Abstract—Critical infrastructure (CI) modeling and analysis is a very challenging research topic. One of the most pressing issues is to find an effective representation for addressing the system vulnerabilities caused by interdependencies, which, if exploited, could result in nontrivial accident scenarios. Until now, this question has been tackled for different sector-specific infrastructures (electricity grid, telecommunications networks, supply chains, etc.), and very few generalizable analysis tools have been developed. However, all CI share some features that can be leveraged in order to build a common modeling framework. This paper identifies these common features, which it exploits to develop a modeling language: the infrastructure resilience-oriented modeling language (I®ML). I®ML is designed to facilitate the analysis of operational interdependencies among the infrastructure components and overall resilience, i.e., the ability of the infrastructure to withstand and recover under off-nominal (anomalous) conditions. A number of examples are used to illustrate the modeling concepts and highlight the analytical capability of I®ML.

Index Terms—Critical infrastructures, interdependencies, resilience, risk, system of systems (SoS).

I. INTRODUCTION

MODERN infrastructures are networks of heterogeneous systems (telecommunications, power grids, transportation, etc.), which interoperate in order to provide services on a large scale. These infrastructures are classed as critical infrastructures (CI) when their failure is liable to affect strategic assets and have repercussions on the economy and society at large. European Council Directive 2008/114/EC [1] on critical infrastructure protection (CIP) define a European infrastructure as critical when risks are shared by at least two countries. This definition identifies the principle of cross-border risks that are beyond the control of national regulatory authorities. A similar directive on the identification and protection of CI in the U.S. was approved by the U.S. Department of Homeland Security (DHS) [2] in 2003.

The risks to CI are not easy to address using current systems analysis methodologies. An infrastructure is not an ordinary system because of its internal organization and operation. In this context, infrastructure design is not an applicable term. An infrastructure behaves like a whole, in which each system component directly or indirectly depends on the others. This is the result of a bottom-up process that ends with the aggregation (rafting [3]) of single components. In most environments, information and communications technologies (ICT) play the role of coordinators, orchestrators, controllers, and supervisors. In this respect, a CI has to be regarded as a special instance of a system of systems (SoS), i.e., “an assemblage of components which individually may be regarded as systems and which possess two additional properties”: operational and managerial independence of the components [4]. An assessment of the maturity level of this topic suggests that, despite some theoretical groundwork, good systems modeling and analysis practices are still wanting. An applicable analogy would be the rather informal software architecture practices and notations used in the field of software engineering during the 1970s and 1980s by contrast with the emergence of software architecture as a standalone discipline in the 1990s. This was a significant advance, which brought with it a further clarification and reorganization of the issues and paved the road to future developments [5], [6].

The main challenge of infrastructure system analysis is to find a representation that is able to accommodate heterogeneous components in the same modeling framework. This is important for supporting the analysis of overall infrastructure behavior and must be accompanied by a set of quantities related to structural and dynamic properties. Resilience stands out as one of the quantities of major interest. Definitions of resilience are specific to the application context, and the lexicon will often change too. The U.S. DHS [7] defines resilience as the “ability to adapt to changing conditions and prepare for, withstand, and rapidly recover from disruption.” If applied to infrastructures, resilience is the ability to endure all hazards (i.e., survive and recover) while guaranteeing an acceptable level of service. This definition is further developed in [8], including the following set of resilience measures: 1) avoidance by prevention; 2) absorption; and 3) recovery. The three measures correspond to three different time periods—before, during and after the incident—at which resilience attributes may materialize. These measures can be allocated to either human, technical, or organizational levels [9].

This paper presents a modeling framework for analyzing interdependent modern infrastructures in order to inform
either operators or decision-makers of system vulnerabilities and overall resilience. The framework consists of a modeling language and a set of analytical tools. The infrastructure resilience-oriented modeling language (I®ML) provides a high-level representation of a SoS through a set of elementary building blocks: services, systems, agent domains and resources. The focus of an I®ML model is on the operational relationships among the components: provider/user, producer/consumer, and controller/controlled. The control relationship is also a key I®ML element, as it explicitly accounts for a component that controls another component in pursuit of a particular goal.

An I®ML model can be further transformed in order to derive intra and intercomponent functional dependencies that characterize the infrastructure behavior. An I®ML model is transformed into the goal dependency structure (GDS). The GDS is a directed graph that provides a conceptual and functional representation of the infrastructure with its goals. A goal is the target outcome of systems that interact with each other. Even so, goal achievement will also depend on the functional dependencies with other goals.

Both structural and dynamic properties of the infrastructure can be analyzed using the GDS. A structural analysis of the GDS yields the most critical and vulnerable goals. A dynamic analysis assesses the capability of the infrastructure to withstand, respond to and recover from the propagation of a disturbance, that is, its resilience. The modeling scope of the framework is comprehensive, and it offers a standalone set of analysis tools, ranging from the initial description stage to the final assessment. Because the conceptual modeling is based on functional dependencies, the scope is theoretically unbounded. Large-scale accident scenarios, which involve many diverse infrastructures, are within the scope of the framework. The results of the analysis are amenable to operators and decision-makers, providing a preliminary instrument for prioritizing risks and deciding how to protect the most critical assets.

The paper is structured as follows. Section II provides a short introduction to system analysis methodologies that are applicable to SoS and infrastructures. Section III presents the I®ML modeling features and its building blocks. Section IV introduces the GDS and the tools for analyzing structural properties and resilience. The methodology is applied to a case study in Section V. The conclusion is discussed in Section VI.

II. MODELING CI

A. Introductory Survey

Infrastructure modeling calls for a trade-off between the level of detail of the representation and the effort required to conduct the analysis. On one side, a comprehensive description depends on the provision of several views. On the other, there are behaviors that will never emerge from the separate study of the parts [10]. See [11] for a survey and [12] for a historical overview and appraisal of the literature.

The main obstacles to the application of existing system analysis methodologies to infrastructures are as follows.

1) Poor consideration of cross-sector interrelationships and systemic issues [3].

2) Definition of rigid and artificial system boundaries that delimit the scope [13].

3) Impossibility of representing conflicting goals (e.g., trade-offs) [14].

The first obstacle is based on the assumption that the behavior of the whole system can be understood by separately analyzing its parts, which are merged to form an overall picture. This assumption is correct for well-structured complex systems and is the dominant model in engineering (e.g., divide-and-conquer approaches). Phenomena are subject to well-known patterns, which specialize to the sector of reference (e.g., electricity, transportation, gas, etc.) [13], [15]. In such cases, system analysis is comprehensive and systematic. An example is probabilistic risk assessment (PRA), which is a standard methodology for nuclear power plants [16]. Unfortunately, these models are not able to deal with events that occur as a result of nontrivial interdependencies, typically in SoS and CI. The second obstacle is that a complex networked infrastructure hardly ever responds to either a set design or a classical life cycle. Rather it evolves with respect to a myriad of inputs and constraints, some of which are unpredictable and are driven by users, operators, markets, and so forth [17]. The third obstacle will probably require out-of-the-box thinking about the problem.

Clearly, any methodology anchored to a specific sector will be unsatisfactory. Indeed, many scientific contributions to the problem of modeling CI advocate the adoption of a holistic or systemic viewpoint. The systemic approach looks at the system as a quasi nondecomposable entity, whose behavior cannot be inferred from the separate representation and analysis of its parts, but emerges from their interactions [18]. As such, it unveils global properties, such as safety and resilience [19], [20]. Decision-makers require such a broader scenario, in which high-level abstraction takes precedence over details. Several contributions were based on these scientific intuitions. The SimCIP simulator, developed as part of the integrated risk-reduction of information-based infrastructure systems (IRRIIS) project [21], was designed as a federated simulator environment that integrates simulations of different CI through their interdependencies. ResiliNets is a model for communication networks, and especially the Internet [22]. Examples of other systemic approaches grounded on a hypergraph-based notation are reported in [23] and [24]. Research reported in [25] mixes holistic and reductionist approaches in the context of the MICIE project. Large-scale simulation platforms modeled on the IEEE Standard for modeling and simulation high level architecture (HLA) [26] also broaden the scope of the analysis. The HLA accommodates different types of specialized models (views) within a comprehensive framework. Models are simulated in their specific environment, while exchanging data at run time. Nontrivial computational resources are necessary for this purpose.

Entirely holistic principles are applied in systems theoretic accident model and process (STAMP), a framework for safety analysis [27]. STAMP is interesting because it provides a technical and organizational system representation, which is a useful feature for CI. The functional resonance analysis method (FRAM) is another integrative (holistic-oriented) modeling
framework [28]. FRAM introduces the concepts of functional resonance and emerging behavior for assessing system resilience. Johansson and Hassel [29] proposed another noteworthy systemic approach to interdependency modeling based on a failure/repair model. This approach returns nodes that have most impact on the resilience of a particular service in the presence of graceful (not simply black-or-white) degradation.

B. Modeling Language Objectives and Features

The problem of modeling and analyzing CI can be formulated as follows.

“Develop a single, unified framework able to model the multiplicity and variability of interdependent networked infrastructures in such a way that is amenable to decision-making analysis in order to improve overall resilience.”

Existing contributions, of which a fairly complete list was given in Section II-A, suggest several options for dealing with this problem and particularly its complexity and mathematical treatment. Generally, these approaches tend to prioritize the holistic paradigm over reductionism, and integrative over sector-specific description. Abstraction and conceptual descriptions also play an important part in model building for the purpose of broadening the scope of the analysis. Our proposal aims to address all these features in a single modeling framework with the intention of describing, understanding, and evaluating the impacts of nontrivial hidden vulnerabilities that may propagate through the network dependencies and escalate into accident scenarios [8]. The modeling framework that we propose should be as follows.

- **O1**: Model infrastructure interdependencies.
- **O2**: Support resilience-oriented analysis.
- **O3**: Return results amenable to decision-making.

O1, O2, and O3 prescribe an integrative, holistic, and assessment-oriented model that focuses on resilience. The three objectives respond to the need to consider diversity in sectors and technologies that are typical of networked infrastructures. In particular, O1 dictates that the modeling language should provide a bird’s-eye view of the infrastructure and its interconnections, which are necessary for the infrastructure to operate but are also the means for propagating local disturbances [30]. O3 dictates that decision-makers are the priority users, and this has implications for the nature of the descriptions, which must be conceptual [31]–[33]. Conceptual modeling has been used for decades in software engineering [33]–[35] in order to provide an understanding of complexity, as well as in safety engineering. Implementation details are independent of conceptual approaches like HAZOP [36], for example, which provides thinking aids for discovering potentially hazardous events during operation that may result in accident scenarios. Similarly, I®ML will strike a balance between a detailed description of reality and the necessary abstraction to manage complexity.

The three objectives are achieved by the following modeling language features underlying I®ML.

- **F1**: represent agents in socio-technical systems.
- **F2**: represent resources.
- **F3**: represent services.
- **F4**: represent control relationships.
- **F5**: represent producer-consumer relationships.
- **F6**: represent provider-user relationships.
- **F7**: represent (inter)dependencies among services.

F4–F7 deal with the representation of, in this case, operational interdependencies. These modeling features provide the necessary elements for the conceptual representation of a SoS. Table I maps the seven features to the above three objectives.

### III. Modeling and Analysis with I®ML

#### A. Modeling Language

I®ML represents an infrastructure as an aggregation of heterogeneous systems that interoperate through their relationships. The language is nonsector specific and, unlike other modeling proposals, such as those reported in [18], does not suggest any organization of the model in layers, such as physical, intermediate (cyber), and application layers. The language satisfies the objectives and has the features outlined in Section II-B. I®ML is a graphical language that is reminiscent of UML [34] or, more specifically, SysML [37], which is the UML adaptation for complex systems. As specified in the previous section, however, I®ML is applicable for analysis and not for design. I®ML has a library of elementary building blocks: domains, systems, and services, plus interconnection elements. Table II summarizes these components, which are described below. The on-scene care example shown in Fig. 1 will be used to illustrate the language elements.

Jackson [38] described a domain as “a set of phenomena that is usefully treated and represented as a unit in problem analysis.” A domain conforms to the infrastructure description, as it is able to preserve the specific identity and diversity of the constituent elements. I®ML defines two types of domains: agent and resource domains. Agent domains are active elements, e.g., with control and decision-making ability. An agent domain can take into account the physical, abstract, or organizational points of view. I®ML represents agent domains as square boxes. Examples of agent domains are purchases or ambulance maintenance in Fig. 1.

Resource domains are passive elements that provide a physical quantity, e.g., goods and materials, which can be stored and consumed. A resource has a state, and this information is
available to the resource consumer, which is an agent domain. The state ranges from full to empty or from available to unavailable. I©ML represents a resource as a cylinder-shaped element. Examples of resources are ambulances and historical data in Fig. 1.

An agent domain can be connected to another agent domain or to a resource. There are no direct connections among resources, as these are mediated by an agent domain (for example, an agent domain responsible for moving materials from one resource to another or similar operations). The different types of connections will be discussed in turn below.

Agent–resource connections are useful for expressing how an agent domain provides materials that are stored in a particular resource or how an agent domain relies on a resource. An agent domain may: 1) feed the resource or 2) use the resource. In this respect, the arc that connects an agent domain to a resource represents the addition, removal or update of elements that the resource contains. Any relevant material exchange will always be represented by some accompanying information just as the information specified on a delivery note represents an exchange of materials (books, clothes, etc.). Fig. 1 illustrates an agent-resource connection between purchases and ambulances.

Agent–agent connections support the exchange of information among the elements that together pursue a particular goal. As mentioned above, this exchange of information may also be related to an exchange of materials. Two interconnected agent domains establish a control relationship in I©ML. The control relationship is denoted by a filled “•” on the controller side and an empty “○” on the controlled side. Fig. 1 shows that the connection between demand analysis and contracting is a control relationship.

I©ML control relationships are conceptually based on the control paradigm in systems theory [27]. A connection between agent domains implies that there is a goal, an effector- and sensor-based goal achievement method, and (if applicable) knowledge of the controlled elements and/or their operating environment. Consequently, four attributes can be associated with each control relationship. They are control goal (a control system assures that the target, e.g., a “set point,” is achieved); effectors (the means to act on the controlled agent domain); sensors (the means to perceive the current state of the controlled agent domain); and internal model (the model of the controlled agent domain). Note that control effectors can be either physical devices or departments, which impose procedures. Similarly, sensors can read physical data, as well as...
job reports, incident reports, after-action reviews, etc. A control relationship may also be just an agent domain that reads from or writes to another agent domain.

Agent and resource domains are arranged together to constitute a system, which, in turn, provides a service.

A system is a set of agent domains and resources that collaborate with each other. A system provides one or more services, which may be qualitatively different. I®ML represents a system by a dotted line that encloses the agent domains. Fig. 1 shows two systems: the ambulance management system and the incident management system.

A service is the result output by a system or by a set of interconnected systems. A service is represented by a rounded corner box and is connected to the provider system. A service may also rely on another service, which is another type of connection available in I®ML. Service interconnections are provider-user relationships and not control relationships. Rounded corner boxes, such as incident information in Fig. 1, are examples of services. In this case, the service is provided by the incident management system. Fig. 1 also shows services that depend on other services. Some such services (e.g., 112 emergency calls) are not further developed, because they are also provided by another system that is beyond the scope of this particular analysis effort.

B. How I®ML Meets Requirements

Table I summarizes the requirements presented as objectives and features in Section A. The “+” sign indicates that a particular feature helps to achieve a particular objective. I®ML also includes the seven modeling features listed in Section II, summarized in Table II with their respective I®ML graphical elements. The model features achieve the objectives as follows.

I®ML achieves O1 because a model that is able to represent agent domains, resources and how they fit together in a service provision system provides a bird’s-eye view that is not constrained by a particular technology and does not discriminate different (structural, behavioral, organizational, etc.) viewpoints, as the model is neutral in this respect.

O2 addresses resilience. To account for resilience, it is necessary to take into account a system’s ability to absorb disturbances and later restore normal conditions. This definition applies to SoS too. In this respect, the control relationships are equivalent to control relationships and dynamic dependencies (features F4 and F7). Exploration of possible control elements, such as effectors and sensors, malfunctioning, or internal model inaccuracy (with regard to the actual state) will help analysts to identify the causes that lead to off-nominal behavior. A similar reasoning applies with respect to resources availability (F5).

Finally, I®ML achieves O3 because it is able to represent the control relationships and the within-service connection network.

C. How to Build Models With I®ML

An I®ML model is built top-down. It starts by identifying the services, followed by the constituent systems, and finally the agent domains and resources within each system. This is a cyclic five-step procedure, which are as follows.

1) Identify services.
2) Establish connections among interdependent services.
3) Use agent and resource domains to model the system and its internal representation for each service, if applicable.
4) Identify the control relationships for each agent domain. To do this, address the following questions: Is this a controller or a controlled agent domain? If it is a controller domain, which domain does it control? What are the goals of the controller and controlled agent domains? This outputs pairs of agent domains in a controller-controlled relationship. Remember that a controller agent domain in one relationship can be also a controlled agent domain in another relationship with a third domain. Draw a box around sets of domains that have been connected via controller-controlled relationships. This defines the system providing that service.
5) Address the following question: does this system rely on any service(s) provided externally? If so, connect the

---

**TABLE II**

**MODELING FEATURES**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Graphical element</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>I®ML provides the elements to represent agent domains</td>
<td>Square box</td>
</tr>
<tr>
<td>F2</td>
<td>I®ML provides the elements to represent resources</td>
<td>Cylinder</td>
</tr>
<tr>
<td>F3</td>
<td>I®ML provides the means to represent the services provided by a collaborating set of agent domains, i.e. a system</td>
<td>Oval linked to a boundary line that delimits a particular system</td>
</tr>
<tr>
<td>F4</td>
<td>I®ML provides the means to represent control relationships among agent domains</td>
<td>Arc linking two agent domains, with filled dot on the controller side and empty dot on the controlled side</td>
</tr>
<tr>
<td>F5</td>
<td>I®ML provides the means to express the resource production and consumption by connecting agent-domains to resources</td>
<td>Directed arc linking an agent-domain and a resource</td>
</tr>
<tr>
<td>F6</td>
<td>I®ML provides the means to represent service usage and provision by connecting a delimited set of collaborating agent domains (a system) to a service</td>
<td>Dotted lines enclosing a set of agent domains represent systems. Services are connected to systems.</td>
</tr>
<tr>
<td>F7</td>
<td>I®ML provides the means to express that some services rely on other services</td>
<td>Directed arc from one service to another</td>
</tr>
</tbody>
</table>
system to the identified service(s). If the service has not yet been identified, then go back to step 1.

This process is repeated for as long as it makes sense to do so. Analysts may decide to quit in order to avoid infinite regression (multiple calls of the same service).

IV. ANALYSIS TOOLSET

A. GDS

The GDS is a graph-like structure, derived from an I®ML model, which provides an overall view of the infrastructure dependencies. GDS nodes and arcs represent goals and operational dependencies among goals, respectively. A goal is associated with service provision, a specific function delivered by an agent domain and resource availability. The GDS provides for two arc types.

1) Depends-on arc: This relationship represents a dependency where the achievement of one goal relies on another goal. By convention, the tip of the arc points to the dependent node.

2) Consists-of arc: This relationship is used to delimit a set of goals within a system, which are needed to provide the service (which is another goal).

In this way, the GDS is suitable for computerized analysis. Note that goal dependencies do not constitute a task-subtask ordering, unlike a classical goal tree diagram where subgoals are necessary in order to achieve higher goals [39], [40].

The GDS is derived from the I®ML model after a transformation. Trivially, a one-to-one transformation is possible and will convert every element of I®ML into a graph node. Generally, though, the transformation may bring together elements and relationships into a single goal, outputting a more compact, modular representation.

The GDS is derived from the I®ML model in a bottom-up fashion as follows.

1) Identify goals within each system.

2) For each system:
   a) build the dependency graph by connecting the goals with depend-on arcs;
   b) identify output dependencies that the system provides to other systems, or services;
   c) identify input dependencies that the system receives from other systems or services.

3) Connect all service goals that depend on each other.

Fig. 2 shows part of the GDS of the on-scene care example derived from the I®ML in Fig. 1.

B. Structural Analysis

The scope of structural analysis is to assess GDS node (i.e., goal) criticality, vulnerability, and degree of interdependency [30]. These attributes are derived by analysis of the topology. The following definitions are as follows.

1) A criticality set of a goal G is the set of goals that depend on this goal.

2) A vulnerability set of a goal G is the set of goals on which it depends.

Because of the transitive dependency relationship, the criticality and vulnerability sets can be derived by running through the graph starting at any given goal. Whereas criticality shows the set of goals that may be affected by a cascading disturbance from a goal that is assumed to be unachieved, vulnerability represents the goals that may be at the origin of that goal disturbance. The two sets are normally disjoint, unless there are interdependencies.

For every criticality and vulnerability set, it is possible to define the coupling coefficient. The coupling coefficient is calculated as the inverse of the average distances from the goal under examination to the goals in its criticality (or vulnerability) set. As the graph is not weighted, the distance is defined as the number of arcs that separate two nodes (goals). If there are more paths, the shortest one is taken. Given a set of n nodes, the coupling coefficient will depend on the particular layout of the criticality or vulnerability set; this will range from 2/(n+1) for a chain layout to 1 for a star layout. This graph is particularly useful for identifying the goal that is most closely coupled with respect to either criticality or vulnerability. A shorter distance among nodes will correspond to a higher coupling and vice versa.

Structural analysis is also able to identify the interdependencies in the GDS. Any interdependency is always associated with the existence of a loop in the GDS. In a loop every goal...
depends on the others, and ultimately on itself. The number of goals in the loop is defined as the loop order. A simple heuristic holds: smaller-order loops show a tighter interdependency among goals than higher-order loops. Nonetheless, these loops are easier to identify, as they are composed of goals that are near neighbors. On the contrary, a loop with many goals may remain undiscovered and, consequently, unprotected.

C. Resilience Analysis

Resilience analysis is concerned with the behavior of the infrastructure upon the failure of one (or more) of its constituent elements. The analysis is based on “what-if” (forward-inductive) reasoning and is conducted on the GDS. The first step is to select a goal, which is assumed not to be achieved for a given time interval. This is the disturbance that may propagate throughout the GDS. The system response to the disturbance will be characterized by inertia to fail and by other affected goals’ promptness of recovery. They account for absorption and recovery resilience measures, respectively [9]. Absorption measures counteract any propagation of disturbances that materialize at the input dependency. They retard their negative effects somewhat depending on the amount of available local resources (e.g., redundancy, buffering, spare, etc.). Recovery measures are able to make a failed component work again in order to achieve the respective goal. By time tracking the response for a given disturbance, it is possible to foresee the failure propagation and expansion throughout the GDS. The severity of the scenario may be quantified with respect to the duration of a service outage (i.e., time to goal achievement) and the propagation depth (i.e., the number of affected goals). Propagation depth is theoretically bound to the criticality set of the goal that is the source of the disturbance.

A resilience analysis can be either qualitative or quantitative. Qualitative analysis is similar to model checking [41]: all possible system responses are generated from the failure of a goal in the GDS. Each system response is a resilience scenario, i.e., a sequence of failure and recovery events from an initial state to an end state. A plethora of scenarios may be generated, classified as: 1) recoverable scenarios, for which the initial state is restored and 2) nonrecoverable scenarios, for which the infrastructure is not resilient. The model has to be simulated in order to check if nonrecoverable scenarios are possible. Quantitative analysis simulates the response of the infrastructure. It requires the assignment of model parameters; the time to failure propagation (inertia to fail) and time to recovery for every goal in the criticality set. In the presented resilience analysis, we decided to assign a binary state, i.e., achieved or unachieved, to every goal. In a more general problem set-up, different achievement values may be defined, thus obtaining a multistate representation. This assumption will unnecessarily complicate the model further. Finally, scenario simulation may be either deterministic or probabilistic. In probabilistic (e.g., Monte Carlo) simulations, the model parameters are assumed to be random variables, and the simulation will return the likelihood of nonrecoverable scenarios. Sensitivity analysis can be performed in order to discover the infrastructure breakpoint, i.e., the maximum tolerable disturbance beyond which the infrastructure is not resilient.

Resilience analysis is a forward-inductive process, starting with the cause and outputting the possible consequences. Nonetheless, the reasoning can be reversed in order to trace the causes back from the consequences (backward-deductive reasoning). In this case, the vulnerability set of the analyzed goal will be the starting point for an accident analysis, playing the equivalent role to the criticality set in the resilience analysis. Given a goal failure, it will be possible to backtrack to the initiators, i.e., the goals that are understood to be behind the accident.

V. CASE STUDY

A. System Description

The case study is taken from NISTIR 7628 Guidelines for Smart Grid Cyber Security: Vol. 1 published by the U.S. National Institute of Standards and Technology [42]. The document provides a high-level description of a smart grid architecture split into seven areas containing elements that collaborate and communicate (i.e., actors) by several means and through a variety of interfaces. The smart grid areas are: 1) transmission; 2) distribution; 3) operations; 4) bulk generation; 5) markets; 6) customer; and 7) service.

The scope of the case study is limited to the transmission, distribution and customer areas, and includes the actors that collaborate to prevent energy disruptions and provide a balanced energy flow. Internet dependencies are also introduced. These add-ons are not reported in the NIST documents and are derived as outlined in [43].

The I@ML model of the smart grid is shown in Fig. 3. The elements in red were not extracted directly from the NIST document. They were added to include the Internet dependency of some elements. This leads to a more interesting and relevant analysis. The element (actor and interface) names and identifiers used in the NIST document are unchanged for the purpose of cross-referencing. The first system is load management (customer). The system provides the “manage load during peaks/emergencies” service and depends on the “operations support,” and “keep desired level” services. It consists of five agent domains.

1) LMS/DRMS (#32): Load and demand-response management system. It sends load management commands to customer appliances, such as commands to increase or decrease customer load. These commands are managed by the customer energy management system (agent domain #5) and transformed into local commands at the customer site (U106). Status information is provided by the customer information system CIS (#23).

2) CIS (#23): Customer information system. It manages the relationships of the public utility with its customers. It provides customer pricing and load information to the LMS/DRMS (#32).


Data are sent to the EMS (#5).

The NIST document refers to these areas as domains. NIST domains do not have the same meaning as I@ML agent domains. The agent domains referred to in this paper are actually equivalent to the NIST actors.
4) **EMS (#5):** Customer energy management system. It receives data provided by the meters (#8, U41) and commands from the LMS/DRMS (#32, U106). The system manages energy consumption (U44) at the customer end (#3).

5) **Customer End (#3):** Customer appliances and equipment. All customer electric appliances belong to this domain. The energy consumption is controlled (U44) by the customer energy management system EMS (#5).

The transmission system provides the service that keeps desired level of voltage within the parameters established by the operator. It depends on the load management during peaks/emergencies and operation support services. It consists of four agent domains.

1) **Transmission IED (#46):** Transmission intelligent electronic devices. They read the data from the transmission engineering equipment (#49). Their goal (U81) is to maintain the voltage level by sensing anomalies and to send voltage management commands (or tripping circuit breakers). They also receive data read from the transmission engineering status (U135).

2) **Transmission Supervisory Control and Data Acquisition (SCADA) (#37):** It sends commands to the transmission RTU (#47) for controlling power system equipment and manages energy consumption (U82).

3) **Transmission RTU (#47):** Transmission remote terminal unit. This unit cooperates with SCADA (#37) by sending status information on equipment and transmitting commands (received from SCADA) to field equipment (U136) to transmission engineering.

4) **Transmission Engineering (#49):** It consists of equipment between conductor lines, designed for more than 345 000 volts. Its status is monitored by the transmission IED (#46).
The distribution system provides the regulation of energy consumption service and depends on the managed outages service. It consists of the following agent domains.

1) Distribution Engineering (#26): Agent domain for planning and managing the distribution system design or upgrading (addition of new customers, new settings for capital investments, etc.). This agent domain is relevant for this model because it is distribution engineering that reports new settings (U109) to the distribution RTUs/IEDs (#15).

2) Distribution RTUs/IEDs (#15): Remote terminal unit and intelligent electronic devices. This domain cooperates with distribution generation (#25) on goal (U137) by generating commands to compensate frequency and voltage anomalies.

3) Distribution SCADA (#29): Distribution supervisory control and data acquisition. It acquires and transmits distribution device status (U117) and manages consumption by controlling (U65) the distribution generation and storage management (#25) agent domain.

4) Distribution Generation and Storage Management (#25): Generation of electricity from small energy sources. It reduces the amount of energy lost in transmission, as power is generated near to where it will be used, helping to regulate energy consumption (i.e., the overall goal of the distribution system).

The Internet elements (in red in Fig. 3) are mainly related to the distribution and the access communication networks and include the following services.

1) Online Communication at the Distribution Layer: It is used to provide two-way communication between the SCADA client and the server side.

2) Online Communication at the Access Network Layer: It represents the Internet communication used by IP-enabled meters.

The smart grid uses these services via the distribution SCADA agent domain, which is partitioned into two agent domains (client and server) belonging to two different systems in Fig. 3. The transmission SCADA accounts for the server side only. The client side has been moved to a system named SCADA clients. This client/server model for SCADA is taken from [44]. The new services are as follows.

1) SCADA Commands: They represent the SCADA commands from the SCADA clients.

2) SCADA Feedback: This represents the distribution service’s obligation to provide feedback to SCADA clients and operators.

In the I®ML model, both the SCADA distribution and SCADA transmission agent domains in the distribution system account for the server side only. The client is now an independent agent domain within the SCADA clients subsystem. Two-way client/server communication is modeled explicitly by two separate commands and feedback services, both of which rely on Internet communication at the distribution layer.

The model shows an important dependency of the smart grid on the Internet, which is on the SCADA client side. A trivial dependency is that SCADA devices (e.g., PCs) rely on the power supply, and therefore on the grid load management stability. A more complex, albeit predictable operational dependency loop runs from the SCADA client via the SCADA commands keep desired level, manage load ..., and SCADA feedback services and back to the SCADA client again. More interestingly, the SCADA client depends directly on the manage load...service. This dependency is located outside the above loop. This service may be affected by disturbances in the online communication at the access network layer service, which will potentially have an impact on SCADA client behavior. Similar nontrivial (inter)dependencies can be discovered and analyzed through the I®ML analysis tools described in the following sections.

B. Structural Analysis

The GDS in Fig. 4 is derived from the I®ML model in Fig. 3. Nodes are labeled with numbers and refer to specific goals, which are described in the text to the right. The structural analysis of the GDS returns the following quantities: 1) criticality and vulnerability sets and 2) coupling coefficient.

Results are summarized in Table III, and Fig. 5. Goals G4 and G10 have the largest criticality and vulnerability set, respectively, whereas G3 has the highest criticality coupling and G5, the highest vulnerability coupling. These results are not contradictory, as they are based on different metrics. For instance, targeting G4 is likely to cause the biggest damages, whereas targeting G3 will spread failure propagation faster. The structural analysis also identifies the interdependencies (i.e., loops) among goals. In the given example, there are four differently ordered (i.e., number of goals) loops in the GDS: (G2, G3), (G3, G6, G7), (G5, G6, G7), and (G3, G5, G6,
G7). G3 is a member of all loops, which makes it the most interdependent of all the nodes in the GDS.

C. Qualitative Resilience Analysis

As specified in Section V, a disturbance is applied to a goal, which challenges the infrastructure to respond to failure propagation and recover. The ability to respond is modeled for every goal in the G3 criticality set as: 1) buffering (i.e., the ability to retard failure propagation) and 2) recovery. The mechanism for propagating failure and recovery throughout goals is as follows.

1) A goal is challenged to fail if at least one of its ancestor goals has failed.
2) A goal is enabled to recover if all direct ancestor goals have recovered.

The generated scenarios are classified as: 1) recoverable scenarios, to which the infrastructure is able to respond and recover and 2) nonrecoverable scenarios, from which the infrastructure is unable to recover. In addition, the severity of the scenario can be quantified in terms of the depth of propagation, e.g., the number of affected components. For recoverable scenarios, the duration of the recovery process is another metric.

In the given example, the qualitative analysis is conducted on G3, which showed up as the most critical goal in the structural analysis. The scope is further restricted to the (G3, G6, G7) loop. Results are shown in the event sequence diagram illustrated in Fig. 6. The initiating event is depicted at the top left, and the end state on the right. In between, the diagram consists of blocks, each of which contains a list of triggerable events-active failure events (F) and active recovery events (R)—in the GDS. These events are concurrent and may each trigger a different event sequence. Scenarios that terminate as a nonrecoverable state are called deadlocks. In a deadlock none of the systems are able to recover because they are waiting for the input dependency to restore them to an operational state. For instance, this is the case of the failure event sequence from G3 to G6 and G7. The diagram repeats identical patterns from a certain sequence onward, and there may be a long sequence of events with several state changes before the recovery or the deadlock is reached. The outcome of qualitative resilience is the list of possible deadlock scenarios. Qualitative resilience also identifies scenarios from which it takes too long to recover. Quantitative analysis can be used to simulate the system response and check if these deadlock scenarios really occur for the given parameter settings.

D. Quantitative Resilience Analysis

The quantitative analysis of resilience is based on discrete event simulation through the GDS, in which every event is associated with a dynamic behavior in response to disturbance (buffering) or to recovery from failure. These events are allocated a time (or a time distribution). In this case study, we assume buffering and recovery time to be equal to 1 (time unit) for every goal in the GDS. This data set serves as a proof of concept and does not refer to any actual problem setting. In a similar fashion, the duration of the disturbance in G3 is chosen from 1, 1.25, and 1.5 time units. Every goal is given a binary state variable \( x \), with the convention that \( x = 0 \) if the goal has failed, and \( x = 1 \) if it is available.

The results of the simulation are plotted in Fig. 7 as the sum of goal states \( x_{G3} + x_{G4} + x_{G5} + x_{G6} \) versus time, i.e., the loop state. The initial value for the loop state is 4, and it starts to decrease (in discrete steps) as the disturbance is applied to G3. The analysis shows that the infrastructure is resilient to a disturbance of up to 1.25 time units, whereas it fails for 1.5, and enters deadlock. A possible option for avoiding this deadlock scenario is to try to improve the system response by either augmenting buffering or speeding up recovery for the affected goals. In this example, we decided to double the buffering for one of goals G5, G6, and G7. Results of the new simulations are shown in Fig. 8. Doubling the buffering is not effective for G5, whereas it is for G6 and G7, thus converting the deadlock scenario into a recoverable scenario. Doubling the buffering for G6 yields the best system response in terms of transient duration and depth of failure propagation.

The scope of this paper is limited, and assumptions were intentionally simplified. Nonetheless, it examines a real application of I\(^\text{®}ML\) and its analysis toolset for addressing infrastructure resilience. In particular, it provides evidence to help infrastructure operators and decision-makers responsible for resource allocation to avoid local disturbances that may propagate and turn into a large-scale, out-of-control accident.

VI. Conclusion

Modern infrastructures surpass traditional systems in organizational complexity and operational issues. As a consequence, the development of a modeling framework for such systems is a formidable challenge. Several proposals have been made in this respect, as discussed in this paper. They range from large-scale simulation platforms, which exchange data from specific modeling and analysis frameworks, to a more conceptual, holistic representation of the infrastructure. This paper reports a I\(^\text{®}ML\) modeling language and its analysis tools, which conforms to existing research lines, and introduces a number of novel features, which are recalled here.

I\(^\text{®}ML\) is based on a conceptual, holistic representation of the infrastructure. This representation achieves three objectives: 1) it models infrastructure interdependencies; 2) it
The I®ML language supports a resilience-oriented analysis; and 3) it provides results for a thorough preliminary assessment of criticality and vulnerability as support for decision-makers. The modeling focus is on the operational dependencies among the elements of the infrastructure, which are either technical components or socio-technical departments within organizations. In this respect, the I®ML language provides a library of technology-independent building blocks, such as services, systems, agent domains, and agent resources, which are interconnected by producer/consumer, provider/user and controller/controlled operational relationships. The modeling scope and the way in which these elements are connected will reflect the particular focus of the analyst, who may trade off a detailed description of reality against the necessary abstraction required to overcome complexity.

I®ML comes with a set of analysis tools and procedures that investigate structural properties and resilience. An I®ML model has to be transformed into a GDS to apply these tools. The GDS is a directed graph in which goals (nodes) are related to each other by operational dependencies. The structural analysis of the GDS returns the most vulnerable and critical goals. Then resilience analysis addresses the infrastructure response to the presence of off-nominal conditions (i.e., failure, nonachievement, partial achievement) of one or more of its goals. In the resilience analysis of the GDS, a simple dynamic behavior is associated with the components that provide goals. This accounts for the capability of somehow resisting disturbances (by buffering resources) and recovery. The methodology has been applied to a smart grid case study, taken from the U.S. NIST document.
The I®ML language and the analysis tools were designed to meet the requirements that an infrastructure analysis methodology should possess, as described in [8]. Some concepts, like the event sequence diagram for scenario generation, were also inspired by the exploratory procedures carried out by other high-level system analysis techniques like HAZOP [45], [46] and PRA [16]. Nonetheless, the methodology has a number of distinctive features, which we briefly note here. The most important feature is the conceptual view of the infrastructure with its focus on the operational dependencies among components. This representation results in an efficient model, which lends itself to a thorough computational analysis of the infrastructure. The analysis outcomes point out structural and dynamic criticalities and vulnerabilities, which can be translated into recommendations. For example, it is possible to raise awareness about nontrivial dependencies in order to allocate additional resources to improve resilience. On this ground, I®ML primarily targets infrastructure operators or decision-makers, as they prioritize a more abstract and comprehensive picture of a system over a specialized view of its parts.

I®ML proved to be effective for the example considered, especially for representing the following features: functional dependency chains among different systems and domains; clear identification of vulnerable nodes (systems) and their relative importance and interplay of the various parts when challenged by disturbances. I®ML analysis leads to a screening of structural and dynamic properties, which are related to the resilient behavior of a SoS, in order to provide additional insights about possible misbehaviors at a large scale. As a consequence, new resilience-oriented recommendations and measures may be considered for future implementation. This is an achievable goal thanks to the set of metrics introduced. Additionally, a hard issue has been dealt with in the example: the usage of a common framework that breaks through the specificities of particular domains and returns a coherent functional description of the whole, including its (desired and undesired) behavior.

I®ML and its analysis toolset can be applied as a standalone modeling framework or in combination with other system analysis tools, such as a risk assessment framework. This framework can estimate the likelihood and consequences of a given resilience scenario from the inputted outcome of a qualitative resilience analysis (i.e., the event sequence). As future directions of research, we envisage the application of I®ML to accident analysis and the development of a software tool that will implement I®ML modeling and its associated analysis framework.

REFERENCES


**FILIPPINI AND SILVA: I@ML: AN INFRASTRUCTURE RESILIENCE-ORIENTED MODELING LANGUAGE**

**Andrés Silva** received the Ph.D. degree in requirements management, in 2001.

He is currently an Associate Professor with the Facultad de Informática (Computer Science School), Universidad Politécnica de Madrid (UPM), Madrid, Spain. He has two years of industrial experience in software engineering, and did a predoctoral stay at the Institute for Systems, Informatics and Safety, ascribed to the Joint Research Centre (JRC) of the European Commission. His research career has been centered on Requirements Engineering and Software Engineering. He has authored several research papers published at major conferences, such as ICSE and relevant journals such as the IEEE SOFTWARE, the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, and Safety Science, Reliability Engineering and System Safety. He currently plays the role of Requirements Engineering Advisor at the Biomedical Informatics Research Group of the UPM. In the last years he has been involved in the topic of resilience modeling and analysis for critical infrastructures, such as the Smart Grid, in collaboration with the Institute for the Protection and Security of the Citizen (JRC of the European Commission) with relevant publications on the subject.

Dr. Silva served as a member of the Program Committee of several conferences, edited a book on Requirements Engineering for Sociotechnical Systems, and coedited a special issue of *IET Software* devoted to software architecture.

**FILIPPINI AND SILVA: I@ML: AN INFRASTRUCTURE RESILIENCE-ORIENTED MODELING LANGUAGE**

**Andrés Silva** graduated in computer engineering in 2000, and received the Ph.D. degree in automation and robotics from the University of Pisa, Pisa, Italy, in 2006.

He has spent most of his career in International Research Centres. He was with CERN, Geneva, Switzerland, in 2003–2006 and 2013, the Paul Scheerer Institute, Villigen, Switzerland, in 2007–2009, and the Joint Research Centre of the European Commission in 2009–2012. He was also with the Italian National Research Council in 2000–2003 and the Research Centre E. Piaggio of the University of Pisa in 2006–2007. He is currently a Risk Manager of EBG MedAustron, Wiener Neustadt, Austria. He has a record of about 40 publications in peer-reviewed journals and conference proceedings in the domains of resilience, risk, dependability, particle accelerator technology, and robotics. His current research interests include reliability and safety engineering and integrated risk management for resilient critical infrastructures. He is an Italian Ph.D. Engineer.