A Rule-based Approach for Modelling Behaviour in Crisis and Emergency Scenarios

Antonio De Nicola, Giordano Vicoli, and Maria Luisa Villani

Abstract Crisis management is a critical task requiring a deep knowledge of the related scenario where usually interoperability and collaboration among different services and actors take place. To support crisis management a promising approach is based on simulation and analysis. Precondition to them is the possibility to model both structural and behavioural aspects of the addressed domain. In a previous paper we presented the CEML domain specific language to model structural aspects of a crisis scenario. Here we focus on behavioural aspects and we present the CEML behavioural modeling framework to model them. This framework consists of the CEML language and an incremental method to specify behaviour based on the stimulus-response model, the event-condition-action rules, and the Reaction RuleML language.

Keywords Crisis management • Behavioural modeling • Reaction rules

1 Introduction

Management of crisis due to natural and technical disasters (e.g., earthquakes, floods, fires, power outages) affecting population and critical infrastructures (CI) s requires collaboration and interoperability of institutional operators and public and private service providers (e.g., power plants, fire brigades). In such contexts, different actors are requested to flexibly and continuosly react both at internal level, e.g., by re-organizing production of goods or provision of services, and at mutual level, e.g., by re-organizing interoperation and collaboration.

Besides reducing economic losses, crisis management has as main objectives to recover lifelines networks (e.g., electrical distribution and telecommunication

A. De Nicola (✉) • G. Vicoli • M.L. Villani
ENEA, Roma, Italy
e-mail: antonio.denicola@enea.it

R. Poler et al. (eds.), *Enterprise Interoperability V: Shaping Enterprise Interoperability in the Future Internet*, Proceedings of the I-ESA Conferences 5,
DOI 10.1007/978-1-4471-2819-9_9,

network, gas and water pipelines, water treatment systems) and to rescue people. For these reasons, the role of institutional and private operators devoted to solve emergency is gaining a growing importance in the society.

However, to better organize the recovery intervention, operators have to consider several factors, like time, possible side effects, available resources, and existing communication channels. To cope with such complexity and manage crisis, they need to be trained to increase their knowledge about such scenarios and about possible interventions.

A promising approach to reach such understanding and skills is to use simulation tools to simulate several crisis scenarios and different recovery strategies.

A precondition to build effective simulators is the availability of a modeling language and a modeling methodology allowing domain experts to represent behaviour and to build formally grounded models to be converted into simulation models.

According to the simulation purpose, behaviour may be expressed in several forms. For instance, one may be interested in using simulation to decide the best evacuation plan of some urban area. Thus, the general behaviour of people has also to be taken into account to avoid bottlenecks and an explicit representation of how people move may help in this analysis. Otherwise, one can be interested in evaluating the impact of some decision on the overall *Quality of Service* (QoS) of the represented system (e.g., electrical distribution network). This can be computed considering the QoS variation of each service (e.g., a power plant) and how these services interact among themselves. Thus, the behaviour of each element is specified in terms of mathematical functions that just focus on QoS (see i2sim [1]).

The CEML (Crisis and Emergency Modeling Language) language [2], developed along the lines of the MDA (Model-Driven-Architecture) [3] approach, provides methods and tools to model structural aspects concerning collaboration and interoperability among CI services, population, institutional operators, and stakeholders operating in crisis and emergency scenarios.

Our aim here is to provide a method that supports user in defining behavioural specifications of various forms that can be then mapped into some (maybe partial) executable form to feed existing simulators. This gives us as an advantage to keep modeling specifications still independent from technology and the possibility to project the model created onto a simulation program for a specific simulator, based on the type of analysis that one intends to perform.

For this reason we propose the CEML behavioural modeling framework consisting of the CEML language and an incremental method to specify behaviour. This method is based on the stimulus-response model [4], the event-condition-action (ECA) paradigm [5], and the Reaction RuleML language [6].

The rest of the paper is organized as follows. Section 2 presents related work in the area. Section 3 presents the CEML behavioural modeling framework. Section 4 describes a behaviour specification example. Finally, Section 5 provides conclusions and future research directions.

2 Related Work

In this section we present behavioural models implemented by existing principal simulation tools.

The Ptolemy [7] project concerns modeling, simulation, and design of concurrent, real-time, embedded systems. The result of this project is Ptolemy II, an open-source software framework based on actor-oriented design. Actors are software components that execute concurrently and communicate through messages sent via interconnected ports. Ptolemy II supports several models of computation (e.g., process networks, discrete-events, and continuous-time models). Behaviour can be specified in terms of finite state machines [8], actors whose behaviour is described using a finite set of states and transitions between the states.

NetLogo [9] is a programmable modeling environment for simulating natural and social phenomena. NetLogo is particularly well suited for modeling complex systems developing over time. Modelers can give instructions to thousands of "agents" all operating independently. This makes it possible to explore the connection between the micro-level behaviour of individuals and the macro-level patterns emerging from the interaction of many individuals.

DEVS (Discrete Event System Specification) [10] is a modular and hierarchical formalism for modeling and analysing general systems. These systems can be: discrete event systems, where behaviour might be described by state transition tables; continuous state systems, where behaviour might be described by differential equations; and hybrid continuous state and discrete event systems.

Finally, i2Sim [1] is a simulation platform to model, simulate, and analyse different infrastructures interacting with each other simultaneously and dynamically along the time line. It allows to reproduce cascading effect and to find critical points and vulnerabilities. A key aspect is that behaviour is specified using Human Readable Tables (HRTs) relating outputs for a number of possible inputs.

Ptolemy, NetLogo and DEVS are general purpose tools allowing behaviour modeling of complex scenarios. With respect to them, in order to better support modeler in specifying behaviour, i2Sim and our approach focus on a specific domain, i.e., critical infrastructures and crisis management. Finally, with respect to i2Sim, we extend the expressive power of HRTs, that are just numerical functions, by adopting the expressive power of reaction rules [6].

3 The CEML Behavioural Modeling Framework

As mentioned in the introduction, the CEML language needs to be somehow integrated to support modeling of behavioural aspects of a crisis scenario. In fact, to map a CEML model into a simulation program, its description has to be completed with a behavioural specification. This can be done both at level of construct (local level), which means associating each element of the model (i.e., each instance of a

CEML meta-class that has been selected for a model) with a formal description of the computational steps that it can perform during a simulation run, and at diagram or pattern (global) level, to represent collective behaviour of some elements, for example collaboration rules to manage a specific crisis situation. Here we focus on the local level.

The CEML behavioural modeling framework consists of the CEML language and an incremental method to specify behaviour. The CEML language is used mainly to create a structural model identifying elements acting in the crisis scenario and the structural relationships between them. Then the proposed incremental method provides the guidelines to support users in specifying behaviour of CEML model elements. First they are requested to specify how an element respond to a given stimulus. Then, for each stimulus-response pair, they are requested to further specify behaviour using the event-condition-action rules paradigm. Finally, this specification can be easily transformed into Reaction RuleML code to allow interoperability of behavioural models. For the sake of space, a detailed description of the transformation tool is out-of-scope in this paper. In the following we describe the language and the incremental method.

3.1 The Crisis and Emergency Modeling Language (CEML)

CEML [2] is an abstract level language to model crisis and emergency management scenarios. The related CEML meta-model consists of a set of modeling constructs, a set of relationships, a set of modeling rules, and its formalization using SysML [11] and OCL [12]. CEML supports structural modeling of a crisis and emergency scenario.

The modeling constructs are *abstract service*, specialized into *service*, *communication service*, and *human service*, *external event*, *user*, *message*, *resource*, and *connectivity*. The relationships are *resource flow*, *connectivity flow*, *message flow*, *abstract port*, *communication port*, *message port*, *resource port*, *connection port group*, and *impact*.

A complete description of the CEML is presented in [2].

3.2 Modeling CEML Behaviour According to the Stimulus-Response *Model*

Here we focus on *reactive* behavioural descriptions, that is, the behaviour of some entity consists of a set of actions performed as a response to "something interesting that happens", like a change in the environment, a communication received from another entity, or a change of an internal state. We refer to this model as *stimulus-response* model in the general sense, to include both stimuli coming from the external world and those issued by the internal status of the entity.

In the following, we present a conceptual classification of stimulus-response types for the CEML constructs following two dimensions. The first is related to the *nature of the stimulus*, that is, it refers to the possible reason of a behavioural activation for a given entity. The second pertains the *type of response*, that is, the way this change is carried out, in terms of elementary operations that the entity may perform and/or the effects of the change. For the sake of space, here, we focus only on the following constructs: *abstract service*, *user*, and *external event*.

3.2.1 Abstract Service Behavioural Modeling

In CEML an abstract service represents the active entity processing either a resource entity or a message entity or a connectivity entity. It can be either a service (e.g., power house) or a human service (e.g., fire brigades) or a communication service (e.g., telecommunication provider).

Abstract Service Stimulus

The external stimulus is originated from the external environment. It is related to either the receipt of an input (i.e., resource, connectivity, and message) or an external event or time. Basically, it can be classified as follows:

- **Resource-based stimulus**. It allows to model a stimulus due to input resources (e.g., incoming of needed goods).
- **Connectivity-based stimulus**. It allows to model a stimulus related to connectivity (e.g., availability of wireless connection).
- **Message-based stimulus**. It allows to model a stimulus related to 0..n input messages (e.g., incoming of a request message). It also depends on the availability of a connectivity between the service and the message's source.
- **External event–based stimulus.** It allows to model a stimulus related to exceptional changes in the environment (e.g., due to the epicentre of an earthquake near a power plant).
- **Time-based stimulus**. It allows to model a stimulus generated at a given time of the simulation (e.g., at 11:00 of the simulated time). In fact, the service may perform some activities at given points in time, for example, a service can activate QoS monitoring actions at given time-steps.

An internal stimulus depends on internal events such as an alert on the operation level of the service or a change in the organization. Thus, the types of the internal stimulus may be:

- **Operation-based stimulus**. It allows to model an internal stimulus due to the operational status of a service (e.g., a service partially working).
- **Policy-based stimulus**. It allows to model a stimulus based on an organizational/business rule (e.g., all public services stop working at 10:00 p.m.).

Abstract Service Response

The response to a stimulus is a description of what service does as a consequence of some stimulus. This includes producing (and issuing) an output (i.e., resource, message, and connectivity), performing some internal actions, and changing state/operation mode. Please note that a response may cause the generation of another (type of) stimulus.

Here we present a more detailed classification of the elementary responses whose aggregation leads to complex behavioural descriptions.

The first point leads to a classification of the actions performed by the service that are visible by the external world, namely:

- **Producing a resource**. It allows to model production of a resource (e.g., production of energy).
- **Producing a message**. It allows to model production of a resource (e.g., production of a request message).
- **Issuing a message**. It allows to model issuing of a message. It also depends on the availability of a connectivity between the service and the message's destination.
- **Providing a resource**. It allows to model provision of a resource (e.g., energy).
- **Providing connectivity**. It allows to model provision of connectivity (e.g., wireless connection).
- **Changing internal state**. It allows to model change of an internal state (e.g,. change of service production rate).
- **Changing operation mode**. It allows to model change of operation mode (e.g., from a service not working to the same service working at full steam).

Furthermore the above responses are usually implemented in the terms of one or more *behavioural chunks*, i.e., the elementary operations needed to fulfill a response to a given stimulus. In the following we list them:

- **Condition check.** It allows to specify a logical expression to be evaluated and its parameters (e.g., if the input message is a request). This may represent an invariant on the input values or a business/organization condition (for example, related to a cost/revenue analysis), or a functional and/or QoS monitoring check, like operation performance.
- **Computation of a mathematical function**. It allows to specify the mathematical function that has to be evaluated and its parameters (e.g., $y = f(x_i)$ where y is an output resource and x_i are input resources). These may be either attribute values of the service or time-dependent parameters.
- **Data update.** It allows to specify one or more attributes of the service whose values have to be updated. This might also include the internal time clock and/or knowledge update (e.g., update of the production rate property).
- **Change of operational status**. It allows to specify that operational status of the service has to be updated. This is the behavioural chunk necessary to implement the "changing operation mode" response.

3.2.2 User Behavioural Modeling

In CEML, an user represents someone (or some abstract entity) that benefits from the collaboration of a set of services and makes actions depending on his/her/its wellness level, which measures the benefit received.

User Stimulus
The **external stimuli** refer to receipt of inputs. Therefore, they may be: *resource-based*, *connectivity-based* and *message-based*, as described for the abstract service (see above). The **internal stimulus** is originated by wellness level.

User Response
The response may be *issuing a message* (see above) or *changing wellness level* (e.g., from *satisfied* to *not satisfied*). As for the abstract service, response can be implemented in the terms of the above mentioned behavioural chunks, except "change of operational status".

3.2.3 External Event Behavioural Modeling

In CEML an external event represents an active entity (e.g., failure, earthquake) affecting either the operational status of an abstract service or the wellness of an user. The behavioural model of the external event is motivated by the fact that sometimes in simulations of crisis and emergency scenarios one could be interested in the analysis of the impact of such an event not just at the instant of its occurrence but also over time, in case of a durable phenomenon (for example, an earthquake, tsunami, etc.). In this case, the phenomenon can be described through a time-dependent function modeling its intensity and the behaviour of the external event construct would be concerned with sending suitable signals to services/users of the scenario at a certain time intervals.

External Event Stimulus
Time-based stimulus. (see above)

External Event Stimulus
At the given time, this entity will send signals concerning intensity of the phenomenon.

Sending signal. It allows to describe the signal (e.g., a request message) to be sent and the time instant.

Clearly these elementary stimulus-response types may be combined to obtain more complex response descriptions.

3.3 ECA Implementation of the Stimulus-Response Model

In order to specify how the response to some stimulus is performed and what the output is, we propose to adopt the *Event-Condition-Action (ECA)* paradigm, which is largely used for event-based information systems such as workflow management

systems and also in artificial intelligence for expert and agent-based systems. The ECA foundation of our behavioural notation allows us to easily represent it into an XML format, with the indication of the actual low-level language to be executed by some engine. Also, some standard formats are being developed for rule languages (including ECA) such as the W3C RIF [13] and ReactionRuleML [6] to the purpose of tools interoperability.

The ECA paradigm allows to express *reaction rules*: a set of rules triggered by events, and evaluated under zero or more conditions. Whenever a rule is triggered and evaluated with success (i.e., all of its conditions are true), it can be fired, that is, the action part is executed, which may consist of one or more elementary actions. The ECA rule format follows the following scheme:

on *event* ***if*** *condition* ***then*** *action*

Special cases of ECA rules are the *production rules*, i.e, those with the format:

if *condition* ***then*** *action*

Several engines are available to process production rules, such as JESS [14] and DROOLS [15].

To create ECA rules templates we first need to define a mapping from the stimulus-response theoretical model to the ECA structure. It is quite natural to relate the event part to the stimulus type, and the condition-action part to the response specification. The condition part can be built from the data carried out by the stimulus, e.g., from either properties of an input message or some internal knowledge/data of the entity or related to its state. Indeed, stimuli may be also internally generated as a part of a response. Alternatively, the condition part may be fixed to the *true* value and the action part will be composed by a state-dependent behaviour.

3.4 Reaction RuleML Specification

For the implementation of the rules, we decided to refer here to the Reaction RuleML [6] specification language, an extension of RuleML, which is the *standard de facto* rule specification language and includes an XML rules serialization. Reaction RuleML adds constructs for representing reaction rules and complex event/action messages, and it is intended as a platform-independent rule interchange format, to be translated into specific executable rule languages, like DRL (Drools Rule Language) [15] used in Drools. The general structure of an ECA rule is presented in the Table 1.

The used tags have the following meanings.

- *label*. It is used to specify metadata like: the rule name and a textual description of the rule, to be shown in the editor GUI; the author data; the executable language used to specify the condition and action parts of the rule and/or any tool that might be used for its interpretation.

Table 1 The general structure of an ECA rule

```
<Rule style = "active" evaluation = "strong">
<label > <!– metadata – > </label>
<scope > <!– scope – > </scope>
<qualification > <!– qualifications – > </qualification>
<oid > <!– object identifier – > </oid>
<on > <!– event – > </on>
<if > <!– condition – > </if>
<do > <!– action – > </do>
</Rule>
```

- *scope*. It is used to specify the CEML component instance or scenario fragment to which the rule refer. This reference can be the XML path of the component definition in the XML document containing the structural part of the scenario.
- *qualification*. It is used to specify the priority of the rule.
- *oid*. It is used to specify the identifier of the rule.
- *on, if, do*. They are used to specify, respectively, the stimulus type, the condition check, and the response.

For the description of the < on>,<if>, and < do > parts, based on the classification of the stimulus-response presented earlier, we have defined a XML schema containing the event and response types, (http://enea.utmea-cal/ceml/xsd). For example, the action type ConditionCheck is described by the name of the predicate and a list of parameters, whereas an ExternalEvent-Stimulus type contains the name of the event and other information like intensity and duration.

The clear identification of the rule instances with respect to the stimulus-response type classification supports the user (who might not be a programmer) in the editing of the rule, and also allows us to automatically select the rules for a specific simulation engine.

4 Behaviour Specification Example

Here we briefly present how to specify behaviour of an *UPS* (i.e., Uninterruptible Power Supply) in a crisis scenario due to an earthquake. First modeler has to identify CEML elements (i.e., *UPS* and *earthquake*). Then he has to identify the stimulus type (i.e., *external event-based stimulus*) and the response type for the *UPS* (i.e., changing operation mode). Then he has to decompose the response type in behavioural chunks (i.e., condition check and change operational status). He has to identify the element generating the stimulus (i.e., earthquake) and he has to specify condition (i.e., EARTHQUAKE.intensity == degree 7) and the data to be updated (i.e., UPS. OperationalStatus == ON). At this stage the behaviour of the UPS is specified as follows:

On event *External event stimulus* ***if*** *name = EARTHQUAKE and EARTHQUAKE. intensity == degree 7°* ***then*** *changeOperationMode(UPS,on)*

Table 2 Behaviour representation using Reaction RuleML syntax

```
<!–This example shows a global active ECA reaction rule.–>
<RuleML xmlns = "http://www.ruleml.org/0.91/xsd" xmlns:ceml = "http://enea.utmea-cal/
ceml/xsd" xmlns:cemlScenes = "http://enea.utmea-cal/ceml/scenarios"xmlns:xsi="http:
//www.w3.org/2001/XMLSchema instance" xsi:schemaLocation = "
http://www.ruleml.org/0.91/xsd rr.xsd">
<!–Detect "ExternalEvent" if Earthquake and intensity = 7 do "changeStatus("on")" ISO Prolog
notation eca( detect(CommonBaseEvent:ExternalEvent,P), % event earthquake(P) and
equal(intensity(P),7), % condition state changeOperationStatus(on), %action _, % empty post
condition _ % empty alternative action ).–>
<!– The rule applies to UPS service –>
<oid> <Ind uri = "cemlScenes:ScenarioAquila/Services/UPS"/> </oid>
<Assert> <Rule style = "active" evaluation = "strong">
  <!– event –>
  <on> <!– The rule it triggered by the externalEvent event –>
    <Atom> <Rel per = "value"> detect</Rel>
      <Var type = "ceml:CommonBaseEvent"> ExternalEvent</Var> <Var> P</Var>
    </Atom>
  </on>
  <!– condition –>
  <if> <!– event P is earthquake - and intensity = 7 –>
    <And>
      <Atom><Rel per = "effect"> earthquake</Rel><Var> P</Var></Atom>
      <Atom><Rel per = "value"> intens.</Rel><Var> P</Var><Ind type = "xs:int"> 7</Ind></Atom>
    </And>
  </if>
  <!– action –>
  <do>
    <Atom>
      <!– class/object –><oid><Ind uri = "cemlScenes:ScenarioAquila/Services/UPS"/></oid>
      <Rel per = "effect"> dataUpdate</Rel><Var type = "xsd:String"> OperationalStatus </Var>
      <Var type = "xsd:String"> on</Var>
    </Atom>
  </do></Rule></Assert>
</RuleML>
```

Finally, as an example, we provide the XML representation following the syntax of Reaction RuleML [6] in Table 2. This is an extension of RuleML, the *standard de facto* rule specification language, as it adds constructs for representing reaction rules and complex event/action messages.

5 Conclusion

In this paper we presented a framework to model behaviour in crisis and emergency scenarios. Our proposal consists of the CEML language and method to specify behaviour as ECA rules. We chose Reaction RuleML for XML serialization of the rules so keeping our solution technology-independent and adaptable to executable rule languages.

Finally, the main objective of our work is to provide a formal foundation to build simulation models to be run in appropriate agent-oriented simulators. For this reason we are developing a simulation environment for CEML structural and behavioural models based on the java NetBeans platform and the principles of discrete-event simulation and Model Driven Architectures (MDA) [3] paradigm.

Acknowledgement This work has been partly funded by the European Commission through the Project MOTIA (JLS/2009/CIPS/AG/C1-016).

References

[1] Martí, J.R., Ventura, C.E., et al.. "I2Sim Mod. and Sim. Framework for Scenario Development, Training, and Real-Time Decision Support of Multiple Interdependent Critical Infrastructures during Large Emergencies", NATO, RTA/MSG Conf. on "How is Mod. and Sim. Meeting the Defence Challenges Out to 2015?", 2008.
[2] De Nicola A., Tofani A., Vicoli G., Villani M.L.. "Modeling Collaboration for Crisis and Emergency Management", COLLA 2011 Int. Conference, 2011.
[3] OMG-MDA, "MDA Guide, version 1.0.1," Available at: http://www.omg.org/mda/presentations.htm, 2003. Retrieved on 20th September 2011.
[4] Simon H. A., "The Sciences of the Artificial", The MIT Press. 3rd ed. 1996.
[5] Dittrich K. R., Gatziu S., Geppert A.. "The Active Database Management System Manifesto: A Rulebase of ADBMS Features", LNCS 985, Springer, 1995.
[6] Paschke, A., Kozlenkov, A., Boley, H.. "A Homogenous Reaction Rule Language for Complex Event Processing", 2nd EDA-PS Workshop, Austria, 2007.
[7] Ptolemy: http://ptolemy.eecs.berkeley.edu/index.htm
[8] Lee E. A.. "Finite State Machines and Modal Models in Ptolemy II", Technical report, EECS Department, University of California, Berkeley, UCB/EECS-2009-151, 2009.
[9] Tisue, S., Wilensky, U. "NetLogo: A simple environment for modeling complexity". In Int. Conf. on Complex Systems, 2004.
[10] Gabriel A. Wainer and Pieter Mosterman Eds.. "Discrete-Event Modeling and Simulation: Theory and Applications", 1st ed., CRC Press, 2010.
[11] OMG-SysML, "OMG Systems Modeling Language" version 1.2. Available at: http://www.omgsysml.org/. 2010.
[12] Warmer J. and Kleppe A., "The Object Constraint Language: Getting Your Models Ready for MDA", Addison-Wesley, 2003.
[13] Kifer M., "Rule Interchange Format: The Framework", LNCS 5341, Springer, 2008.
[14] Friedman-Hill, E.. "Jess in Action", Manning, Greenwich, CT, 2003.
[15] Browne, P.: "JBoss Drools Business Rules". Packt Publishing, 2009.