An Ontological Approach to Simulate Critical Infrastructures

Alberto Tofani[a,*], Elisa Castorini[a], Paolo Palazzari[a], Andrij Usov[b], Cesaire Beyel[b], Erich Rome[b], Paolo Servillo[c]

[a] *ENEA, Via Anguillarese 301- S. Maria di Galeria 00123 Roma, Italy - www.casaccia.enea.it*
[b] *Fraunhofer Institut IAIS, Schloss Birlinghoven - 53754 Sankt Augustin, Germany - www.iais.fraunhofer.de*
[c] *CRIAI, Piazzale E. Fermi 1 - Porto del Granatello 80055 Portici (NA), Italy - www.criai.it*

Abstract

This paper presents a Knowledge Base System (KBS) as the key component of a federated simulation framework aimed at investigating the dependencies among Critical Infrastructures (CIs). The KBS, based on the ontological formalism, represents the properties and the relations of each simulation domain and the dependency relations among different domains. Some auxiliary data structures, necessary to model the interaction among the simulators of different CIs, have been defined and have been populated through suitable queries to the KBS. The adoption of the ontological formalism allowed the definition of a common formalism to deal with the heterogeneity arising from the presence of different domains.

Keywords: CI Protection, CI Dependency Analysis, Knowledge Base System, Federated Simulation, Ontology

1. Introduction

The study of Critical Infrastructures Protection (CIP) has been indicated by the European Commission as a fundamental research topic [1] for the development and improvement of the European economy and society. A

*Corresponding author
Email address: alberto.tofani@enea.it (Alberto Tofani)

definition of European Critical Infrastructures can be found in [2]. In spite of its high relevance, the understanding of system of critical infrastructures with all their interdependencies is still immature. The study of these complex infrastructure systems demands joint interdisciplinary efforts of researchers, industrial stake-holders and governmental organisations to overcome all the difficulties involved as availability of models and data for the single infrastructures, interoperable simulation of multiple infrastructures, testbeds and benchmarks for protection solutions.

The various aspects of infrastructure networks present numerous theoretical and practical challenges in modelling, prediction, simulation and analysis of cause-and-effect relationships in coupled systems [3]. These systems comprise a heterogeneous mixture of dynamic, interactive, and often non-linear entities, unscheduled discontinuities, and numerous other significant effects. Then, modelling and analysis of these systems requires consideration of their large-scale, non-linear, and time-dependent behaviour.

The report [4], about European CI disruptions, describes and classifies the cascading effects among European CIs. Due to the high relevance of studying, modelling and simulating CIs and their dependencies, the European Commission has funded the Design of an Interoperable European federated Simulation network for Critical InfraStructures (DIESIS) [5], [6] project with the aim to establish the concepts and methodologies of an e-Infrastructure for the modelling and simulation of CIs. In particular, the project is a feasibility study for the realisation of an European Infrastructures Simulation and Analysis Centre (EISAC)[1] that should have the same aims and functionalities of the American NISAC initiative [7].

Currently, two major general approaches [8] are used in the CIP research field: integrated vs coupled modelling approaches. In the former case, the approach is to create an integrated system model that attempts to model multiple infrastructures and their (inter)dependencies within one framework. On the other hand, within the coupled modelling approaches a series of individual infrastructure simulations are coupled together in order to illustrate the cascading influence between them. In both the cases, the scenario description (CI networks and their interconnections) is usually embedded in the simulation framework and this makes the scenario representation and

[1]The research leading to these results is partially funded by the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n 212830.

the analysis tasks very difficult.

In this work, which summarises some of the results achieved in the DIESIS project, we propose the separation of the scenario representation (*Scenario Definition Phase*) from the simulation framework (*Federated Simulation Phase*). The main idea is to develop a Knowledge Base System (KBS) based on ontologies and rules providing the semantic basis for the *federated* simulation framework [9], [10]. In particular, a federation of simulators can be considered as a System of Systems where each simulator represents an independent entity with its own behaviour and purpose. The super-system considers the interaction of these entities and puts in evidence an emergent behaviour that does not reside in any component system [11]. Therefore, in a federated simulation, the stand-alone simulators must be linked together so that an understandable and meaningful information exchange could be performed. This requires that simulators could interact and cooperate. In order to achieve this kind of interoperability we propose the adoption of ontologies which allow both an uniform modelling of heterogeneous infrastructures and the easy representation of inter-domain dependencies. This work is organized as follows. After some considerations about the literature related to the proposed approach, we introduce the KBS architecture in section 3. Then, a graph formalization of the knowledge represented in the KBS is proposed in section 4. The graph formalization allows to easily define the basic procedures to query the KBS used within the DIESIS framework and the main data structures that allow the semantic interoperability among the considered CI domains. A description of the DIESIS framework main components and data structures could be found in section 5. The distributed simulation framework is described in section 6. The section 7 summarizes the conclusions of the paper.

2. Related works

Several approaches have been explored to model CI interdependencies to enable federated simulations. In particular, the HLA (High Level Architecture) standard [12] and the MSI (Multi-Simulator Interface) program [13], currently are among the most relevant works in this field. However, these approaches do not seem to be suitable for simulations in the infrastructure domain. The coupling of simulators is based on a common data model that has to be implemented by all involved simulators. However, this data model is on a purely syntactical level and does not provide semantic information

about the modelled domain. HLA makes also difficult to integrate different timing models. The work proposed in [9] is conceptually similar to the approach presented in this paper. In fact, both address the issue of knowledge reuse and both are based on an ontological framework able to realise a semantic layer between CI concepts and computer systems, independently from technology, architecture and applications. Unlike the work presented in [9], that uses simulation models, our approach refers to infrastructure domain models. This allows to gain more generality in relation to the software simulator to be used, increasing the reuse potentiality of such models for other simulators. Another similar approach is investigated in [14], where ontologies are used to face the problem of semantic inaccessibility when different simulation tools are used in a federated environment. This approach allows significant increases in the quality of information sharing and communications for distributed modelling and simulation application. The requirement to use pre-existing ontology models implies the need to perform a complex comparative ontology analysis including the management of concept mismatches. Our work proposes a KBS able to create an abstraction of the considered domains, to represent and formalise parameters and dependencies, which are relevant for the problems to be investigated. The strength and innovation of our approach lies in the creation of a common level of abstraction for the systems involved, where the semantic of dependencies can be formalised in a simulator-independent fashion. Moreover, the work presented in this paper realises a real integration of the ontological framework inside a federated simulation framework, by introducing further components interfacing specific domain simulators and managing/synchronising the federated simulation session. Moreover, the proposed approach has been validated using a realistic scenario that involved three CI domains (electrical, telecommunication and railway) and a flood simulator as generator of external events. In particular, the select scenario concerned a specific district of Rome.

3. KBS

The KBS establishes a common formalism, for scenario and domain knowledge experts, to represent the main aspects, elements and properties of the considered domains and their interconnections. The KBS is based on the Ontology Web Language (OWL) and the ontologies are defined through proper specifications of *classes*, *properties* and *individuals* (instances of classes). The individuals represent the physical/logical entities that form the universe of

a specific domain. For instance, a specific electrical load is an individual of the electrical ontology. The OWL allows to group individuals in classes that define the properties shared by all their individuals. The properties can be used either to specify relationships between individuals and data type values (*Datatype Properties*) or to describe relationships among individuals (*Object Properties*). Then, if we denote with Pr the set of properties we have

$$Pr = DP \cup OP, \text{ with } DP = \{p \mid p\, Datatype\, Property\}$$

$$\text{and } OP = \{p \mid p\, Object\, Property\}$$

The KBS architecture includes the following ontology definitions:

- World ONTology (WONT);
- Infrastructure domain ONTologies (IONTs);
- Federation ONTology (FONT).

The WONT is a general template providing the basic structures and rules to define CI domains. In particular, the WONT allows the definition of CI domain elements (through the WONT class *Component*), their physical and logical interconnections (through the WONT object property *isConnected*) and the dependencies among different CI domains (through the WONT object property *dependsON*).

The IONTs inherit the basic template from the WONT and represent the specific knowledge of the critical infrastructure domains. For instance, the electric IONT class *Load* (that models the electric load element) and the telecommunication IONT class *Switch* (that models the telecommunication switch element) are subclasses of the WONT class *Component*. In addition, the railway IONT property *isLinked* that models the connection between two railway stations is a sub-property of the WONT property *isConnected*; similarly, the telecommunication IONT property *transmits* that models the connection between a transmitter and a receiver is another sub-property of the WONT property *isConnected*. Analogously, all dependencies among the considered CI domains are modelled through ad-hoc designed sub-properties of the WONT property *dependsON*.

In the following, given a CI domain X_i, C_i indicates the set of all components of X_i and Pr_i indicates the set of properties related to the components of X_i. Then, a generic IONT can be represented as $IONT_i = \{C_i, Pr_i\}$.

Once the IONT has been defined to model a particular domain, it is possible to create individuals (instances of IONT classes) to represent actual network domains (for example the electrical power network or the telecommunication network of a specific city district). Similarly to the IONT definitions, we indicate with C_i^* the set of the all instantiated components belonging to the domain X_i and with Pr_i^*

the set of instantiated properties related to X_i. Then, the IONT instance $IONT_i^*$ can be expressed as $IONT_i^* = \{C_i^*, Pr_i^*\}$.

The FONT includes all IONTs of the domains involved in the considered scenario (our scenario entails the electrical, the telecommunication and the railway domains). The FONT properties (sub-properties of the WONT property *dependsON*) allow to model dependencies among components of different domains (e.g. the FONT property *feedsTelco* models the electrical supply of telecommunication nodes). The sets of the FONT properties and of the FONT instantiated properties are defined respectively as:

$$FPr = \{sp(a,b) \mid sp \text{ sub-property of } dependsON, a \in C_i, b \in C_j, i \neq j\} \text{ and}$$

$$FPr* = \{sp(a,b) \mid sp \text{ sub-property of } dependsON, a \in C_i^*, b \in C_j^*, i \neq j\}$$

The Figure 1 summarizes the proposed KBS architecture

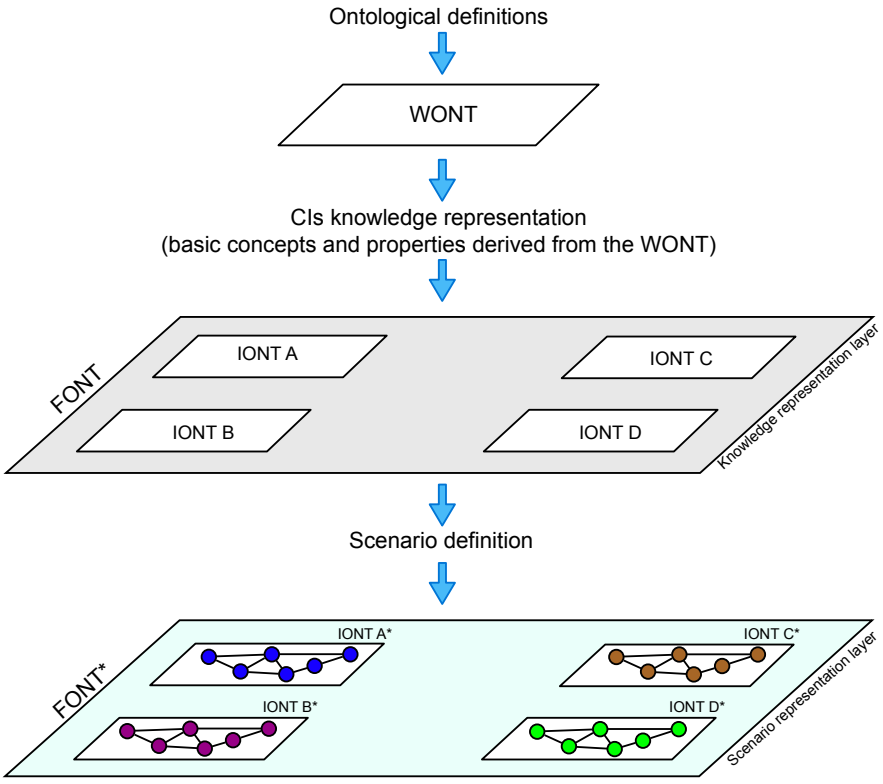

Figure 1: KBS architecture.

The FPr^* allows the definition of FONT rules that express the semantic dependency. The FONT rules, expressed using the Semantic Web Rule Language (SWRL), have been actually translated into their JAVA counterpart and implemented through "if-then-else" constructs embedded within the Federation Managers which incapsulate the simulators of each CI domain (Federation Managers are explained in Section 5). As explanatory example, we can consider the FONT rule which models the dependency of the functioning status {ON ‖ FAULT} of a railway signal ($Signal_{rw}$) on the functioning status {ON ‖ OFF} of the communication channel ($Channel_{tlc}$) used to tele-control the signal:

$$rwIONT : signal(?x) \wedge tlcIONT : node(?y) \wedge wont : dependsOn(?x, ?y)$$
$$\wedge wont : hasFunctioningStatus(?x, wont : off)$$
$$\rightarrow wont : hasFunctioningStatus(?y, wont : fault)$$

This rule, based on the FONT property *teleControl*, is implemented as follows:

$$\textbf{if } (Channel_{tlc} == OFF) \textbf{ then } Signal_{rw} = FAULT$$

The DIESIS distributed simulation framework and the proposed data structures (see Section 5) allow the components (involved in the defined FONT properties) to exchange the functioning status values.

4. Scenario Graph

Using the KBS, it is possible to derive a directed graph $G = (V, E)$ where V is the set of nodes that represent all the individuals and E is the set of edges that represent all the physical and dependency connections. Considering the IONT instance $IONT_i^*$ corresponding to domain X_i, we denote with x_i^j the j^{th} individual of the $IONT_i^*$.

The set of edges E is partitioned in two subsets P and D ($P \cup D = E$, $P \cap D = \emptyset$). The edge $(v_i, v_j)_p \in P$ represents a *physical* or *logical* connection between the components v_i and v_j of a domain X_i. The edge $(v_i, v_j)_d \in D$ represents the *dependency* connection between the components v_i and v_j belonging to different IONT instances: for example, considering the telecommunication domain, an edge $(v_i, v_j)_p$ may represent a physical link between a router v_i and a workstation v_j. On the other hand, if a medium voltage power electric station v_i feeds a telecommunication base station v_j, this kind of dependence is represented through an edge $(v_i, v_j)_d$.

The set of edges E is derived using the definition of the properties *isConnected* and *dependsON* in the KBS (see Section 3). The procedures to derive the P and

D sets are shown in Figure 2 and Figure 3 respectively. These procedures use the following definitions. If S is a property, the predicate $S(a, b)$ is true for all the pairs of individuals (a, b) that satisfy the property S. Moreover, $E(S)$ denotes the set of all the individual pairs in E that satisfy S i.e.

$$E(S) = \{(a, b) \mid (a, b) \in E, S(a, b) \equiv true\}$$

get_P_GraphEdges
$P := \emptyset$
for all sub-properties S_n of the properties *isConnected* **do**
  for all pairs of individuals $(a, b) \in E(S_n)$ **do**
    $P := P \cup (a, b)_p$
  end for
end for

Figure 2: The *get_P_GraphEdges* procedure.

get_D_GraphEdges
$D := \emptyset$
for all sub-properties S_n of the properties *dependsON* **do**
  for all pairs of individuals $(a, b) \in E(S_n)$ **do**
    $D := D \cup (a, b)_d$
  end for
end for

Figure 3: The *get_D_GraphEdges* procedure.

Finally, given a scenario graph $G = (V, E)$ and the set D of the dependency connections among components of different IONT instances, for each $v_i \in V$ it is possible to define the following sets of vertices:

Definition The set of *Source Vertices SV* for v_i is defined as the set of vertices that are connected to v_i through *incoming* dependency connections i.e.
$SV(v_i) = \{v_k \mid \exists\, (v_k, v_i)_d \in D\}$.

Definition The set of *Target Vertices TV* for v_i is defined as the set of vertices that are connected to v_i through *outgoing* dependency connections i.e.
$TV(v_i) = \{v_k \mid \exists\, (v_i, v_k)_d \in D\}$.

The sets *Source* and *Target Vertices*, that are defined considering only the dependency connection edges, are fundamental to the definition of particular data structures (described in Section 5) that allow the correct exchange of information among the different CI domains.

5. DIESIS architecture and data structures

The DIESIS federated simulation framework consists of domain simulators, Federated Managers and the Federation Control Module (FCM). Each $IONT_i$ defined in the KBS corresponds to a domain simulator that has to be embodied within a specific Federation Manager to be compliant with the behaviour defined for the DIESIS federated simulation environment. The Federation Manager is in charge to interface the specific domain simulator with the Federation Managers of the other domain simulators, with the KBS and with the Federated Control Module which is a centralised component that manages and synchronises a federated simulation session. The synchronisation and federated simulation management procedures are described in Section 6.

A DIESIS simulation session is formed by a number of domain simulators that have to be federated in order to obtain a global realistic scenario simulation. The proposed KBS (and the related scenario graph G) plays a central role in the definition and setting of this environment.

5.1. Dependency lists

In order to manage the dependencies among the considered CI domains, suitable data structures for Federation Managers have been defined. The state of a component in a domain X_i can influence (in this case it is a source) and/or be influenced (in this case it is a target) by the state of components belonging to other domains. To force such a dependency behaviour, the Federation Managers use the *Source Dependency List (SDL)* and the *Target Dependency List (TDL)* data structures. The *SDL* and *TDL* data structures are based on the definitions of *Source* and *Target Vertices* of Section 4. The procedures to create the *SDL* and *TDL* data structures are described in Figure 4 and in Figure 6.

6. Federated simulation controller

6.1. Definition and tasks

A CI simulator used to represent a particular domain in the federated simulation network is usually a stand-alone application which uses its own fixed time model as well as a fixed structure of the CI model and works completely independent from other domain simulators. The federated simulation controller is an

```
createSDL(X_i, G)
begin
SDL_i := ∅
V_{X_i}:=getVerticesOfDomain(X_i)
for all v_j ∈ V_{X_i} do
  for all v_k ∈ SV(v_j) do
    SDL_i:=SDL_i ∪ v_k
  end for
end for
end
```

Figure 4: The *createSDL*(X_i, G) procedure.

adapter which connects a particular CI simulator (a "federate") to other federation nodes (other simulators or technological components like KBS and the federation controller module). Its primary task is to manage and synchronise the common simulation time and event forwarding between the simulators in a network. Furthermore it allows extending the simulator model by additional components if they are needed for the definition of dependencies between the CIs. Another important property of the federation simulation controller is the ability to encapsulate and automate any kind of communication required for the management of a federated simulation. The adapters can communicate directly via TCP/IP as well as via the additional communication layer which is used for the demonstrator and is described in [15]. Figure 5 shows a simulation node (in the middle) and its communication channels (links) to the technological federation nodes. The creation of data links to other federates (to submit the simulation data) is automatically done by a federation adapter and is based on the dependency lists acquired from the KBS via the link "GetDependencies".

Immediately before the federated simulation is started the federation simulation controller receives a list of other node locations (IP addresses and the application IDs) from the FCM. This allows it to open direct peer-to-peer connections to other federates during the simulation. Moreover it queries the KBS to load the proper source and target dependency lists for its domain. Altogether the federation simulation controller disposes of the following important variables:

1. A node locations list to establish peer-to-peer connections to other federates.
2. A list of local state variables which can be affected by events produced by other federates and has to be synchronised with the simulator's model elements.
3. A target dependency lists for the simulated domain in order to send the

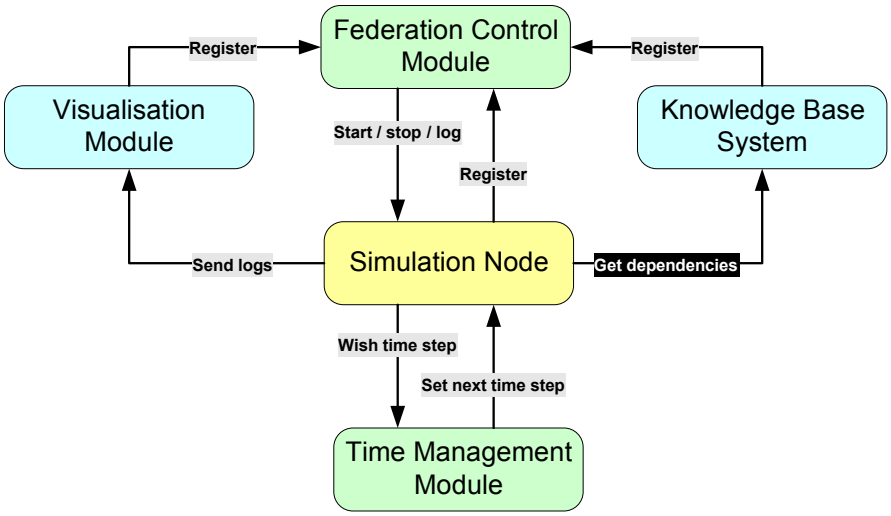

Figure 5: Links and services provided by a federation adapter.

changes of local variables (detected by monitoring the local variables' list) to proper dependent domains.

4. A cache for the changed values of foreign state variables and a source dependency list to ensure completeness of the received information about the external events.
5. A clock for the federation-wide simulation time. This clock has to be synchronised with the internal simulator's time as well as with the external common simulation time.

The next section shows how this data is used by a federation simulation controller to manage and synchronise the simulation time and non-local events.

6.2. Distributed time and event synchronisation

Though the DIESIS architecture concept offers some more flexible alternatives to HLA's concepts e.g. through the introduction of the lateral coupling and service network concepts [16], the time management of DIESIS, thus far realised in the scope of the Technical Proof of Concept (TPoC), is comparable to the conservative time management strategy as defined in HLA [17], [18]. This concept can however be extended through additional and alternative bridges for a pair- or group-wise synchronisation. The following section describes the distributed time and event synchronisation as implemented in the DIESIS TPoC. The concept of the common simulation time is mandatory for the cooperation of different simulators (the

```
createTDL(X_i, G)
begin
TDL_i := ∅
V_{X_i}:=getVerticesOfDomain(X_i)
for all v_j ∈ V_{X_i} do
  for all v_k ∈ TV(v_j) do
    TDL_i:=TDL_i ∪ v_k
  end for
end for
end
```

Figure 6: The *createTDL*(X_i, G) procedure.

"federates"). It ensures the correctness of the event order even if some simulators do not use the time concept for their internal computations (as it is the case for steady state simulators) and is also required for the visualisation and analysis.

The federation simulation controller makes the entire node working event based from the point of view of the federation. To achieve this goal the federation simulation controller has to be able to control the internally used simulator by making it to perform simulation steps of a pre-defined variable length (a "step" from the point of view of the federation may require several internal simulation steps or runs) and to pause or stop the local simulation to wait for external results. For the realisation of such a solution, the requirements of the simulator and its API are: (1) the adapter should be able to pause or stop the simulation as soon as the given internal simulation time is reached. If the simulation was stopped (and not paused) the ability to restart the simulation using the previously computed model state as a new initial state is required, (2) it should be possible to compute a simulation time interval from the current time point to the next event (or a breakpoint) in the future. Alternatively, a possibility to restore the previous model state and to restart the simulation using it as an initial state should be supported.

This approach is applicable to a large number of the currently available CI simulators that use various time models (like the event based model, constant time steps, steady state, etc.) and allows the synchronisation of the controller's clock with the simulation time of the internal simulator. A Time Management Module (TMM) is a special federation node which is used to synchronise the internal clocks of all federation. For the demonstrator the TMM was implemented as an integral part of the federation control module. Each federation node computes its desired value for the next time step (a time to the next scheduled event) and sends it to the time management module. If no events are scheduled a node submits a negative

value (this denotes that it does not care about the duration of the next simulation step). As soon as the time management module has collected the suggested time steps from all federates, it chooses the minimal non-negative value and sends it back to the federates. The federation adapters increase their internal clocks by this minimal value and are able o perform a globally identical simulation step. Receiving only negative values by the TMM denotes a global steady state. In this case the simulation can be automatically stopped. This mechanism forces all federates to have identical simulation clock values at the beginning of the same simulation step.

Figure 7 shows the algorithm of time and event synchronisation for a particular federation node for a domain i. Here *ExternalChanges* represents the cache that receives and stores the changes of the external state variables represented in the SDL_i (as described in the previous section), *SimulationTime* is the federation adapter's clock and *NextStep* denotes the duration of the next simulation step. The federation simulation controller checks the ExternalChanges for new or modified values of the external state variables. If it detects those modifications (received in the previous simulation step) it applies the FONT rules for the simulated domain (e.g. computes the effect of these changes to its local variables). After that, the following two tasks will be performed in parallel:

1. Simulate the time interval given by the previously computed value of NextStep, estimate a new value for it and report resulting changes of the local variables to the nodes represented in its target dependency list TDL_i. If no model changes were computed, an empty change list will be submitted.
2. Wait for the model change reports from all federation nodes from the SDL_i (including the empty change lists) and store them in the list *ExternalChanges*.

If some external model changes in a domain from SDL_i were received (*ExternalChanges* is not empty) then this may require an immediate modification of the local state variables. This modification may on its part cause some events which have to be propagates to other domains from the TDL_i and handled at the same simulation time. Therefore the value of *NextStep* will be set to zero, otherwise all potential events produced by the local domain will be handled by other federates not "immediately" (e. g. at current simulation time) but "too late" (e. g. delayed by the next non-zero time step). The last action in the simulation loop is sending the suggestion for the next step to the TMM, receiving the federation-wide minimal value, and adding it to the internal simulation clock. The simulation loop can be interrupted at any time if a *StopSimulation* message from the FCM has been received.

```
simulateAndSynchronise
begin
NextStep := 0
SimulationTime := 0
ExternalChanges := ∅
repeat
  if ExternalChanges ≠ ∅ then
    Apply FONT rules for my domain
  end if
  parallelised do
    Perform simulation step of length NextStep
    if next event can be scheduled then
      NextStep := time to the next scheduled event
    else
      NextStep := −1
    end if
    for all dependent domain X represented in my TDL do
      Send modified local state variables or ∅ to the federate X
    end for
  in parallel to
    ExternalChanges := ∅
    repeat
      Receive external state variable changes ExtVars from other federates
      ExternalChanges:=ExternalChanges ∪ ExtVars
    until received messages from all domains represented in my SDL
  end parallelised
  if ExternalChanges ≠ ∅ then
    NextStep := 0
  end if
  Send NextStep to the time management module
  NextStep := value received from the time management module
  SimulationTime := SimulationTime + NextStep
until a StopSimulation message has been received from the FCM
end
```

Figure 7: The *simulateAndSynchronise*(X_i, G) procedure.

6.3. Other time and event synchronisation strategies

The federation simulation controller that has been implemented for the TPoC of DIESIS is flexible enough to enable the coupling of a large number of currently

available CI simulators. The limitation of this approach is the usage of a conservative time management strategy and of a fixed communication concept. The DIESIS architectural approach, however, does not prescribe the usage of a federation adapter that has been described above in this section (for details cf. [16]). The DIESIS architecture foresees the development an individual realisation of communication bridges (links) between particular simulators. A software component like a federation adapter may facilitate the integration of new simulators into the federation if and only if those simulators support (or can be adapted to support) its concepts for time, events and communication. These adapters for coupling two simulators and also the links they consist of can be reused for other scenarios and simulation tasks, if the same simulators are involved again. However, for some tasks it is sensible to employ another type of a federation adapter, to use it for a set of federates (not only for a single simulator) or not to use it at all. It is, for example, needless to create federation adapters to interconnect two simulators both already supporting another coupling technology (like HLA). The existence of various time models within a federation (like optimistic look-ahead schemes) requires the development of an individual coupling solution based on the methodology described in [16]. This may result in partially centralised models but this is not a requirement.

7. Conclusion

In this paper we summarise some of the main results achieved within the DIESIS (Design of an Interoperable European federated Simulation network for critical InfraStructures) european project. In particular, we describe the Knowledge Base System (KBS) developed to model Critical Infrastructures (CIs), their interconnections and dependencies and the procedures and data structures defined to implement the federated and distributed simulation of CIs. The KBS is based on the ontological framework and is used to model both the characteristics of each CI and the interdependencies among the CIs.

The main advances w.r.t. the state of the art in CI interdependencies modelling and simulation approaches are: (1) the creation of an abstraction of the considered domains to represent and formalise parameters and dependencies which are relevant for the problems to be investigated, (2) the proposal of a flexible federated simulation framework supporting the explicit separation of the *Scenario Definition Phase* from the *Federated Simulation Phase*. Indeed, the proposed federated simulation framework and the KBS have been used in actual federated simulation experiments, to study the global behaviour of the resulting "System of System" under different conditions and constraints (e.g. parameter variations or changes of the environmental conditions). The proposed approach has been validated using

a realistic scenario that involved three CI domains (electrical, telecommunication and railway) and a flood simulator as generator of external events. In particular, the select scenario concerned a specific district of Rome and the population of the KBS consisted on the instantiation of the IONT components specialised with their actual characteristics.The electric and telecommunication IONT instances have been populated with realistic data coming from a previous European project [19], whereas the railway IONT has been populated by acquiring public available data and knowledge about the domain. The proposed KBS template allowed to define the three CI models (representing the knowledge about the domains), the related networks (the actual network domains) and the dependencies among them. Moreover, the proposed simulation framework allowed to effectively federate the domain simulators and it was possible to show the propagation of the fault effects generated by a flooding of the area where the considered domain networks are located.

References

[1] Council Directive 2008/114/EC of 8 December 2008.

[2] Communication from the Commission COM(2006) 786, 2006.

[3] S. M. Rinaldi, J. P. Peerenboom, T. K. Kelly: *Critical infrastructure interdependencies*, IEEE Control Systems Magazine 21(6), 2001.

[4] E. Luiijf, A. Nieuwenhuijs, M. Klaver, M. van Eeten, E. Cruz: *Empirical findings on critical infrastructure dependencies in europe*, CRITIS Proc., 2008.

[5] DIESIS project website: *www.diesis-project.eu*.

[6] E. Rome, S. Bologna, E. Gelenbe, E. Luiijf and V. Masucci: DIESIS – *Design of an Interoperable European Federated Simulation Network for Critical Infrastructures*, In: *Proceedings of the 2009 SISO European Simulation Interoperability Workshop (EURO SIW '09)* – July 13-16, 2008, Istanbul, Turkey, pp. 139–146, SCS, San Diego, CA, USA, 2009.

[7] NISAC website: *www.sandia.gov/nisac*.

[8] P. Pederson, D. Dudenhoeffer, S. Hartley, M. Permann: *Critical infrastructure interdependency modeling: A survey of U.S. and international research*. Tech. report, Idaho National Laboratory, 2006.

[9] T. Rathnam, C.J.J. Paredis: *Developing Federation Object Models using Ontologies*, Proceedings of the 36th Winter Simulation Conference, 2004.

[10] R.K. McNally, S.W. Lee, D. Yavagal, W.N. Xiang *Learning the critical infrastructure interdependencies through an ontology-based information system*, Environment and Planning B: Planning and Design, 34, pp. 1103-1124, 2007.

[11] M.W. Maier: *Architecting principles for systems-of-systems*, Systems Engineering, 1 (4): 267-284, 1999.

[12] J. S. Dahmann, R. M. Fujimoto, R. M. Weatherly: *The department of defense High Level Architecture*. Proceedings of the 1997 Winter Simulation Conference.

[13] A. Rubin, C. Hein, G. Prasad: *Multi- Simulation Interface (MSI) for Complex Simulations*.

[14] B. Perakath, A. Kumar, V. Ajay: *Using ontologies for simulation integration*. Proceedings of the 2007 Winter Simulation Conference.

[15] G. Görbil, E. Gelenbe: *Design of a Mobile Agent-Based Adaptive Communication Middleware to Enable Federations of Critical Infrastructure Simulations*, CRITIS Proc., 2008. In: Pre-proceedings of the fourth International Workshop on Critical Information Infrastructures Security (CRITIS '09), Erich Rome, Robin Bloomfield (eds.), Sankt Augustin: Fraunhofer IAIS, 2009, pp. 145–160. Conference: Bonn, Germany, September 30–October 2, 2009.

[16] Public DIESIS deliverable available at: http://www.diesis-project.eu/include/Documents/DIESIS_Final_Architectural_Design.pdf

[17] R.M. Fujimoto: *Time Management in the High Level Architecture*, SIMULATION Special Issue on High Level Architecture, vol. 71, no. 6, 388-400, 1998.

[18] C. D. Carothers, R. M. Fujimoto, R. M. Weatherly, A. L. Wilson: *Design and Implementation of HLA Time Management in the RTI version F.0*, Winter Simulation Conference, December 1997.

[19] IRRIIS project website: *www.irriis.org*.