refer the attributes of an agent not directly connected. This kind of connections is said *connection chain*. Let us specify for instance, the agent Area_Controller, connected to Section_Controller:

```
AGENT : class : Area
      hierarchy_level : 3
      connection : Section_Controller : Nsec : sect1, sect2.
END_AGENT_class
```

Within the agent Area Controller now it is possible to refer to the set of grid elements Line connected to, according to the following schema:

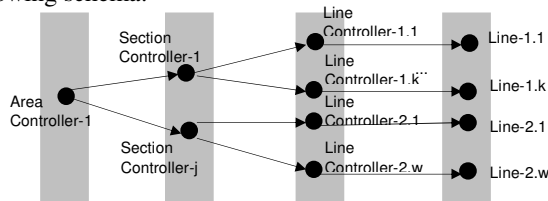


Figure 3. The set of instances belonging to the connection chain

This connection chain includes a number of paths, each one starting from the instance Area_Controller-1. The language provides a powerful mechanism according to two different connection chains denote, in general, different instances. Two connection chains are different if at least one agent class referred is different. As an example, let us introduce the agent Corridor, which is connected to the agent Line_Controller. At the same time, the agent Area is connected to the Corridor agent as follows:

```
AGENT : class : Corridor
      hierarchy_level : 2
      connection : Line_Controller : Nlin : -.
END_AGENT_class
```

```
AGENT : class : Area
      hierarchy_level : 3
      connection : Section_Controller : Nsec : sect1, sect2.
      connection : Corridor : Ncor : -.
END_AGENT_class
```

Connection chain specifications can be represented by means of the following graph:

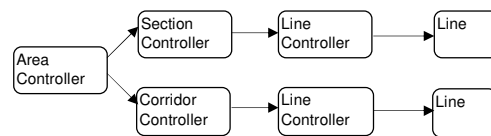


Figure 4. The graph to represent the connections from the Area agent to the Line grid element

In this case, two instances of Line_grid element connected to the same instance of the Area_Controller are different if they belong, respectively, to the paths: Area_Controller-Section_Controller-Line_Controller-Line and Area_Controller-Corridor_Controller-Line_Controller-Line.

D. Definitions of attributes

The specification of an attribute is completed by the possibility to define the value by means of a mathematical expression called *definition*. For integer and double types the definition is given by a mathematical expression, in which numerical constants and attributes are combined by algebraic functions to express a value. A number of suitable operators are predefined in DefPlans to manage those attributes defined in connected agents, as they denote a list of instances; among the *mathematical operators*:

- SUM: computes the sum of a list of numbers;
 - PRD: computes the product of a list of numbers;
- and among *list operators*:
- NUM: computes the number of elements of a list;
 - FST: gets the first element of a list.

An attribute definition can be also specified for boolean type attributes. In this case a *definition* is expressed by means of a boolean valued *clause expression*. For instance, let us specify that the boolean attribute P_low_over of Line_Controller agent is true when the power P measured on the controlled line overcome the threshold value.

```
AGENT : class : Line_Controller
      hierarchy_level : 1
      connection : Line : 1 : l1, l2.
      attribute : low_overload : double : false : -.
      high_overload : double : false : -.
      P_low_over :td_bool:true:false :FST[Line.P] > low_overload.
      P_high_over :td_bool:true:false :FST[Line.P] > high_overload.
END_AGENT_class
```

Clause expressions are also used to express conditions in the agent rules, the next section provides some examples.

E. Rules to specify behaviors

The core of the design of a defense plan consists in defining the set of actions to be undertaken as a consequence of phenomena or contingency that may occur. In DefPlans the behavior model of the agent is expressed by a set of rules. The "IF-THEN" rule schema has become the most popular form of declarative knowledge representation in AI tools. Knowledge represented as *if-then* rules are easily understood as they single out causes from effects; further they constitute a set of inference modules.

This set can be easily modified adding or removing single element. A rule is specified by: a *name* identifier, a *condition* and a *consequence*. As a condition is a logical expression, the rule evaluation states that the rule fires when the condition is logically true. In this case the set of facts held into the consequence are made true.

Basic elements of a rule

Condition and consequence of a rule are specified by means of *clause expressions*. Clause expression associated to a condition is a set of *boolean clauses* combined by means of the logical connectives: AND, OR. Clause expression associated to a rule consequence is a subset of boolean clauses.

Boolean clauses

A boolean clause (*bc*) belongs to one of the following cases:

a) *an agent attribute*

a boolean attribute, or a non-boolean attribute combined into an algebraic relation (e.g., =, >, <, etc.);

b) *an attribute of a connected agent*

an attribute defined in a connected agent (by either a simple or a connection chain).

c) *a timed expression*

a timed expression is a boolean clause of type a) or b) occurring in the scope of a timed operator. A timed operator allows one to shift the evaluation of a boolean clause in a time instant in the future, or in the past, with respect to the (implicit) current one 'i'.

The main timed operators are:

- *Future* : FTR(*bc*,*t*): (the boolean expression) *bc* is true in the time instant *i*+*t*;
- *Past* : PST(*bc*,*t*): *bc* is true in the time instant *i*-*t*;
- *Lasts* : LSS(*bc*,*t*): *bc* is true in the set of time instants *t'* such that $i \leq t' \leq i+t$;
- *Lasted* : LST(*bc*,*t*): *bc* is true in the set of time instants *t'* such that $i-t \leq t' \leq i$;
- *Within Past* : WTP(*bc*,*t*): *bc* is true in at least one time instant *t'* such that $i-t \leq t' \leq i$;
- *Within Future* : WTF(*bc*,*t*): *bc* is true in at least one time instant *t'* such that $i \leq t' \leq i+t$;

In the specification of the Section_Controller Agent, a rule is specified to state the section critical status when one of the three tie lines is out, and another one is low overloaded. First, a *boolean* attribute critical section to represent the critical status.

Further a rule to specify the behavior is specified:

```
AGENT : class : Section_Controller
hierarchy_level : 2
connection : Line_Controller : Nline : lc1, lc2.
attribute : critical_section : boolean : true : false : - .
rule : 1 :
    Nline = 3 AND NUM[Line_Controller.Line.out] = 1 AND
    EXT[Line_Controller.P_low_over] : critical_section.
END_AGENT_class
```

Clause expression in a rule consequence

The clause expression associated to the consequence of a

rule is a set of boolean clauses of type a) or b), or a boolean clause of type c) in which the timed operator occurring refers to the future in a conjunctive way. That is, the allowed timed operators in this case are: FTR and LSS.

Bounding of variables

The term identifying agent(s) in a boolean clause of type c) is a variable(s), denoting the set of instances which belong to the connection relation. For instance, the boolean clause:

```
Line_Controller.P_low_over
```

denotes the set of Line_Controller instances, connected to the current Section_Controller, whose power P has exceeded the lower threshold. Once this boolean expression is enclosed into the scope of EXT operator it is a *bound* boolean clause, that is the identifier Line_Controller is opaque to the rest of the boolean clauses. As an example, the following two expressions do not share instances each other:

```
NUM[Line_Controller.Linea.out] = 1 AND Line_Controller.P_low_over
```

Alternatively, DefPlans allows one to bound variables in order to relate the instances between two or more *boolean* clauses. In particular this allows bounding instances between condition and consequence. As an example, the control area agent monitors the overloading state of the controlled sections, and if load shedding is required it opens the line. Formally:

```
AGENT : class : Area
hierarchy_level : 3
connection : Section_Controller : Nsec : sect1, sect2.
rule : 1 :
    sect1.critical_section AND
    sect2.Line_Controller.Line.name = "NN2287" :
    sect2.Line_Controller.Line.c_open.
END_AGENT_class
```

A realistic exemplification of the DefPlans ability is provided in [10].

IV. VERIFICATION AND VALIDATION OF DEFENSE PLANS

The DefPlans verification and validation abilities are supported by an environment including these facilities:

- an *XML interface* to specify the set of agent classes and the set of agent instances;
- an *automatic translator* to get an executable program from the specification;
- an *inference engine* to execute the program as a module linked to the *Power System Simulator (PSD)*;

Translation of a defense plan specification

The translation is an automatic step performed to obtain an executable code from the specification. The translation is defined according to a set of specific rules, that put into correspondence each agent with a portion of an *object oriented programming* (OOP) code. The translation is defined assuming the inclusion of a suitable library of functionality (*lib*).

The translation rules are mainly defined by structural induction on the components of the agents. The translation is

defined exploiting the similarity stated between agent programming and object programming. In particular, the OOP notions of object, attribute and method have been put into correspondence, respectively, with the DefPlans agent, attribute and rules.

Multi-agent inference engine

The DefPlans inference engine manage the cyclic evaluation of a defense plan. In order to achieve this goal, the inference engine sets a scheduling list of the agents and provides a number of library functions to support the evaluation of each agent for the current cycle.

In the scheduling list agents are upward ordered with respect to the hierarchy level. The lower the hierarchy level the first the agents are placed into the list. Within the same hierarchy level there is no order between agents.

For each evaluation cycle, each agent is selected from the scheduling list, according to the order, and then it is executed. The agent execution consists to:

- evaluate each attribute definition in order to update the attribute value;
- evaluate the set of specified rules.

As the upward evaluation of agents is completed, all the required information are acquired. That permits the higher level agents to set the best plans and actuate them on the whole power system.

The settlement schedule list is fixed once and for all. However, the policy to evaluate agents can be modified just revising the access to the scheduling list. As an example, we are currently studying the effects of an upward-downward policy on the agents abilities. Indeed, this policy allows both: to gather the power system status (upward evaluation) and to collect the set of plans defined by the coarser level agents (downward evaluation) in order to optimize the actions to be performed.

The inference engine just depicted is mainly tailored on the verification and validation of the defense plan, rather than the deployment of the defense plan on the power system. The DefPlans execution abilities presented so far are centered on the specification. The deployment of the defense plan on a real power system will be the subject of a next research activity.

Validation of a specification

The validation of a defense plan has the role to measure the ability of a set of actions to prevent blackout of a power system in a specified status. In particular, the main aim of this phase is to test whether the *defense plan* is able to:

- recognize the event, or the network status, that can lead to the complete blackout;
- plan and apply suitable actions in order to confine criticalities, as far as possible;
- allows the fast restoration of the faulted power system

components with respect to different power system configurations. Verification and validation phases couple the power system simulation cycle and the defense plan simulation cycle in order to constitute a single *simulation cycle*.

V. CONCLUSION

The paper presented DefPlans, an integrated environment for the specification and validation of emergency control schemes of the bulk power systems. DefPlans is based on the notions of: agent, rule to represent the behavior, real time constraints thanks to suitable timed operators. Such a proposal can help the verification and validation of defense plan functionalities.

Once the validation step has given the expected results, it is possible to proceed to the design and implementation of defense systems. This further step requires some adjustment on the code as it must be distributed on a wide area architecture.

Currently, DefPlans does not provide a distributed agent architecture. To bridge the gap between analysis and real implementation on the national power system the translation step described should be integrated such that the agent distribution can be gained. For instance, to this aim, we currently taking into account the possibility to adapt the JADE environment.

REFERENCES

- [1] J.P.Bigus, J. Bigus, "Constructing Intelligent Agents Using Java", Wiley, Second Edition, 2001.
- [2] "Defense plan against extreme contingencies", Task Force C2.02.24, April 2007.
- [3] Hao Li, G. W. Rosenwald, Juhwan Jung, Cheng-Ching Liu, Strategic Power System Infrastructure Defense, in Proceedings of the IEEE, volume 93, n. 5, May 2005.
- [4] J. Jung, et al., "Wide Area Protection and Control Using a Strategic Power Infrastructure Defense System", CIGRE 38-104 session 2002.
- [5] J.McCalley, W. Fu, "Reliability of Special Protection Systems", in IEEE Transaction on Power Systems, vol.14. No. 4. November 1999.
- [6] J. McCalley, W. Fu, "Chapter 6: Reliability of Special Protection Schemes," in "System Protection Schemes in Power Networks," by Task Force 38.02.19, January, 2001.
- [7] J. McCalley, Z. Zhong, V. Vishwanathan, and V. Honavar, "Multiagent negotiation models for power system applications," in "Autonomous systems and intelligent agents in power system control and operation," C. Rehtanz, editor, Springer-Verlag, Berlin, 2003, pp. 49-74.
- [8] X. Wang, W. Shao, V. Vittal, "Adaptive Corrective Control Strategies for Preventing Power System Blackouts" in Proceedings of 15th PSCC, Liege, August, 2005.
- [9] L. Wehenkel, "Emergency Control and its Strategies", in Proceedings of the 13th Power Systems Computation Conference, PSCC99, page 35-48 - June 1999.
- [10] E. Ciapessoni, E. Corsetti, "Mitigazione dei rischi per il sistema elettrico: monitoraggio dello stato di sicurezza e nuovi strumenti di analisi", in the RdS report n. 08005930.
- [11] IEA, "Learning from the blackouts", 2005.