An MDA-based Approach to Crisis and Emergency Management Modeling

Antonio De Nicola, Alberto Tofani, Giordano Vicoli, Maria Luisa Villani
Computing and Technological Infrastructure Lab.
ENEA: Italian National Agency for New Technologies, Energy and Sustainable Economic Development
Rome, Italy
{antonio.denicola, alberto.tofani, giordano.vicoli, marialuisa.villani}@enea.it

Abstract— **Managing crisis and emergency requires a deep knowledge of the related scenario. Simulation and analysis tools are considered as a promising mean to reach such understanding. Precondition to these types of tools is the availability of a graphical modeling language allowing domain experts to build formally grounded models. To reach this goal, in this paper, we propose the CEML language and the related meta-model to describe structural aspects of crisis and emergency scenarios. The meta-model consists of a set of modeling constructs, a set of domain relationships, and a set of modeling rules. Then we introduce a set of methodological guidelines to reach an executable code, consisting of a system architecture and the mapping rules to transform the CEML modeling constructs into others typical of discrete event simulation. Finally, we propose a preliminary set of collaboration design patterns to model interaction and communication exchange arising among emergency services providers and citizens to solve the crisis. An emergency scenario example demonstrates the applicability of the presented approach.**

Keywords - Conceptual Modeling; Collaborative Networks; Critical Infrastructures; Model Driven Architecture.

I. INTRODUCTION

Recent natural disasters (e.g., earthquakes, floods, fires) and technical faults (e.g., power outages) and their impact on critical infrastructures (CI) and population have caused a growing attention on how to manage crisis and emergency. In this context, CI services (e.g., telecommunications network, water pipelines) may not work or could not guarantee an acceptable level of service. Since dependencies among CI services are often unpredictable, they could generate further unexpected faults in the CI network. Communications channels could be unavailable to teams needing to collaborate to solve the crisis. Furthermore, beneficiaries of CI, not provided with the needed resources, can act in uncontrolled mode, hindering the work of operators who are trying to restore CI services.

To cope with such complexity and mitigate such effects, a promising approach is to simulate these scenarios. Simulation allows creating a portfolio of virtual crisis and emergency management experiences to be used, for instance, for training institutional operators with the responsibility of solving the crisis.

A precondition to build effective simulation tools is the availability of a modeling language and a modeling methodology allowing domain experts to build formally grounded models that can be converted into simulation models. The MDA (Model-Driven-Architecture) [1] approach can help us to this aim as it provides methods and tools that can be used by domain experts, i.e., institutional operators with a deep knowledge of crisis and emergency scenarios but with limited high-level IT skills. The first required feature of such language is the *domain adequacy*, i.e., how the language is suitable to represent the addressed domain [2]. This is achieved by providing experts with modeling constructs and relationships better reflecting their knowledge about the domain. In the CI domain, it is required to allow modeling of collaboration and interaction among CI services, population, institutional operators and stakeholders operating in crisis and emergency scenarios. Then the language has to permit modeling of both structural and behavioral aspects. It has to be formally grounded to allow models to be processed as source code of appropriate simulation programs. It has to be based on widely accepted existing standards to support model interoperability between different simulation tools. Finally, it has to be supported by a graphical notation to allow intuitive and user-friendly modeling.

In this paper we propose CEML (Crisis and Emergency Modeling Language), an abstract level language to model crisis and emergency management scenarios. In particular, we describe the related CEML meta-model, consisting of a set of modeling constructs, a set of relationships, a set of modeling rules, and its formalization using SysML [3] and OCL [4]. Here, we focus mainly on presenting how CEML supports structural modeling of a crisis and emergency scenario. Modeling of behavioral aspects will be treated in another paper.

CEML's objective is to support domain experts in building a model of a CI scenario. However, a CEML model has not been conceived to be directly simulated, since it needs to be transformed into a format closer to the computer programs. For this reason, with respect to [5], here we propose some methodological guidelines to reach an executable code. They consist of a system architecture and a set of mapping rules to transform the CEML modeling constructs to the constructs typically used in the discrete event simulation tools.

Then we propose a modeling methodology tailored to model collaboration needed in crisis and emergency scenarios. This methodology is based on Collaboration

Design Patterns (CDP)s. A design pattern is a reusable solution to a recurrent modeling problem [6]. In particular, collaboration design patterns model interaction and communication exchange arising during the crisis. As example, here we propose five CDPs: clustered service, basic communication, heterogeneous networking, single service provider, and infrastructure.

The rest of the paper is organized as follows. Section 2 presents related work in the area. Section 3 describes the meta-model for crisis and emergency scenarios and its formalization. Section 4 presents some guidelines to implement the CEML language into a simulation platform. Section 5 proposes a preliminary set of collaboration design patterns for crisis scenarios. Section 6 describes an example concerning emergency management after earthquake events and shows an application of the proposed modeling framework. Section 7 contains a discussion on the evaluation of our approach and, finally, Section 8 presents conclusions and future work.

II. Related Work

Nowadays there is an increasing interest on crisis and emergency management modeling and simulation. The aim is to propose effective modeling and simulation approaches to analyze crisis scenarios, and to test crisis and/or disaster management procedures.

The main concepts and definitions related to critical infrastructures (CI) are presented in [7]. An interesting approach to describe various aspects of CI is the ontological approach. In [8], for instance, five meta-models are proposed to characterize various aspects of an infrastructure network, such as managerial, structural and organizational aspects. These meta-models are defined as a UML profile with the aim to completely describe the critical infrastructures domain and their interdependencies. Instead, here we concentrate on the problem of graphically building structural models of crisis management scenarios, also involving humans, for simulation purposes.

Ontologies to describe either emergency plans or disasters affecting critical infrastructures are presented in [9], [10], [11], and [12].

All these works, which we have considered as a starting point for our research, are complementary to our result, as they provide means to semantically enrich simulation models realized with our language.

Several papers propose an MDA approach to simulation. Among them we cite [13], [14], and [15]. In particular, we share with them a layered approach proposing a different type of model representation for the PIM (i.e., Platform Independent Model) level and the PSM (i.e., Platform Specific Model) level [16]. The main difference with these approaches concerns the scope. Whereas they are general purpose, we focus on the crisis and emergency management domain. Consequently, CEML has been conceived to model such domain whereas it is not the best solution for a different one.

In [17] and [18] SysML is proposed as "standard" meta-model for high level discrete event simulation models to be mapped to Arena and DEVS programs. Indeed, this is proposed to ease the access to simulation technology to non ICT experts and to allow exchange of simulation models between tools.

Instead, in [19] UML is proposed as modeling language for agent-based simulators of interdependent critical infrastructures. To this aim, the authors define a methodology for the development of the simulator that suggests the UML diagrams to be used and how these may map to an agent-based model. As the UML meta-model is used as it is and the methodology is given in the form of design suggestions, this work is addressed to software engineers and does not aim at the formalization required by MDA.

With all of these works we share the choice of the UML meta-model (and/or of its profiles) as a root for a modeling language in this domain. Indeed, generally, UML is the most used language for the specification and development of software applications, and, specifically for our work, many tools are available, especially in the MDA world, to implement and validate our approach.

In addition to what is presented by others in the same field, we propose a set of CDPs to support crisis management experts in modeling crisis scenarios. At the best of our knowledge there is no similar proposal in the crisis and emergency management sector.

III. A Meta-Model for Crisis and emergency scenarios

In this section we present the CEML meta-model aimed at guiding a modeler in representing the structural aspects of a crisis and emergency scenario. A meta-model is a design framework describing the basic model elements, the relationships between them, and their semantics. Furthermore it defines the rules for their use [20]. As stated in the introduction, CEML is defined at a high level of detail since it has been conceived mainly for domain experts. In fact, according to the MDA approach, CEML is located at the PIM level.

For this reason, the modeling constructs and relationships have been defined starting from an analysis of crisis and emergency scenarios and from interviews with domain experts. In particular, we have given importance to two requirements. One is simplicity: domain modelers prefer a limited number of constructs and more focused rather than many abstract constructs most of which not needed for their purposes. The other requirement is that models should be service-based: services are the abstractions used by the domain experts for the entities of their scenarios and are at the right level of granularity compared, for example, to individual functions.

A. CEML Modeling Constructs

The CEML modeling constructs define the "terms" used when describing crisis and emergency scenarios. They can be classified as active and passive constructs. The active constructs allow modeling entities able to perform activities (e.g., processing a resource, issuing a message) and their behavior. They are: the abstract service, specialized as service, human service, and communication service; the

behavior; the external event; and the user. The passive constructs allow modeling entities managed or processed by active entities. They are: the message, the resource, and the connectivity. A natural language description of the CEML modeling constructs now follows.

Abstract Service. It represents the active entity processing either a *resource* entity or a *message* entity or a *connectivity* entity. It can be either a *service* or a *human service* or a *communication service*.

Service. It represents the active entity either producing (e.g., power house) or providing (e.g., information service) or transporting (e.g., electrical power grid) a given *resource* entity.

Human service. It represents the active entity providing a given *resource* in the form of human activities (e.g., fire brigades).

Communication service. It represents, from a physical perspective, the active entity allowing communication and information exchange between two of the following entities: *service*, *human service*, and *user* (e.g., between two *services*, between a *service* and a *user*).

Behavior. It represents an operational feature of either a *service* or a *human service* or a *communication service* or a *user* entity. This allows completing the structural model with behavioral specifications.

External event. It represents the active entity (e.g., failure, earthquake) affecting the operational status of either a *service* entity or a *human service* entity or a *communication service* entity or affecting the wellness of a *user* entity.

User. It represents the entity using or consuming a *resource* entity (e.g., hospital). It is characterized by a wellness level.

Message. It represents information content exchanged in a communication.

Resource. It represents the passive entity processed (i.e., produced, provided, transported) by either a *service* entity or a *human service* entity. It can be either material (e.g., water) or immaterial (e.g., fire brigades activity). It can be input to either another *service* entity or a *communication service* entity or a *human service* entity or a *user* entity. It can contribute significantly to *user*'s wellness level.

Connectivity. It represents, from a physical perspective, the output of a *communication service* entity.

B. CEML Relationships

The CEML relationships allow modeling flowing of passive entities, through the flow and the port relationships, and how an external event affects another entity through the impact relationship. Flow relationships are the resource flow, the connectivity flow and the message flow. Port relationships are: the abstract port, specialized as communication port, message port, and resource port; and the connection port group. A natural language description of the CEML relationships now follows.

Resource Flow. It represents resource passing through ports from a *service* or *human service* entity to either a *user* or a *service* or a *human service* or a *communication service* entity.

Connectivity Flow. It represents, from a physical perspective, the communication channel provision (through ports) from a *communication service* entity to either a *service* or a *human service* or a *user* or another *communication service* entity.

Message Flow. It represents, from a logical perspective, the exchange of information content through ports between two of the following entities: *service*, *human service*, and *user* (e.g., between two *services*, between a *service* and a *user*).

Abstract Port. It represents the abstract entity linking either an *abstract service* entity or an *user* entity to either one or more *connectivity flow* entities, or one or more *message flow* entities, or one or more *resource flow* entities. It can be either a *message port* or a *communication port* or a *resource port*.

Communication Port. It represents the abstract entity linking either a *communication service* or *a human service* or *a service* or *a user* entity to one or more *connectivity flow* entities.

Message Port. It represents the abstract entity linking either a *service* or a *human service* or a *user* entity to one or more *message flow* entities.

Resource Port. It represents the abstract entity linking either a *service* or a *human service* or a *communication service* or a *user* entity to one or more *resource flow* entities.

Connection Port Group. It represents the abstract entity grouping one *communication port* entity and one or more *message port* entities and belonging to either a *service* or a *human service* or a *user* entity.

Impact. It represents how an *external event* entity affects one or more of the following entities: *service*, *communication service*, *human service*, and *user*.

C. CEML Modeling Rules

The CEML modeling rules are the syntactic rules to generate well-formed CEML models. The 13 modeling rules now follow.

C1. An element can be categorized only as a modeling construct or as a relationship.

C2. A **service** element has *0..n* incoming **resource port** elements, *1..n* outgoing **resource port** elements, and *0..n* **connection port group** elements.

C3. A **human service** element has *0..n* incoming **resource port** elements, *1..n* outgoing **resource port** elements, and *0..n* **connection port group** elements.

C4. A **communication service** element has *0..n* incoming **resource port** elements and *1..n* outgoing **communication port** elements.

C5. The **service** element, the **human service** element, and the **communication service** element are specializations of the **abstract service** element.

C6. The **message port** element, the **communication port** element, and the **resource port** element are specializations of the **abstract port** element.

C7. Every **abstract service** element is characterized by *0..n* **behavior** elements.

C8. Every **abstract service** element is affected by *0..n* **external event** elements by means of the **impact** element.

C9. A **user** element has *0..n* incoming **resource port** elements and *0..n* **connection port** group elements**.**

C10. A **user** element is affected by *0..n* **external event** elements by means of the **impact** element.

C11. A **message flow** element is linked to *1..n* **message** elements and holds between two **message port** elements belonging to two **connection port group** elements.

C12. A **resource flow** element is linked to *1..n* **resource** elements and holds between 2 **resource port** elements. The **resource flow** element is directed from a **resource port** element belonging either to a **service** or **human service** element and to a **resource port** belonging either to an **abstract service** element or to an **user** element.

C13. A **connectivity** element is directed from a **communication port** element, belonging to a **communication service** element, to a **message port** element, belonging to a **connection port group**.

D. CEML Meta-model formalization

In order to equip the language with a sort of formal grounding, so that smart editors could be defined with validation facilities, we have identified SysML [3], a standard language sponsored by OMG (Object Management Group), as a good candidate. SysML comes as a *profile* of UML 2.0, that is, extends the UML meta-model with constructs to enable "system" other than "software" modeling and provides some new diagram types. Therefore, SysML inherits all the advantages of UML: the multi-views representation of a system model; the simplicity of the notation, which is addressed to stakeholders with different levels of technical knowledge; the XML schema for tools interoperability (XMI); and, finally, the "semi-formal" specification, which has been better clarified starting from version 2.0, that allows model-driven development to take place. Our meta-model is an application of SysML profile tailored to critical infrastructures modeling and, as such, it is a domain-specialization of a subset of SysML. We do this by creating a new profile following the stereotype extension mechanism specified by UML.

Specifically, we consider the components of the *Internal Block Diagram* of SysML, which is based on the *Block* entity. According to the OMG specification, blocks "are modular units of a system description, which define a collection of features to describe a system or other elements of interest. These may include both structural and behavioral features, such as properties and operations, to represent the state of the system and behavior that the system may exhibit".

Figure 1 shows the relationship of the *User* and *AbstractService* constructs of our meta-model with the *Block* entity of SysML. They can have a behavior specified and can be connected with other blocks through ports. However, differently from services, a *User* does not provide functions/resources to other model elements. Note that the *User* construct in our meta-model cannot be mapped to the UML (or SysML) Actor meta-class as we intend the *User* be inside the model (and not part of the environment).

Flow ports are introduced in SysML as a specialization of UML ports "to specify the input and output items that may flow between a block and its environment". *Flow ports* are generally typed with respect to the item that can flow (in, out, or inout). In our meta-model we have decided to introduce three port types as shown in Figure 2.

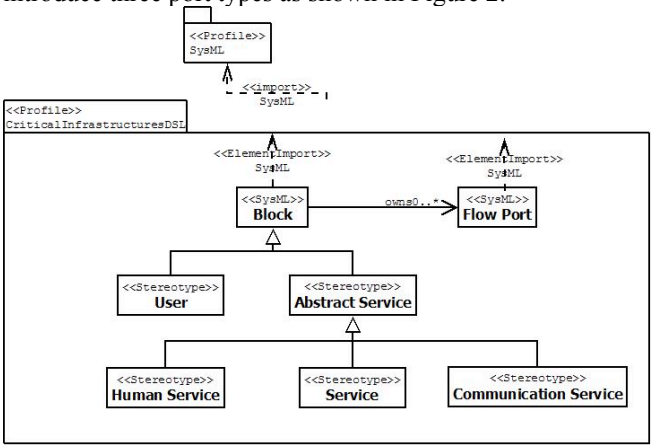

Figure 1: Relationship of the *Abstract Service* and *User* constructs with the *Block* entity of SysML

In order to relate the message flow generating from a service/user with the transport mean that allows it (e.g., internet connection), we have identified a particular type of (non-atomic) *Flow Port*, namely the *Connection Port Group*, with the aim of grouping together one or more message ports with one (in) communication port.

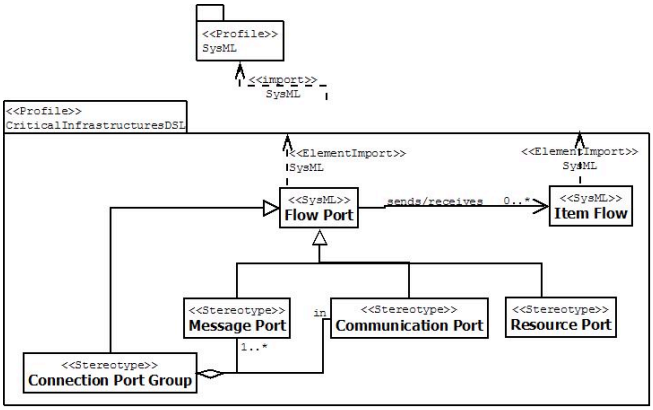

Figure 2: Relationship of the *Message Port*, *Communication Port*, *Resource Port*, and *Connection Port Group* with the *Flow Port* of SysML

The specialization of *Flow Ports* in three types obviously requires that also *Item Flow* be specialized accordingly. The type of the item that can flow through an atomic port (e.g., water, power) in SysML is specified by the *FlowProperty* stereotype, which can be simply a label. In our case, we want to distinguish between: *message*, *connectivity*, and *resource*, which we define as a specialization of *FlowProperty*. Instead, non-atomic *Flow Ports* in SysML are defined through a FlowSpecification object, which is a collection of FlowProperty objects, each referring to a single item. In SysML, items flow through *Connectors*, used to link blocks. For graphical convenience only, we have defined a SysML

connector specialization for *message flow* to represent it as a dashed arrow line (see Table II below).

As we want to design analysis scenarios for crisis management, we need to represent the events that may happen and what services/users they may affect. Here we want to represent just the type of the *external event*, such as earthquake, flood, and so on, and its "affecting" relationship to one or more scenario entities. Therefore, we intend the event being an abstract element outside the model (part of the environment) but influencing it, and so this definition specializes that of the *Actor* in UML.

Finally, each kind of service or user element, being a UML Class, might be modeled internally through a *Behavior* object, which is the link to one or more behavioral descriptions of the scenario that we will treat as future work.

The following tables include the list of all the constructs (Table I) and relationships (Table II) of the proposed CEML meta-model, with the corresponding formal notation describing the extension from the SysML profile and UML references, and the graphical symbol we have associated to them to be used in our diagrams.

TABLE I. CEML MODELING CONSTRUCTS, BASE SYSML META-CLASS, AND CORRESPONDING GRAPHICAL NOTATION

CEML Modeling Constructs	*Base SysML Metaclass*	*Graphical Notation*
Abstract Service	SysML::Blocks::Block	NA
Service	SysML::Blocks::Block	Service Name
Human Service	SysML::Blocks::Block	Human Service Name
Communication Service	SysML::Blocks::Block	Com. Service Name
Behavior	UML::CommonBehaviors:: BasicBehaviors:: Behavior	NA
External Event	SysML::Actor	External Event name
User	SysML::Blocks::Block	User name
Message	SysML::Property::Flow Property	Message name
Resource	SysML::Property:: FlowProperty	Resource name
Connectivity	SysML::Property:: FlowProperty	NA

TABLE II. CEML RELATIONSHIPS, BASE SYSML META-CLASS, AND CORRESPONDING GRAPHICAL NOTATION

CEML Relationships	*Base SysML Metaclass*	*Graphical Notation*
Resource Flow	SysML::Ports&Flows:: ItemFlow	Resource name →
Connectivity Flow	SysML::Ports&Flows:: ItemFlow	Connectivity name →
Message Flow	SysML::Ports&Flows:: ItemFlow	Message name ⇢
Abstract Port	SysML::Ports&Flows:: FlowPort	NA
Connection Port Group	SysML::Blocks::Block	Connection Port Group Name
Message Port	SysML::Ports&Flows:: FlowPort	Message Port Name
Communication Port	SysML::Ports&Flows:: FlowPort	Communication Port Name
Resource Port	SysML::Ports&Flows:: FlowPort	Resource Port Name
Impact	UML4SysML::Association	Affecting →

In a UML profile, "well-formedness" rules, such as the constraints listed in sub-section C, can be encoded in OCL, which is a declarative formal language to express properties of UML models. An OCL rule is defined within a context, that is, the element to which some Boolean expression, specified by the rule, should apply. We give here some representative examples of OCL implementation of the constraints of our meta-model. Specifically, through rule C4, we show how to link one or more subtypes of *Port* to the corresponding subtype of the *AbstractService* construct. Instead, through the first part of rule C12, we show how to link an Item specialization to the corresponding *ItemFlow* and *Port* subtypes. Finally, both rules C8 and C10 are based on a invariant on the use of connector subtypes. Namely, in this example, the *Impact* relationship is *always* originated by an *ExternalEvent* construct towards a *User* or an *AbstractService* construct.

C4. A **communication service** element has *0..n* incoming **resource port** elements and *1..n* outgoing **communication port** elements.

```
Context SysML::Blocks::Block
self. oclIsTypeOf(CommunicationService) implies
(self.attributes->select(oclIsTypeOf(MessagePort))->size()=0) and
(self.attributes>
select(oclIsTypeOf(CommunicationPort).direction='out')->size()>0)
and (self.attributes->
select(oclIsTypeOf(CommunicationPort).direction='in')->size()=0)
and (self.attributes->
select(oclIsTypeOf(ResourcePort).direction='out')->size()=0)
and (self.attributes->
select(oclIsTypeOf(ResourcePort).direction='inout')->size()=0)
and (self.attributes->
select(oclIsTypeOf(CommunicationPort).direction='inout')->size()=0)
```

C8. Every **abstract service** element is affected by *0..n* **external event** elements by means of the **impact** element.

C10. A **user** element is affected by *0..n* **external event** elements by means of the **impact** element

```
Context UML4SysML::Association
self.oclIsTypeOf(Impact) implies
(self.memberEnd->size()=2 and self. memberEnd -> exists(p |
p.oclIsTypeOf(ExternalEvent)) and
(self.navigableOwnedEnd->size()>0 and
self.navigableOwnedEnd -> forAll(p |
p.oclIsTypeOf(AbstractService) or p.oclIsTypeOf(User)))
```

(First part of) **C12**. A resource flow element is linked to *1..n* resource elements and holds between 2 resource port elements.

```
Context SysML::Ports&Flows:: FlowPort
self.oclIsTypeOf(ResourcePort) implies
(self.type.oclIsTypeOf(FlowSpecification) and self.type.attributes-
>size()>0 and self.type.attributes ->forAll(r |
r.type.oclIsTypeOf(Resource)))
```

IV. GUIDELINES FOR CEML MODELS SIMULATION

CEML is a language to support experts of the various critical infrastructures in creating global representations of these systems, and their interactions, in order for them to analyze problems and take strategic decisions. As simulation plays a key role in this activity, it is desirable that the constructed CEML models are then used to generate input code for simulators. This means that a CEML model needs to be implemented in a simulation language of some kind, and integrated with the simulation-specific constructs of the language required to run the experiments.

This objective is achieved by following a model-based design methodology. Indeed, referring to the model driven architecture paradigm [16], CEML is located at the Platform Independent Model (PIM) layer, being a domain specific language formally defined as a profile of SysML. In order to be actually used in the context of simulation, mapping rules of the CEML meta-model to some lower level simulation language need to be provided to convert CEML models into Platform Specific Models (PSM), executable by specific simulators.

In this section we present a software system architecture allowing building and verifying a CEML model and, finally, transforming it into a platform-specific code. Then, we present how the transformation is performed by means of a set of mapping rules from the CEML modeling constructs to a set of generic modeling constructs typical of discrete event simulation. This approach has been applied on a real case study within the EU project MOTIA [21] [22] [23].

A. The Architecture for CEML Modeling and Simulation

The CEML approach is enabled by the architecture depicted in Figure 3, that shows how CEML can be used together with existing simulation environments.

The architecture highlights a complete decoupling between the modeling and simulation functions. Indeed, although every simulator provides its own design interface, often these interfaces require some programming skills and are not graphical. Moreover, in a simulator program the model of the real world to test is mixed with simulation programming functions, thus making models reuse and evolution a more complex task. In the proposed architecture, instead, the modeling and simulation design activities may have a different focus and so can even be performed by different users.

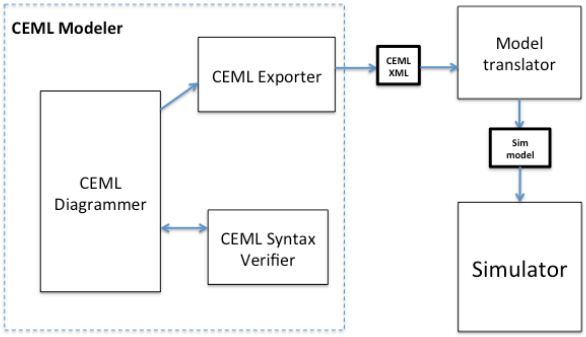

Figure 3: Software system architecture to support CEML models definition and simulation

The ***CEML Modeler*** component allows for editing correct CEML models which may be exported to an XML format. Specifically, the modeling environment consists of: the ***CEML Diagrammer***, that implements the CEML's meta-model and provides a graphical interface for editing the models; the ***CEML Syntax Verifier***, that, during the editing, allows to verify that the model is well-formed with respect to the CEML modeling rules; and a ***CEML Model Exporter*** for an XML serialization of the CEML model. As CEML is formally defined as a SysML profile, with the modeling rules expressed by OCL constraints, the *CEML Modeler* component can be implemented by any UML tool that supports SysML and profiles, such as Topcased [24] or UModel [25]. All of these tools provide an XML export function of the models. However, an XML schema has been created specifically for CEML to simplify the XMI serialization of CEML models leaving just the relevant data for the simulation stored in the XML document.

The ***Model Translator*** component implements the mapping rules that allow transforming a CEML model into code for a specific simulator. The output of the *Model Translator* essentially contains the data and specifications needed to instantiate the structural model or code of a simulation scenario. Clearly, the user of the simulator must configure the simulation scenario and add the required simulator specific code to obtain an executable program.

B. The CISP Simulation Platform

Although CEML has not been initially conceived for any particular technology, an existing discrete event simulator for critical infrastructures developed at ENEA, called CISP [22] [26], has been used to experiment CEML's integration with a simulation environment. Other than critical infrastructures, CISP provides a set of predefined components that can be easily extended in order to simulate other domain specific

scenarios. Indeed, these components implement the building blocks of a general discrete simulation language, such as:

- **entity**: a simulation element that flows into the system (e.g., materials or workpieces, documents or data packets in a computer systems);
- **source**: a simulation element that places entities in the system;
- **sink**: a simulation element that receives entities;
- **queue**: a simulation element that collects entities;
- **decider**: a simulation element that takes decisions, for instance about entities flowing;
- **event**: a simulation element, modeling the change of state in the simulation environment. It is instantaneous and does not require time;
- **activity**: a simulation element, modeling a set of operations that transform the state of an entity and require time to be executed.

CISP is based on Discrete Event Simulation (DES) paradigm [27]. DES is an operational research technique where the simulation is advanced from event time to event time rather than using a continuously advancing time clock as in continuous simulation.

C. *PIM-PSM Mapping Rules: from the CEML Modeling Constructs to the CISP Components*

In the following we show how the transition from the CEML modeling constructs (at PIM level) to the CISP components (at PSM level) can be performed. In particular, we specify a set of mapping rules, covering a subset of the CEML modeling constructs that allow transforming a CEML model into a specification of the structure of a CISP simulation scenario. This scenario has to be completed with behavioral aspects by CISP users.

***AbstractService* mapping rule**

The *AbstractService* CEML modeling construct is implemented at the PSM level by using the *Activity*, the *Queue*, the *Sink*, and *0..n Decider* components. The same applies to the *Service*, the *HumanService*, and the *CommunicationService* CEML modeling constructs. At the PSM level, the *Activity* component allows modeling of operations related to the *AbstractService*; the *Queue* and the *Sink* components allow to store, respectively, input and output resources either to be processed or processed by the *AbstractService*. Finally, the *Decider* component allows modeling decisions taken by the *AbstractService* about its operations and depending on the internal status and external events affecting its behavior. Figure 4 presents the *AbstractService* mapping rule.

***User* mapping rule**

As the *AbstractService* mapping rule, the *User* CEML modeling construct is implemented at the PSM level by using the *Activity*, the *Queue*, the *Sink*, and the *Decider* component. At the PSM level, the *Activity* component allows modeling of user operations to interact with the external world (e.g., exchange of messages, receipt of resources). The *Queue* and the *Sink* components allow modeling, respectively, of issuing and receiving messages and resources. Please note that, with respect to the *AbstractService* mapping rule, here the *Decider* component allows modeling decisions taken by the *User* about message sending and depending on the wellness level and external events affecting it. Figure 5 presents the *User* mapping rule.

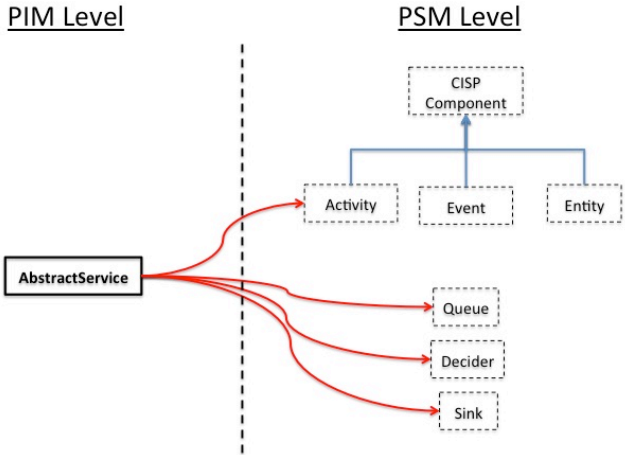

Figure 4: *AbstractService* Mapping to the PSM Level

***ExternalEvent* mapping rule**

The *ExternalEvent* CEML modeling construct is implemented at the PSM level by using the *Event* CISP component. Figure 6 presents the *ExternalEvent* mapping rule.

***Resource* mapping rule**

The *Resource* CEML modeling construct is implemented at the PSM level by using the *Entity* CISP component. The same applies to both the *Message* and the *Connectivity* CEML modeling construct. Figure 7 presents the *Resource*/*Message*/*Connectivity* mapping rule.

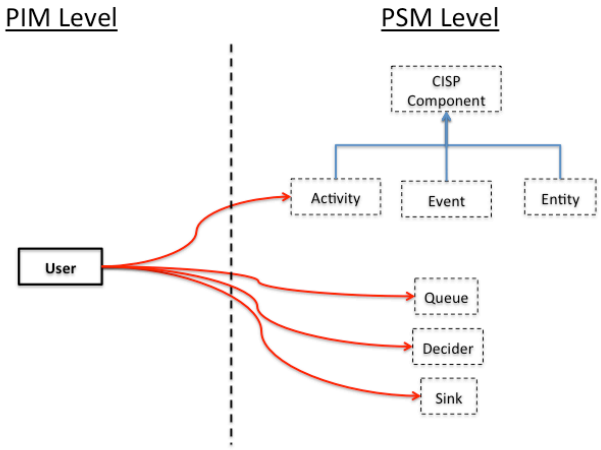

Figure 5: *User* Mapping to the PSM Level

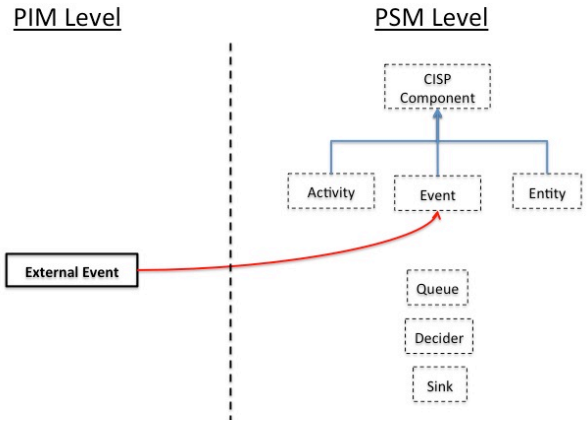

Figure 6: *ExternalEvent* Mapping to the PSM Level

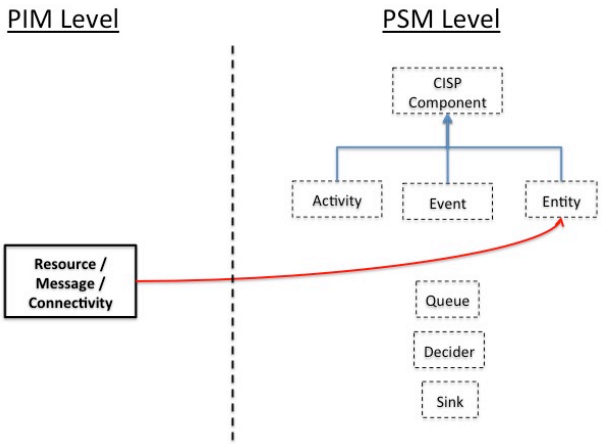

Figure 7: Mapping of either *Resource* or *Message* or *Connectivity* to the PSM Level

V. CRITICAL INFRASTRUCTURES COLLABORATION DESIGN PATTERNS

Design patterns are proving to be one of the most promising methodological tools to support building of models and, more in general, ICT artifacts like software programs. Currently, there are several proposals of design patterns in different fields, e.g., UML design patterns for software engineering [28], workflow patterns for business process management [29], and ontology design patterns for ontology building [30]. Here we propose some examples of domain-specific design patterns, devoted to facilitate modeling of interaction and communication exchange arising among emergency services providers and citizens to solve the crisis. In particular, a CDP allows representing a chunk of the reality where collaboration is performed. By using this approach, modelers can create a repository of CDPs to be reused to describe similar scenarios. Furthermore, these patterns may result useful when analyzing the modeled system. Especially in the case of analysis by simulation of big size models, structured diagrams may help users in the identification of criticalities and in planning ways to probe them. In the following, five CDP examples we have identified are presented: clustered service, basic communication, heterogeneous networking, single service provider, and infrastructure.

CDP1. Clustered Service

Figure 8 shows the clustered service CDP devoted to model collaboration arising among different services working together to either provide or produce or transport a resource (e.g., energy, water). In particular, the objective of this CDP is to model exchange of resources and information. Furthermore, this CDP models the physical connection provided by a communication service and allowing information exchange.

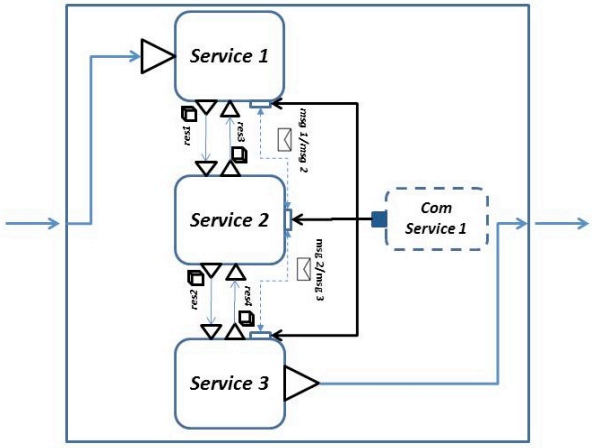

Figure 8 Clustered Service CDP

CDP2. Basic Communication

Figure 9 shows different cases concerning the basic communication CDP representing a simple exchange of information where the physical connection is not deemed relevant for the modeling purposes (e.g., oral communication).

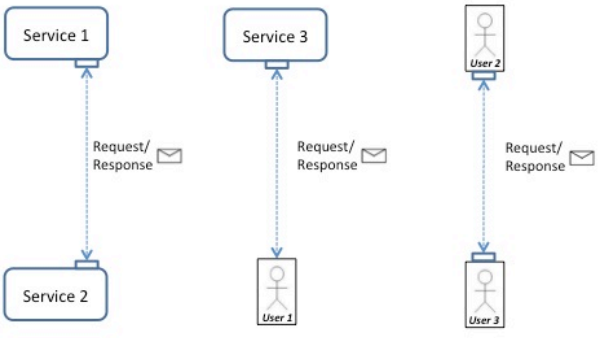

Figure 9 Different cases concerning the Basic Communication CDP

CDP3. Heterogeneous Networking

Figure 10 presents the heterogeneous networking CDP modeling a network of different communication services, guaranteeing the physical connection between two services.

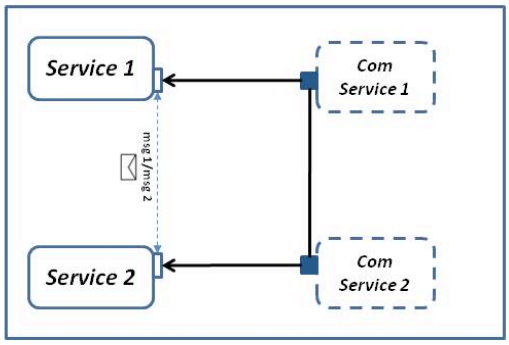

Figure 10 Heterogeneous Networking CDP

CDP4. Single Service Provider

Figure 11 shows the single service provider CDP representing a potentially risky situation where a user (or a service) receives a resource just from a service and this service has vulnerability in case of an external event. This type of CDP is inherently different from the above mentioned CDPs. Whereas CDP1, CDP2, and CDP3 have been mainly conceived to support the domain expert in the modeling phase, CDP4 can be used also in the analysis phase. In fact, CDP4 can be used as a basis for a "situation awareness service", by detecting its presence in a previously built model of scenario.

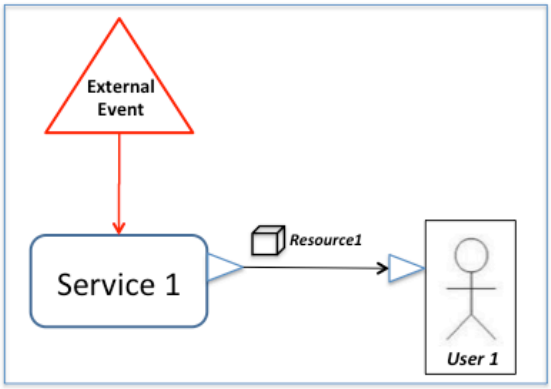

Figure 11 Single Service Provider CDP

CDP5. Infrastructure

Figure 12 shows an example of CDP that can be useful in the analysis phase of the modeled system: the infrastructure CDP. This represents a means to highlight in the model that some services belong to the same infrastructure. The aim is to enable the design and analysis of the infrastructure as a whole, through its relationship with other infrastructures of the model. With respect to the clustered service CDP, the component services of the infrastructure CDP are not necessarily physically connected.

VI. EMERGENCY SCENARIO EXAMPLE

The objective of this section is to demonstrate the usability and flexibility of the proposed modeling framework by describing a real emergency scenario occurred after an earthquake [31]. In particular, here we focus on the main services, resources and users related to the Italian Civil Protection (ICP) emergency management protocol. Figure 13, Figure 14 and Figure 15 present different excerpts from the addressed scenario model. Please note that when some elements of a model appear in more than one figure we omit to repeat some parts of the model itself due to presentation purposes. For the sake of clarity, we also omit some details, as our aim is to demonstrate the usability and flexibility of the proposed modeling framework. A detailed description of the scenario is available in [31]. After an earthquake event, the ICP is able to have a global picture of the impact of this event by using sensor networks, simulation tools, and specific expert team reports. The *Mixed Operative Center* (*COM* in Figure 13) is established near the areas mostly damaged by the earthquake. In this example, the *COM* plays the role of final user. Then there are the *Emergency Services*, the *Emergency Call Service*, and the *Lifeline networks*. The *Emergency Services* represent all actors involved in the emergency management protocol. We describe the details about this service using the clustered service CDP (Figure 14). The *Emergency Call Service* represents the network of emergency call centers devoted to receive feedbacks from user in order to assess how well ICP is facing the emergency. The *Lifeline Networks* element models the infrastructure networks (e.g., electrical distribution and telecommunications network, gas and water pipelines, water treatment systems) of the damaged area. Evaluation of the lifeline performances is one of the most important tasks during an emergency to allow rescue teams to properly and safely operate during an emergency. The networks and their dependencies can be further specified using an appropriate clustered service CDP.

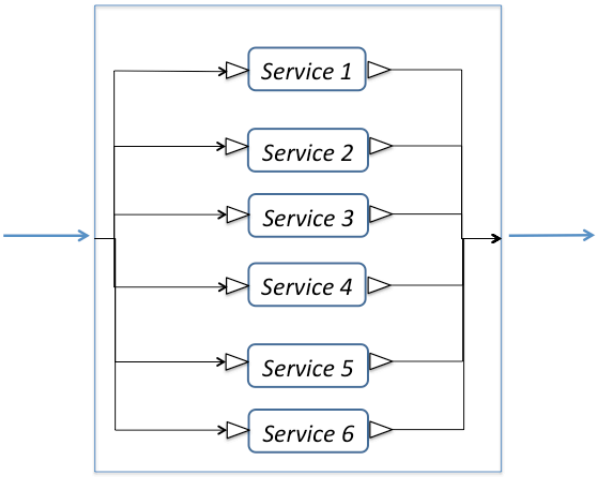

Figure 12 Infrastructure CDP

The *Telco Network* communication service models the connectivity services and resources operating in the area.

By using the clustered service CDP, it is possible to refine the definition of the *Emergency Services* to model the

coordination messages that are exchanged among the major actors during emergency management (Figure 14). The decisional board is represented by the *National Civil Protection Service* (*SNPC*). The coordination messages aim to gather information about available resources at a national, regional, provincial, and local level. The *Direction and Command on site* (*DiComaC*) service is in charge of resources distribution and operations management. All decisions rely on the information about the lifeline performance provided by the *Lifeline Owners* service. Figure 14 shows also the output resources of the *Emergency Services* to the *COM*.

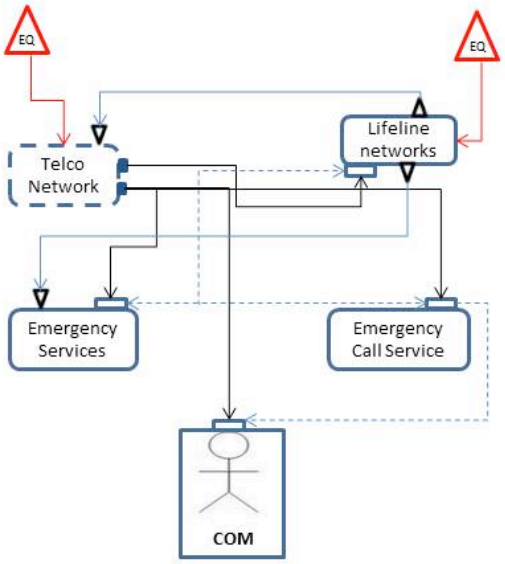

Figure 13 Emergency scenario example

Figure 15 shows how heterogeneous networking CDP can be used to represent different physical connections among services. The *SNPC* uses the public telecommunication network and the internet to exchange messages with the *DiComaC* (Figure 15 a.). On the other hand, for the communication between the *DiComaC* and the ICP rescue teams service it is possible to have several ICT emergency communication channels: telecommunication network, ad hoc network, radio network, and the internet (Figure 15 b.).

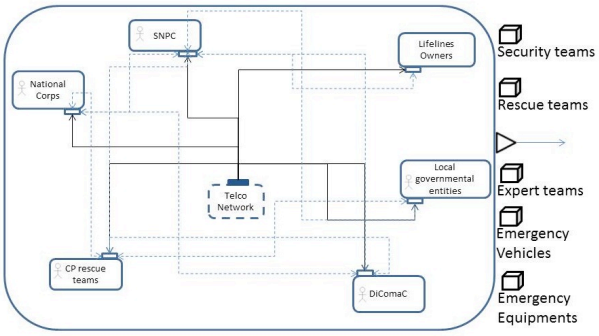

Figure 14 Clustered Service CDP example

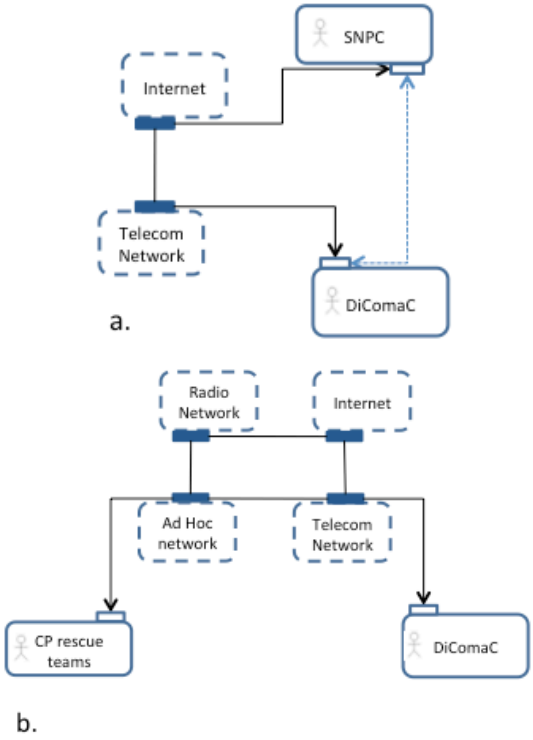

Figure 15 Heterogeneous networking CDP examples

VII. EVALUATION

A further step towards a proper validation of the language has been made by using CEML to model the Italian System for Public Connectivity (SPC), within the activities of the MOTIA project. SPC is a system of federated technological ICT infrastructures of Public Administrations (PAs) to provide eGovernment services to citizens via a shared interface. The various PAs may communicate through a Qualified Internet Service Provider (Q-ISP).

The interest of the MOTIA project for this case study has been to use a CEML model as a means to a simulation-based quantitative analysis of dependencies of the PAs logical network from the underlying telecommunication network. In particular, the functioning of the modeled system has been simulated under normal and perturbed conditions caused by external events. For deeper insights into this system and the experiments we have conducted we remind the reader to [22]. Instead, here we report in greater details some lessons learnt about usability of our approach.

From the modeling side, we have had the opportunity to apply CEML to another real case of crisis management and so to verify the effectiveness of the constructs of the language and of the CDPs. The modeling approach has been evaluated under two perspectives: that of the *domain expert* who knows the system and can judge the adequacy of CEML in representing the system through one or more models; and that of the *system analyst* (a network analyst for this case study) who needs to configure the simulator and hence can judge how the CEML models are useful for his/her understanding of the system.

The structure of the modeled system is simpler than that for the emergency scenario presented in Section VI. Indeed, we have defined only two types of services, namely the *PA service*, as human service, and the *Q-ISP service*, a type of user representing the *Citizen*, and an external event to inject faults in the Q-ISP service. Instead, the complexity of the system is in the size of the PAs logical network and hence on the communication flow between this network and the underlying telecommunications network. This complexity needs to be handled by the network analyst when configuring the simulator with appropriate dependencies metrics, based on the analysis of the structural model.

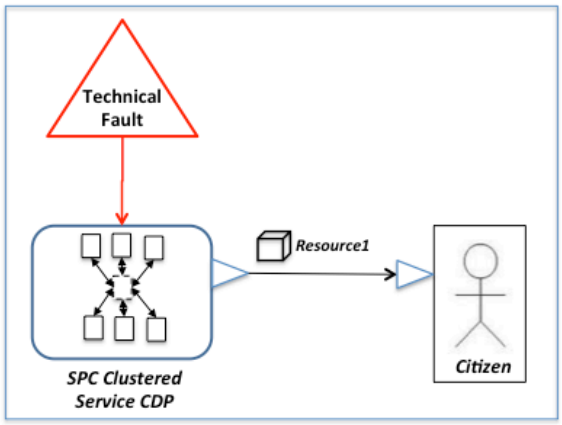

Figure 16 Single service provider CDP example in the SPC case study

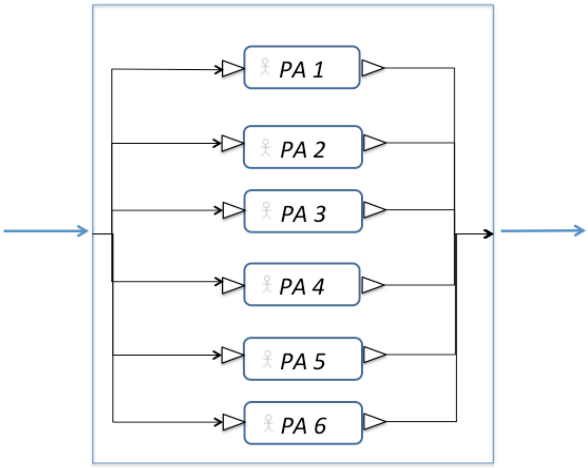

Figure 17 Simplified infrastructure CDP example in the SPC case study

For this activity, which has been carried out with the support of network analysis experts, the use of three CDPs in the model, namely the single service provider CDP together with a clustered service CDP, as shown in Figure 16, and the infrastructure CDP shown in Figure 17, has been convenient. The first two patterns have been used to highlight to the network analyst that the system to study consists of two collaborating ICT infrastructures that produce resources (i.e., PA documents like certifications) to people and that no backup system exists to provide the same resource in case of failure. The other pattern has been used to highlight that one of the two infrastructures (the PAs network) is composed of various services that do not communicate directly and that this infrastructure needs be evaluated as a whole through simulation.

From this experience we have obtained the following important feedbacks:

- in some fields, like critical infrastructures protection, a domain specific modeling language and implementing tools can be used to support the activity of simulation experts (system analysts);
- the domain experts have appreciated the fact that CEML is more concise than a general purpose modeling language (e.g., SysML);
- a domain specific language is a first step to help system analysts in building simulation models but a smarter editor could be implemented to elicit more knowledge from the domain experts and make it explicit;
- CEML and the CDPs can be effective means to develop methodologies and/or processes for the analysis of dependencies in complex systems, like that for the quantitative analysis of interdependencies of ICT systems defined in the MOTIA project [23].

As a future work we intend to formally evaluate these results.

VIII. CONCLUSIONS

In this paper we presented an approach to build models concerning crisis and emergency scenarios. Our approach is based, first of all, on the CEML language and the related meta-model consisting of a set of modeling constructs, a set of relationships, and a set of modeling rules. Then, it proposes some methodological guidelines, consisting of a system architecture and a set of PIM-PSM mapping rules, to allow CEML models to be part of the input data required by simulation environments. Finally, it proposes a modeling methodology based on collaborative design patterns, i.e., reusable solutions to recurrent modeling problems, tailored to model interaction and communication exchange arising during the crisis.

Currently, CEML supports modeling structural aspects of a scenario. We are working on extending the language and the related meta-model to behavioral aspects. For example, in [32], we present a method based on ECA (Event Condition Action) rules [33]. Finally, we are implementing the proposed architecture to permit CEML models to be simulated by other existing simulation tools.

ACKNOWLEDGEMENTS

This work is partially supported by the European Project MOTIA (JLS/2009/CIPS/AG/C1-016). This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

REFERENCES

[1] OMG-MDA, "MDA Guide, version 1.0.1," Available at: http://www.omg.org/mda/presentations.htm, 2003. Retrieved on 4th April, 2011.

[2] F. D'Antonio, M. Missikoff, and F. Taglino, "Formalizing the OPAL eBusiness ontology design patterns with OWL," Proc. of the IESA 2007 Conf., pp. 345-356, 2007.

[3] OMG-SysML, "OMG Systems Modeling Language" version 1.2. Available at: http://www.omgsysml.org/. 2010.

[4] J. Warmer and A. Kleppe, "The Object Constraint Language: Getting Your Models Ready for MDA" (2 ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[5] A. De Nicola, A. Tofani, G. Vicoli, M. L. Villani, Modeling Collaboration for Crisis and Emergency Management. The First International Conference on Advanced Collaborative Networks, Systems and Applications - COLLA 2011 Luxembourg June 19-24, 2011 isbn:978-1-61208-008-6, 2011.

[6] http://en.wikipedia.org/wiki/Design_pattern_(computer_science). Retrieved on 26th February 2011.

[7] S.M. Rinaldi, J.P. Peerenboom, and T.K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," Control Systems Magazine, IEEE , vol.21, no.6, pp.11-25, Dec 2001.

[8] G. A. Bagheri Ebrahim, "UML-CI: A reference model for profiling critical infrastructure systems," Inf. Syst. Frontiers, 12(2), 2010.

[9] W. Wang, X. Zhang, C. Dong, S. Gao, L. Du, and X. Lai, "Emergency Response Organization Ontology Model and its Application," Proc. of ISISE 2009 Symp., pp. 50-54, 2009.

[10] C.X. Dong, W.J. Wang, and P. Yang, "DL-Based the Knowledge Description of Emergency Plan Systems and Its Application," Proc. of MUE'09, pp. 364-368, 2009.

[11] P. C. Kruchten, "A human-centered conceptual model of disasters affecting critical infrastructures," Proc. of ISCRAM 2007 Conf., pp.. 327-344, 2007.

[12] M. A. Sicilia and L. Santos, "Main Elements of a Basic Ontology of Infrastructure Interdependency for the Assessment of Incidents," LNCS, Springer-Verlag, Vol.5736, pp. 533-542, 2009.

[13] S. Parr, A Visual Tool to Simplify the Building of Distributed Simulations Using HLA. Information Security 12, 151-163, 2003.

[14] A. Tolk, Avoiding Another Green Elephant – A Proposal for the Next Generation HLA based on the Model Driven Architecture", 02F-SIW-004, Fall Simulation Interoperability Workshop, Orlando, Florida, September 2002.

[15] H. Zhang, H. Wang, D. Chen, and G. Zacharewicz, A model-driven approach to multidisciplinary collaborative simulation for virtual product development, Advanced Engineering Informatics, Volume 24, Issue 2, Pages 167-179, ISSN 1474-0346, 10.1016/j.aei.2009.07.005, April 2010.

[16] A. G. Kleppe, J. Warmer, and W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[17] L. McGinnis and V. Ustun, "A simple example of SysML-driven simulation," in Proc. of WSC'09, pp. 1703-1710, 2009.

[18] M. Nikolaidou, V. Dalakas, L. Mitsi, G.-D. Kapos, and D. Anagnostopoulos, "A SysML Profile for Classical DEVS Simulators", in Proc. of the 3rd Int. Conf. on Software Engineering Advances, IEEE, pp. 445-450, 2008.

[19] V. Cardellini, E. Casalicchio, and E. Galli. Agent-based Modeling of Interdependencies in Critical Infrastructures through UML, In Proc. of the 2007 spring simulation multiconference (SpringSim '07) – Vol 2, pp. 119—126, 2007.

[20] M. Rosemann and P. Green, "Developing a meta-model for the Bunge-Wand-Weber ontological constructs," Inf. Syst. 27(2): 75-91, 2002.

[21] G. D'Agostino, G. Cicognani, A. De Nicola, and M. L. Villani, Case Study. Deliverable of the Activity 4 of the European Project MOTIA (JLS/2009/CIPS/AG/C1-016), 2012.

[22] G. D'Agostino, A. De Nicola, A. Di Pietro, G. Vicoli, M.L. Villani, and V. Rosato, A Domain Specific Language for the Description and the Simulation of Systems of Interacting Systems, in Advances in Complex Systems, Vol. 15, Suppl. No. 1, 2012.

[23] The MOTIA European Project (JLS/2009/CIPS/AG/C1-016). http://www.motia.eu. Last accessed on 20th June 2012.

[24] TOPCASED The Open-Source Toolkit for Critical Systems, http://www.topcased.org.

[25] UModel Enterprise Edition, Altova, http://www.altova.com/umodel/sysml.html.

[26] G. Vicoli, Discrete Event Simulation. Oral presentation at Workshop UTMEA, Soluzioni per l'energia e l'ambiente, Rome, 2010-09-07.

[27] S. Robinson, Simulation: The Practice of Model Development and Use, Wiley, Chichester, UK, 2004.

[28] A. Spiteri Staines, Modeling UML software design patterns using fundamental modeling concepts (FMC), In Proc. of ECC'08 Conf., pp. 192-197, 2008.

[29] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow Patterns," Distributed and Parallel Databases, 14(3), pp. 5-51, 2003.

[30] A. Gangemi and V. Presutti, "Ontology design patterns," In: Handbook on Ontologies, 2nd edn. International Handbooks on Information Systems. Springer, Heidelberg, 2009.

[31] M. Dolce, S. Giovinazzi, I. Iervolino, E. Nigro, and A. Tang, "Emergency management for lifelines and rapid response after L'Aquila earthquake," Progettazione sismica, n. 3, 2009.

[32] A. De Nicola, G. Vicoli, and M. L. Villani, A Rule-based Approach for Modeling Behaviour in Crisis and Emergency Scenarios, Enterprise Interoperability V, Ed. R. Poler, G. Doumeingts, B. Katzy, and R. Chalmeta, Springer, 2012.

[33] K. R. Dittrich, S. Gatziu, and A. Geppert. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. Lecture Notes in Computer Science 985, Springer, pp. 3-20, 1995.