Discrete-Event Modeling and Simulation of Diffusion Processes in Multiplex Networks

CRISTINA RUIZ-MARTIN and GABRIEL WAINER, Systems and Computer Engineering, Carleton University, Ottawa, Canada
ADOLFO LOPEZ-PAREDES, INSISOC, Universidad de Valladolid, Spain

A variety of phenomena (such as the spread of diseases, pollution in rivers, etc.) can be studied as diffusion processes over networks (i.e., the diffusion of the phenomenon over a set of interconnected entities). This research introduces a method to study such diffusion processes in multiplex dynamic networks. We use a formal Modeling and Simulation methodology (in our case, DEVS, Discrete-Event System Specification). We use DEVS formal models to integrate models defined using Agent-Based Modeling and Network Theory. We present (1) an *Architecture* to study Diffusion Processes in Multiplex dynamic networks (ADPM) and (2) a systematic *Process* to define, implement, and simulate diffusion processes over such networks. We show a theoretical definition and a concrete implementation of ADPM. We show how to use ADPM and the process in a case study based on a real nuclear emergency plan; this illustrates the application of the process, the architecture, and the developed software. Different scenarios are studied as Diffusion Processes to demonstrate the usability of ADPM.

CCS Concepts: • **Computing methodologies → Discrete-event simulation**;

Additional Key Words and Phrases: Dynamic multiplex networks, formal modeling and simulation, network theory, agent-based modeling, DEVS

ACM Reference format:
Cristina Ruiz-Martin, Gabriel Wainer, and Adolfo Lopez-Paredes. 2020. Discrete-Event Modeling and Simulation of Diffusion Processes in Multiplex Networks. *ACM Trans. Model. Comput. Simul.* 31, 1, Article 6 (December 2020), 32 pages.
https://doi.org/10.1145/3434490

1 INTRODUCTION

Understanding phenomena such as the spread of diseases, the adoption of technological innovations, or the diffusion of news (fake and real) is important for policymakers [1, 2]. They need a comprehensive understanding of these phenomena to make sound policies and contingency plans. For example, with diseases, it is important to know how the process works and what the effects of different policies are. They may want to know if it is necessary to vaccinate the population

Authors' addresses: C. Ruiz-Martin and G. Wainer, Department of Systems and Computer Engineering, 1125 Colonel By Dr, Carleton University, Ottawa, Ontario, Canada; emails: {cristinaruizmartin, gwainer}@sce.carleton.ca; A. Lopez-Paredes, INSISOC, Paseo del Cauce 59, Universidad de Valladolid, Valladolid, Spain; email: aparedes@eii.uva.es.

1049-3301/2020/12-ART6 $15.00
https://doi.org/10.1145/3434490

or only specific groups, or they could be interested in knowing if infected individuals need to be quarantined. These phenomena can be seen as *diffusion processes*, in which the object of interest (i.e., a virus, an idea, a molecule, etc.) spreads in an environment starting from an area with a high concentration of the object to areas with lower concentrations.

There are different methods for studying diffusion processes, but most are based on various kinds of entities that are interconnected (networks). Methods based on *simplex* networks [3, 4] use a single type of connection between entities. They assume that all connections have the same properties, but this could lead to misleading results (e.g., individuals are not likely to transmit a rumor equally to their colleagues and to their boss). *Multiplex* network methods, in which not all connections are equivalent, are better suited to study diffusion processes [4].

Regardless of how links are represented (*simplex* or *multiplex*), two main approaches have been used to model the diffusion process over the network: Differential Equations (DEs) and Agent-Based Modeling (ABM) [5]. Agent models are popular in Social Science to explore phenomena that can be understood as a diffusion process, such as the emergence of segregation [6, 7], violent groups [8], adaptation of organizations to change [9], population growth [10], or market dynamics [11]. DEs have been used to study innovation [12], epidemiology [13], and business cycles [14], among others. The main advantage of nonlinear DEs is that they can include a wide range of feedback effects (i.e., how the current value of a parameter of the system affects its future value, as in closed-loop systems). However, when they are used to study diffusion processes, one typically needs to aggregate nodes into fewer states or categories [5]. Instead, ABM uses different attributes in each category, and different nodes in the same category may have different behavior. The network structure is clearly defined, and the behavior of each node is modeled individually at an increased computational cost [5].

Likewise, neither of these two approaches provides well-established modeling and simulation (M&S) mechanisms for incorporating diffusion algorithms into multiplex dynamic networks and run simulations. For example, Xiong et al. studied the effect of the diffusion of innovation into social networks using simulation [15]. They defined models for the behavior of the nodes, but they did not include details on integrating the network into a computerized model or the simulation algorithm, nor on implementation. In most cases, the methodology used to build the model, the simulation platform, or the simulation algorithms are not specified. Models are built *ad hoc* and they are mixed with the simulation and the experiments. This lack of separation of concerns leads to problems; for example, having models mixed with the simulator verification and validation more complex and expensive. It also makes it harder to implement the same model on a different platform. Additionally, mixing models and experiments reduces reusability and hinders replication.

The limitations of DEs and ABM to study diffusion processes pose the following question that we are interested in investigating: how can we study diffusion processes in multiplex dynamic networks to overcome the limitations of DEs and ABM? Is there a framework that allows us to do this? To answer these questions, we introduce a method to study diffusion processes in multiplex dynamic networks maintaining separations of concerns in all phases of modeling, implementation, and experimentation. The results include (1) an Architecture to study Diffusion Processes in Multiplex dynamic networks (ADPM) and (2) a systematic Process to define, implement, and simulate diffusion processes over such networks. We use Network Theory formal specifications to define the topology of the diffusion process, ABM to define the behavior of the entities involved, and a formal specification of both for simulation modeling. This research uses the Discrete Event System Specification (DEVS) formalism [16] to define the formal simulation model. Definitions of ABM, DEVS, and Network Theory are found in Section 2.

A major component of ADPM is a formal *Diffusion Abstract Model (DAM)*, automatically constructed from the specifications provided by Network and Agent-Based models. The Network model does not include all the information needed to simulate the diffusion process. Similarly, the Agent-Based model is not a formal model. The DAM is a formal model that combines both in order to model and simulate diffusion processes (i.e., the behavior of the agents representing the diffusion rules and the connections between the agents representing the diffusion network). The formal model helps with early validation prior to implementation, as we can analyze the models' specifications before coding. Likewise, as models, simulator, and experiments are independent, the same model can be implemented on different platforms, which helps to not only reuse existing models and experiments but also replicate the results. ADPM also allows us to simulate diffusion processes with variable topology, the network's characteristics or behavior, without modifying the model (we need to change only the simulation inputs). We can easily simulate different scenarios and network configurations by updating a model experimentation framework without changing the implementation.

The rest of the article discusses how ADPM and our development process can help to build diffusion models. Section 2 discusses related work on diffusion processes and the methodologies we use to define the ADPM architecture. Section 3 presents ADPM and the development process. Section 4 applies ADPM to build a model of a diffusion process in a multiplex network: a model to study the communications in an organization, specifically an emergency plan. Section 5 presents the advantages of the architecture, which are exemplified by the case study. Finally, Section 6 ends with the conclusions of this research.

2 RELATED WORK

Many real-world systems can be described as a combination of components and their interactions. These systems can be modeled as a network where nodes are components and links their relations [3]. The early methods, called *simplex* networks, used a single type of connection between the entities that represented the system [3, 4]. They assumed that all the connections have the same properties, which could lead to misleading results. *Multiplex* networks, in which not all connections are equivalent, are better suited to model diffusion processes. In multiplex networks, links can represent several types of relations. Links are classified into *layers* according to their type: all links representing the same type of relation are organized into a layer. We can use different metrics to study a network's properties. In simplex networks, for example, the network density tells if the network is highly connected; its diameter represents the longest distance between two nodes, the number of connected components, and so forth [17]. Although some metrics are used for Multiplex networks, including centrality [18], this is still an ongoing research topic [4].

Moreover, the generalization of diffusion processes from simplex to multiplex networks is not trivial (this topic is still being investigated [4, 19]; although some types of diffusion processes, such as linear diffusion and random walks, have been generalized, this is an open research area [20]). Such generalization is important because we usually need to model systems with different relations among components. For example, if we want to model a transportation system, we can consider the cities to be the components that are interconnected through different means of transportation (highways, railways, air, etc.). The characteristics of these means need to be modeled accordingly. Additionally, the relations among components may change over time. When relations are permanent, the system can be modeled as a *static network*. However, many systems are *dynamic* (i.e., relations among components change over time). Modeling them using static networks does not capture all a system's properties, which can lead to wrong results. *Dynamic multiplex networks* can better capture their dynamics. As discussed in the Introduction, DEs and ABM have been used to model diffusion processes [5]. There are well-defined diffusion algorithms that employ DEs for studying diffusion processes in networks, such as the Susceptible-Infected-Recovered (SIR) model [21]. Similarly, in medicine, Wang et al. proposed an algorithm to study the diffusion of preventive measures to protect the population against disease [22]. Granell et al. worked on the same topic using Microscopic Markov Chains [23], where individuals (network nodes) in an epidemic only employed three states, and diffusion rules were simple (they did not consider dynamic networks). Other research works also focused on adapting algorithms and models of diseases to study other problems. For example, Khelil et al. applied an epidemiological diffusion model to study diffusion processes in mobile *ad hoc* networks [24]. Others centered on social networks using diffusion models for contagious processes, such as opinion adoption, social movements, or behavior modification [25, 26].

ABM has also been used to study diffusion processes in multiplex networks. ABM can be defined as a "computational method that enables to create, analyze and experiment with models composed of agents that interact within an environment" [27]. An agent is a "computer system situated in some environment that is capable of flexible autonomous action in order to meet its design objectives" [28]. One of the advantages of ABM is establishing correspondence between entities and their interactions in the real system, and the agents and their interactions in models [29]. There is no mathematical representation of agents, and we simulate the models [30], which has its pros and cons: it is easy to observe the system's emergent behavior, but it is easy to introduce errors when translating the model into a computer. Jiang and Jiang matched elements of diffusion processes in social networks to ABM [31]. They found that there are actors that interact in both cases (statically or dynamically). Agents follow a diffusion protocol and they make decisions based on the exchanged elements. They proposed using ABM in diffusion problems for social networks as an alternative method to the theoretical perspective (to obtain empirical results) and to complement theoretical and empirical research. Xiong et al. studied the effect of the diffusion of innovation in social networks using simulation [15]. They defined models for the behavior of nodes but did not include details on integrating the network into a computerized model, the simulation algorithm, or the implementation platform.

DE models are computationally more efficient than ABM and can include a wide range of feedback effects. When we build a network to study a diffusion process using DE, we normally reduce the model's granularity to manage complexity [5]. These DE models are usually nonlinear and every category we add is a new variable in the model that we need to relate to the others. Instead, ABM can model the behavior of each agent (i.e., node) independently, but at a higher computational cost. Additionally, increasing the level of detail increases the cognitive effort needed to understand the model's behavior, which makes sensitivity analyses more complex [5].

Although the previous research in the field shows how to study diffusion processes in multiplex dynamic networks, there are no well-established M&S mechanisms in this area. In most cases, the employed methodology or simulation platforms are not well defined, and applications are built *ad hoc*. As simulators, models and experiments are defined as a mix of software components with no clear separation of concerns, and it is hard to think about the problem, and testing and validation become complex and expensive. This also hinders reusability. As discussed in the Introduction, we want to provide mechanisms to build the simulation systematically by clearly defining the models, their implementation, a platform for experimentation, and an analysis of the simulation results. To do so, we combine DEVS, ABM, and Network Theory.

DEVS is a formal discrete-event M&S methodology [16] that derives from Systems Theory and allows defining hierarchical modular models. A DEVS model is a mathematical entity specified as a black box with a state and an associated duration. Models made of only one component are called atomic models. DEVS models can be connected by linking the outputs of one model to the inputs of other models to form coupled models. There are different DEVS simulators, such as JAMES

[32], JDEVS [33], DEVSJava [34], or Cadmium [35]. We used Cadmium, a cross-platform simulator implemented in C++11 that facilitates the analysis of the simulation results.

In [36, 37], DEVS, ABM, and Network Theory were used to simulate information diffusion processes in multiplex networks. A server-proxy architecture was employed, where servers represent the behavior of the nodes, and the proxies define the diffusion rules for each layer. Servers and proxies were modeled in DEVS, and coupled models represent network nodes. Dynamic DEVS was used to store all the network configurations needed for simulation. This approach was followed to study information diffusion in social networks [38], although different issues can be identified. Sometimes, the rules representing agents' behavior are difficult to handle and do not follow homogenous patterns, which makes storing the retrieval of rules more complex. In [36, 37], the authors do not consider agents where the behavior parameters contain a different number of elements or agents with distinct behavioral parameters. Moreover, the server-proxy architecture does not allow all the needed properties to be represented. For example, individuals in an organization could use different devices (and their networks). If we were interested in studying what would happen when devices or networks fail, we would need to model both. The server-proxy architecture only allows us to include one: either devices or networks. Finally, Dynamic DEVS forces the storage of all the network configurations that we wish to simulate, which is complex to model. Storing all configurations is not feasible. A directional network with three nodes and one layer has eight network configurations. If we increase the number of nodes to 20, we obtain 1,048,576 configurations, a number that increases exponentially with the number of nodes. Using Dynamic DEVS also limits the DEVS simulation tools that can be employed as their availability becomes more limited.

We present an original approach that is general and can be used for any type of diffusion process in multiplex dynamic networks, including a development process and generic implementation. We define a *DAM* that can be defined with other M&S methodologies. The design is flexible, and it allows diffusion processes to be modeled without storing all the network configurations. We use the inputs in the model to change the network configuration during the simulation time. To be able to store complex behaviors, we define the agent's behavior using an XML format that is more flexible than a table format database. Additionally, with the DAM we can include all the properties we need for the model. If we want to simulate the communications in an organization, we can include the properties of both communication devices and communication networks. Other advantages include the possibility of updating the model's properties at runtime and reducing software dependencies.

3 ADPM

The Architecture (and development process) to simulate Diffusion Processes in Multiplex dynamic networks is presented in Figure 1. The architecture is generic and can be used to study several types of diffusion processes (which we discuss later).

ADPM includes the Requirements and Assumptions Document of the problem; a Network model of the relations among components; an Agent-Based model of behavior; the DAM, a formal representation based on the Network and Agent models; a Diffusion Computerized Model (DCM) of the DAM; the Simulation Logs; and a Results Analysis Report. The DAM is defined using a formal specification, DEVS in our case. This solves some of the limitations of Network Theory, such as lack of a formally verified simulator to simulate a diffusion process over the network. By including ABM, we can also define the behavior of both the nodes and links in the network. The formal DAM confers ABM rigor while separating modeling, verification, and experimentation.

The main advantage of combining Network Theory, ABM, and DEVS is that we can use the most appropriate method to model the various aspects of the problem. Network Theory is well suited to

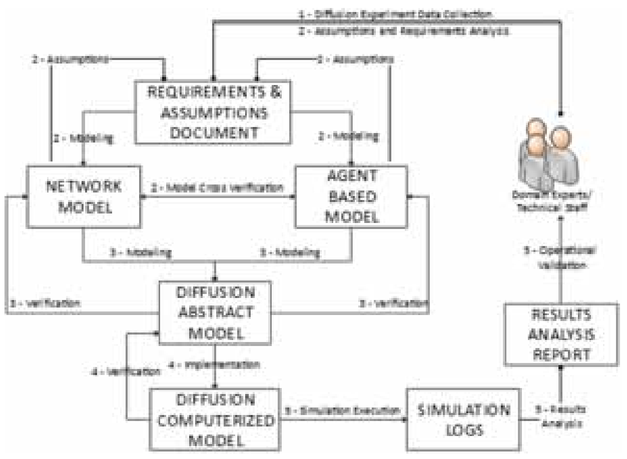


Fig. 1. ADPM organization and workflow.

model the relations between components; ABM is better suited to model behavior. DEVS provides a formal specification to define the whole model as components with hierarchical and modular specifications, which can be executed using the well-established abstract simulation algorithms, which are proven to execute models correctly. This combination allows us to separate concerns and clearly differentiates each part of the problem, as well as separating models from simulation engines and experiments that are independent software entities.

ADPM is combined with a development process, whose different steps are identified with numbers in Figure 1. We now explain the process and the different steps needed.

Step 1 – Diffusion Experiment Data Collection: domain experts, with the help of technical staff, collect all the specifications and details of the system of interest, including a list of assumptions. The output of this step is a *Requirements and Assumptions Document* with all the information that describes the system. It is used in the next step to provide modelers with a detailed description of the system of interest and to define the scenarios that we need to simulate.

Step 2 – Requirement Analysis, Assumptions, Modeling Network, and Agent Models and Model Cross-Verification: these two models are defined in parallel by the modeler (*Modeling* in the figure). Modeling experts review and analyze the *Requirements and Assumptions Document* together with the Domain Experts. This communication is important because requirements can be ambiguous. This step may need several iterations.

Although we can use different perspectives to build these models, we follow [31] and combine the ABM and Network models to build the *DAM*. There is a relation between the components in both the Network Model and ABM. We use this relation for Model Cross-Verification. Although Model Cross-Verification is done manually in this research stage, this process can be automated. The idea is for each node in the *Network Model* to be represented by an agent in *ABM*. Therefore, *ABM* will have as many agents as nodes in the *Network*. The links in the *Network Model* define the possible interactions between agents in *ABM*, and the different types of links in the *Network* match the objects the agents use to interact. Each agent in ABM uses an attribute called "MyLinksTypes" that stores all the types of links that the node handles. Each agent also employs an attribute called "MyRelations" that stores all the possible output connections of the node with the link type (this is detailed in Section 3.3).

Any new assumptions made while defining the *Network* and *ABM* models must be approved by the domain experts and must be included in the original document. Modelers must include the new assumptions in the document, and the domain experts must analyze and approve or reject them. This step can result in revisiting step 1. These models must be validated. In systems where operational validation is not possible, they should be validated by domain experts.

Step 3 – Modeling the DAM and DAM Verification: modelers are provided with the *Requirements and Assumptions Document* and the *Network* and *Agent-Based* models. If there is a mismatch or missing data or further assumptions are needed, they revisit step 2. Modelers decide which generic DAM components should be included (see Section 3.4) and how to instantiate them. Therefore, the generic DAM can be considered a metamodel that is instantiated for specific applications using the information in the *Network* and *Agent-Based Models* (as detailed in Section 3.4). In our case, we used DEVS to formally define the DAM.

Step 4 – Implementation of the DCM and Verification: this step consists in converting the formal *DAM* into an executable model: the DCM. In our case, we used DEVS and the Cadmium toolkit, but the architecture and development process can be followed, and other formalisms or tools can be used. Developers must verify the simulation that derives from the model in step 3; if problems arise, steps 3 and 4 must be revisited. The *DCM* can be customized to simulate the different scenarios proposed in the *Requirements and Assumptions Document.*

Step 5 – Simulation Execution, Results Analysis, and Operational Validation: using the DCM and the simulation scenarios defined in the *Requirements and Assumptions Document,* we execute different simulations. We analyze simulation logs to obtain meaningful information, which is summarized in the *Result Analysis Report.* This report is given to the domain experts for validation (expert validation) and decision making. New iterations may be needed before the results are valid.

In the rest of the section, we discuss all these components in detail.

3.1 Requirements and Assumptions Document

This component includes all the requirements, specifications, assumptions, and data available for the problem under study. Although every case is different, a minimum set of requirements would include the following:

- The elements to be diffused (i.e., rumors, viruses; more than one type can be included)
- Individuals diffusing the above-mentioned elements: e.g., if we are studying the diffusion of a virus, it could be transmitted by both animals and persons
- Individuals' behavior: behavior rules depend on different variables, such as intrinsic characteristics, the relation types they have, or the diffusion element type
- Starting diffusion elements (diffusion can also start in distinct locations over time)
- Effects of diffusion elements and which are relevant
- Mechanisms to spread diffusion elements, including characteristics and connections
- The diffusion mechanisms that can be used by everyone
- The variables we are interested in studying: e.g., we may be interested in the number of individuals infected by the virus, how it is transmitted, or specific infected population groups
- Scenarios to be analyzed: we may study what happens when individuals are vaccinated, the results of a prevention campaign, among others; scenarios should include all their characteristics and parameters (i.e., effectiveness of the vaccine or the number of individuals to vaccinate)

These requirements can be collected manually (e.g., through interviews) or automatically (e.g., with different sensors). If information is incomplete, experts should provide additional information or a set of valid assumptions.

Table 1. Nodes Ids and Labels (Partial)

Id	Label
1	Node name 1
2	Node name 2
3	Node name 3
4	Node name 4

Table 2. Network Connections (Partial)

Source	Target	Label
16	323	Layer 1
16	324	Layer 1
325	368	Layer 2
325	369	Layer 3
1	2	Layer 2

```
<?xml version="1.0" ?>
<AgentBehavior>
  <Id>MyId</Id>
  <MyLinksTypes>
           <Link Id=Link1/>   ...
     <Link Id=Linkp/>
  </MyLinksTypes >
 <MyRelations id=AgentId1>
       <Link Id=Link2/>
      <Link Id=Link5/>
  </MyRelations >       ...
  <MyRelations id= AgentIdt/>
       <Link Id=Link2/>
  </MyRelations>          ...
  <BehaviorRules>
     <Rule Id = RuleId1 Parameter11 = Parameter11Value ... Parameter1n = Parameter1nValue >
        ...
     <Rule Id = RuleIdt Parameter1t = Parameter1tValue ... Parametern = ParametermtValue >
  </BehaviorRules>
  </MessageBehavior>
</AgentBehavior>
```

Fig. 2. Example of the agent's definition using XML.

3.2 Network Model

The Network model is an organized representation of the Diffusion experiment data collected in the *Requirements and Assumptions Documents*. It formalizes the relations among the system components using Network Theory. The resulting network can be implemented, analyzed, and visualized by different tools (Gephi [39], Pajek [40], MuxViz [41], etc.) and stored in various formats (network graph—i.e., a graphical representation of the network, XML, tables, etc.).

In our case, we organized the information into two tables so they could be easily translated into other formats (CSV, XML) and then imported to software tools to analyze and visualize the network. One table stores the nodes of the network. Each node has a unique id and a label (Table 1). The second stores the connections between nodes (Table 2). Each row represents a connection defined by the Source node id, the Target node id, and a label representing the type of link. As we can see, the behavior of nodes and the characteristics of links are missing in the network model.

3.3 Agent-Based Model

The Agent-Based model is a representation of the behavior of those in charge of the diffusion process, the objects they use for diffusing the element, and the properties of the relations among these objects. It is formalized using ABM and can be implemented by different methods: DEVS, XML, specific software platforms like NetLogo or Repast [42]. Figure 2 illustrates an XML implementation of a generic agent with a minimum set of attributes (represented as XML tags) that must be included to capture all the information needed to study the diffusion process, that is, the

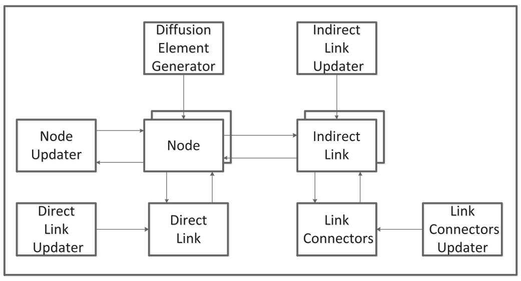


Fig. 3. The DAM structure [43].

connections from the Network Model and the behavior defined for each node and each link in the *Requirements and Assumptions Document.*

The behavior of the agent is defined between tags <*AgentBehavior*>. We have the following attributes at least (which match the network model as discussed in the following):

- <*Id*> of the agent: it has the same *Id* as the node whose behavior we are defining (i.e., one of the Ids in Table 1).
- <*MyLinksTypes*>: it represents the types of input and output *Links* that the agent uses. We filter all the rows in Table 2 where the agent Id is present (as a source/target). We then employ this attribute and store all the link types from the filtered table (i.e., the column label in Table 2). If there are two rows with the same label, we simply add it once.
- <*MyRelations*>: it represents the outputs of the agent (i.e., node). It is an attribute with two elements: (1) an Id to represent which agent we can contact and (2) as many *Link*s as layers we can use to contact the agent. We filter all the rows in Table 2 where the id is a source. We add one <*MyRelations*> per Id in the filtered table. The value in the target column is stored in the <*Id*> attribute. We create one <*Link*> attribute per label.
- <*BehaviorRules*>*:* it represents the behavior rules of the agent. This information is not available in the Network Model. Each element inside <*BehaviorRules*> represents a <*Rule*> for the agent during the diffusion process. The parameters of this attribute must be extracted from the *Requirements and Assumptions Document.*

<*MyLinksTypes*> and <*MyRelations*> capture the multiplex part of the network in the agent. The attribute names can be modified to make the behavior readable in each context; for example, in an information diffusion process, <*MyLinksTypes*> could instead be <*Devices*> or <*CommunicationMechanism*>.

3.4 Diffusion Abstract Model (DAM)

The DAM is an abstract and formal representation of the *Diffusion experiment data* from the *Requirements and Assumptions Document* that matches the elements in the *Network* and the *Agent-Based Model.* It is formalized using a formal specification (DEVS in our case, but other specifications like System Dynamics or State Charts could be used). It is also possible to utilize different methods for the different components if there is a way to connect them (i.e., a metamodel). The *DAM* is a generic container that follows the structure shown in Figure 3.

Appendix A shows the formal DAM definition using DEVS [43]. To build the DAM, we need a model library for the components (a DEVS library in our case) or an expert in the formalism used (an expert in DEVS in our case). For the rest of the section, we detail all the components and how they are obtained with the information in the ABM and Network models.

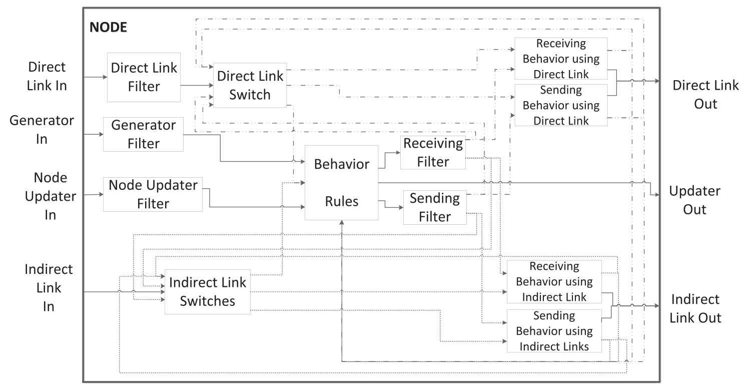


Fig. 4. The Node element structure.

3.4.1 Node Element. The *Node* element (Figure 4) provides a structure for defining the agents from the XML Agent model (from this point onward the *XML Agent* refers to the definition in Figure 2). Each *Node* element corresponds to a node in the Network Model (Table 1) and the corresponding *XML Agent* in the ABM. The DAM will have as many *Node* elements as nodes in the Network model (and *XML Agent* in ABM). The *Node* element is a generic container that follows the structure presented in Figure 4. It uses several submodels: input *Filters and Switches* (*Direct* and *Indirect Link* connections), *Behavior Rules*, and *Behavior* models (for *Sending/Receiving*). The punctuated lines represent all the connections related to the behavior of the agents using indirect links. The dashed lines represent all the connections related to the behavior of the agents using direct links. The straight lines are used by default. This notation is maintained throughout the present work.

Behavior Rules implement the diffusion rules and network connections defined earlier in the ABM (tags <*BehaviourRules*> and <*MyRelations*>). This is a parameterized model in a model library (otherwise, it must be developed for the specific application). It will be instantiated for each node using the information in the *XML Agent*. If all the same parameters are employed, we have a single model. If different *XML Agents* behave differently, we may have distinct models and we will choose one or the other based on the behavior parameters.

As we broadcast messages in the model, we need the *Direct Link, Generator,* and *Node Updater* filters to check if the received messages are for this *Node*. These models do not have a representation on the Network and ABM models, and deal with broadcasted messages.

Direct Links Switch also classifies incoming messages. It has the same number of *Indirect Link Switches* as the *Node's* number of diffusion elements. It is instantiated based on <*MyLinksTypes*>.

Receiving and *Sending Filters* is used to classify the instructions in those with incoming or outgoing instructions.

Receiving Behavior using Direct Link represents the assimilation of diffusion elements. It defines the behavior of the *XML Agent* when the diffusion element is received via a *Direct Link*. This model can be available in a library or must be developed for the specific application. We may have different models if the nodes have different behaviors. Similarly, *Sending Behavior using Direct Link* is used to spread the diffusion element. *Receiving Behavior using Indirect Links* represents the assimilation of the diffusion element by *Indirect Links*. Likewise, *Sending Behavior using Indirect Links* represents the process to spread the diffusion element.

Appendix A shows the formal definition of some of these coupled models, while the formal definition of atomic components, such as *Generator Filter,* can be found in Appendix B.

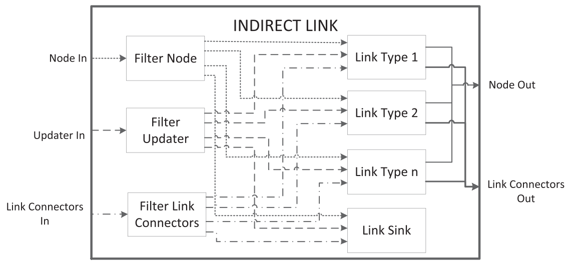


Fig. 5. *Indirect Link* coupled model.

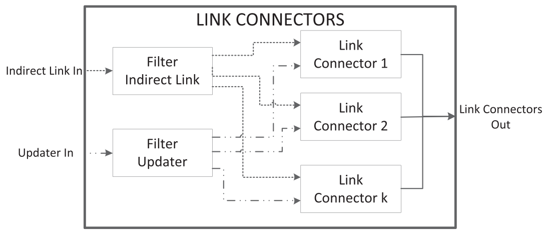


Fig. 6. *Link Connector*s coupled model.

By defining the models formally as in Appendix A, we can perform early validation by analyzing the model definition, which saves time in early phases of the project. For example, in *Generator Filter,* the variable md^b accumulates the number of messages passing through the filter. By analyzing the formal model, we can verify if the model is defined according to the specifications. For example, we can check if md^b is cleared after sending the messages. We can also check if the model passivates every time md^b is empty. Likewise, in the DAM coupled model, we could check if the connections between components are well defined. No software implementation is required for such checks.

3.4.2 Indirect Link. It is presented in Figure 5 and provides a generic structure for communication. It represents the types of links that the *XML Agent* can access (i.e., the input and output links in the Network Model) identified by *<MyLinksTypes>*. They define the objects that the agent uses for the diffusion process. There are as many *Indirect Link* elements as there are *Node* models. It is instantiated using the type of indirect links. When implementing the model, we can define a DEVS coupled model automatically.

Link Type i represents how we transmit the diffusion element. For instance, in a transportation system, *Link Type 1* could be an Airport, *Link Type 2* a bus station, and so forth. For spreading news in Social Networks, *Link Type 1* could be Facebook, *Link Type 2* Twitter, and so forth.

Filters redirect the diffusion elements (news, individuals, etc.) and the *Link Sink* collects all messages without a matching *Link Type.* The behavior of the Link Types should be in a model library or developed by an expert (in our case, a DEVS expert). The *Sink* can be reused for different applications.

3.4.3 Link Connectors. Figure 6 shows how the objects used by the *XML Agents* to diffuse the diffusion elements are connected. For example, in a transportation system, *Link Connector 1* may represent roads, *2* railways, and so forth. For Social Networks, we could have a single *Link Connector,* such as the Internet (in which case *Filters* are not used). As in the previous case, *Filters* redirect the diffusion elements or the updates in the properties of each *Link Connector* to the appropriate model. The models for the behavior of the different Link Connectors should be in a model library or developed by an expert (a DEVS expert in our case).

3.4.4 Direct Link. It represents a direct connection between nodes, that is, the properties of the links in the Network model with a direct connection between nodes. In the Agent-Based model, it represents the connections handled without using any additional objects (it would not be used in a transportation system as cities are always connected by roads/rails/flights; in Social Networks, it could represent the direct connection between individuals in the same location, who can talk face to face). This model should be adapted from a library or developed by an expert (a DEVS expert in our case).

3.4.5 Diffusion Element Generator. It is an atomic model that generates the elements to be diffused over time. It defines the initial location of the diffusion elements in the Network and Agent-Based models and the new ones introduced over time. Different scenarios can be simulated by simply updating a file (changing the implementation is not necessary). In a transportation system it would, for instance, generate passengers and their destination; in a Social Networks study, it would generate rumors. This is an artifact to start the simulation and introduce new diffusion elements while the simulation is running.

3.4.6 Updaters. They modify the properties of the model at runtime by generating updates for the models that they are connected to (simulating different scenarios without changing the implementation). They do not correspond to the Network or ABM models. In a transportation system, the updaters might generate road closures or train stations may close. In a Social Network like Facebook, they might generate configuration changes (e.g., public sharing instead of friends only). The *Node Updater* generates updates in the properties of the nodes based on external information. In a transportation system, it could generate new links between cities (i.e., a new flight), change a train timetable, and so forth. In a Social Network, it could generate updates in face-to-face connections or generate new behaviors for individuals based on external parameters.

3.5 Diffusion Computerized Model (DCM)

The DCM is an implementation of the *DAM*. We utilize the formal definition of Section 3.4 and Cadmium (Appendix C shows an example of the *Generator Filter* defined in Appendix B). Although we employ the Cadmium tool, other DEVS Simulators can be used. Having implemented all the components of the *DAM*, we build a DAM top model using the Network and Agent-Based models in Figure 2. This automation allows us to study different network configurations with the same implementation. Appendix D shows a general implementation of the DAM.

3.6 Simulation Logs and Results Analysis Report

The DCM provides Simulation Logs for the different scenarios we execute. These logs are processed using different statistical and data visualization tools (e.g., R [44], PowerBI [45], etc.) to generate reports used for both validation and decision making (Step 5). We need to conduct expert validations and other kinds of operational validation, including animation, comparison to equivalent models, historic data validation, and so forth [46]. Based on the results in this step, we can define modifications in the system specifications and define new scenarios, modify the *Requirements and Assumptions Document,* and restart the process.

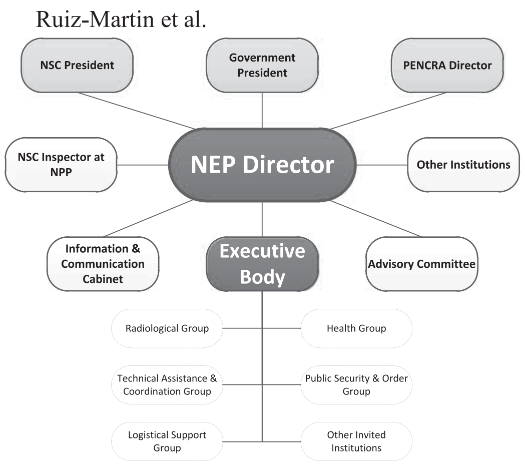


Fig. 7. Sketch of the organizational structure of the NEP.

4 COMMUNICATIONS IN A NUCLEAR EMERGENCY PLAN

To show how to use ADPM, we present a case study of a diffusion problem in a multiplex network: the *communications* used in a Nuclear Emergency Plan (NEP). The section is organized and described based on the ADPM components and process explained in Section 3.

4.1 Requirements and Assumptions Document

We used actual information obtained from the emergency plans for a Spanish Nuclear Power Plant (NPP). The plan, defined by the Spanish Civil Protection Agency, states in detail the actions needed in the event of an accident (NB: detailed information about the NPP and the plan cannot be revealed as this information is confidential; we only discuss important aspects for this research that can be shared with the general public). The NEP is a detailed management plan that defines the structure and functions of a Virtual Organization composed of different public organizations (police, town halls, etc.) and coordinated to solve the emergency. It also defines the tasks to be performed by every suborganization and how they are related. Data collection was done together with NEP experts. We analyzed the documentation, acquired the needed data, and conducted follow-up interviews by the procedures discussed in Section 3. The *Requirements and Assumptions Document* contains a comprehensive definition of the NEP [47]. We discuss the most relevant aspects of this document containing 96 pages to show how to define the Network, Agent and Abstract Diffusion models.

Figure 7 shows the organizational structure of the NEP. At the core is the NEP Director, who makes decisions to manage and solve the emergency. As the emergency evolves, higher National Government ranks (such as the President) can replace the Director. Different systems are used for communications, including landline/cell phones, fax, mixed radio/phone networks, satellite, Internet, Remer and Reman radio channels, and *in situ* communication. The Health Group can also employ beepers. Emergency managers and first responders can also use military communications, which are not available until the NEP Director makes a request and the infrastructure is deployed (we do not consider this in our model; this is a backup plan should everything else fail and is beyond the scope of our study). Communications must follow the NEP hierarchy and the internal structure of each group. The most frequently used communication system is the *phone*: landline or mobile.

Table 3. Summary of the Commands to Be Managed in the Event of an Emergency

Level 0
Notify and verify the incident at the NPP
Start Emergency Level 0
Request data about the state of the emergency
Level 1
Evaluate the available data to determine the emergency category
Start Emergency Level 1 and activate every group in the NEP
Track communications in the NEP
Track the evolution of the emergency at the NPP
Ask first responders to show their accreditation and classify them into working groups
Ask substitute teams to start working on the emergency. Tell those ones working to rest
Level 2
Start Emergency Level 2
Integrate the extraordinary resources needed in the emergency
Verify the safety and security in the emergency area (e.g., protect against looting)
Radiological Prophylaxis (tell individuals to take medication to protect themselves against radiation)
Level 3
Classify individuals based on their exposure to radiation and decontaminate them
Evaluate the state of the infrastructure and any other resources and decontaminate them
Classify animals based on their exposure to radiation and decontaminate them

The NEP defines every command to be made during an emergency (more than 30). The Director selects what to do based on the evolution of the emergency. Table 3 shows various commands classified according to the level of the emergency (0 to 3). At each level of the emergency plan, all the commands available at lower levels can be used.

As we show in Section 4.3, we used these rules to define the attributes of the agents' behavior. Presenting the rules as a table facilitates this transition. This information is not needed to develop the Network model. The messages transmitted in the Network and Agent models (i.e., diffusion elements) are the commands shown in Table 3 and their acknowledgments.

4.2 Network Model Definition

To build the network model, we utilized the information from the *Requirements and Assumptions Document*: the individuals involved in the NEP and the systems they can use to communicate with one another. The messages transmitted inside the network (i.e., diffusion elements) are the commands and acknowledgments found in the document. In the network model, nodes represent the individuals involved in the NEP, and the links denote the relations between individuals. Each type of link represents a communication system. There are 832 nodes and 12 types of links (fax, Internet, landline phone, mobile, satellite, Reman radio channel, Remer radio channel, Civil Guard radio-phone, Radiological Group radio-phone, Autonomous Police radio-phone, *in situ* communications, Beepers).

We analyzed the network properties using Gephi [48]. Figure 8 shows one of the multiple representations of the network at the beginning of the emergency. We can see the groups defined in the *Requirements and Assumptions Document.* We employed a simplex network (Gephi does not support multiplex networks) and we identified that there was no resilience [49] in the communications in the Health and Logistical Support groups. We found that some relations between

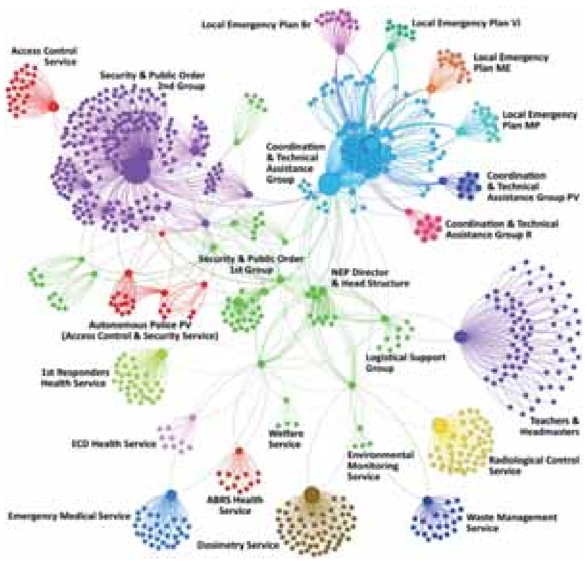

Local Emergency Plan Br
Local Emergency Plan VI
Access Control Service
Security & Public Order 2nd Group
Local Emergency Plan ME
Local Emergency Plan MP
Coordination & Technical Assistance Group
Coordination & Technical Assistance Group PV
Coordination & Technical Assistance Group R
Security & Public Order 1st Group
NEP Director & Head Structure
Autonomous Police PV (Access Control & Security Service)
Logistical Support Group
1st Responders Health Service
Teachers & Headmasters
ECD Health Service
Welfare Service
Environmental Monitoring Service
Radiological Control Service
ABRS Health Service
Emergency Medical Service
Waste Management Service
Dosimetry Service

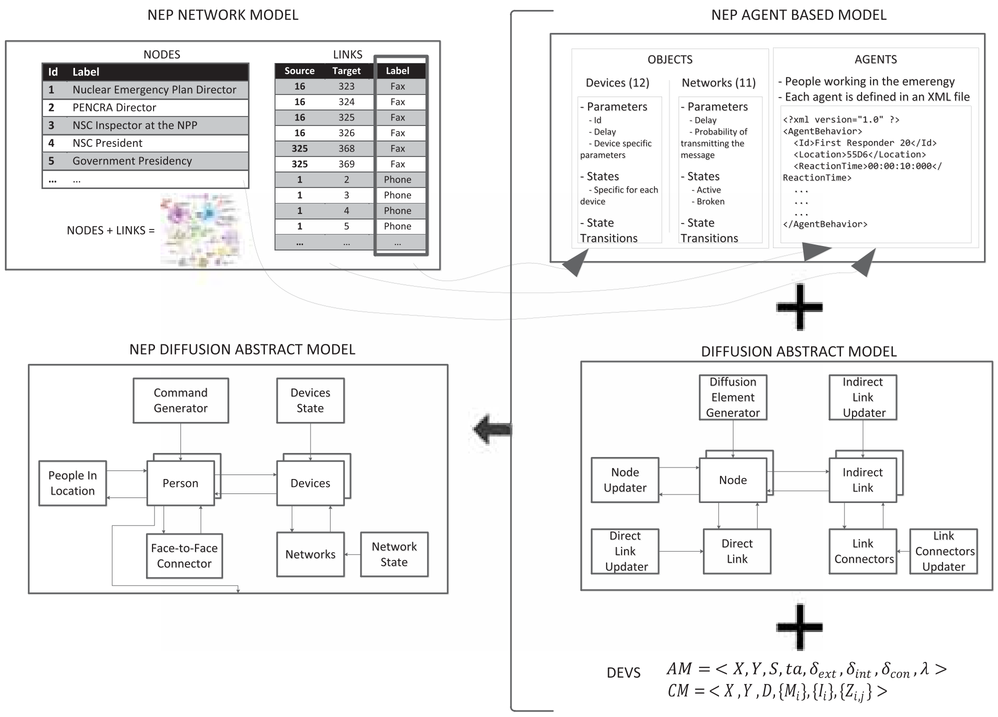
Fig. 8. NEP network at the start of the emergency [48].

individuals were managed only by a single communication system. Nonresilient communications imply that individuals are isolated from the group if the communication system fails. They cannot receive commands to deal with the emergency, which makes things worse. Although this analysis does not allow the behavior of individuals to be included, we used this model to study how failure in different communication systems affects network connectivity before analyzing the system's dynamics.

4.3 Agent-Based Model

We defined the behavior of the individuals (i.e., the rules they follow when receiving and transmitting information) using ABM. We considered the relevant characteristics of the behavior of individuals. We then used the organizational structure, the communication systems, and the communication rules defined in the *Requirements and Assumptions Document* to complete the model. We employed XML to describe the behavior of agents as in Figure 2:

- **Id:** identifies the agent (e.g., NEP Director, Radiological Group Chair, or others).
- **Location:** represents the location of the agent. When the emergency starts, their predefined actions determine their initial location.
- **Reaction Time:** indicates how long it takes to react to a stimulus.
- **Answer Priority Type:** identifies the priority of the agent to receive a command. It can be based on who sends the command or the device receiving the message or could be random.
- **Send Priority Type:** identifies how the agent chooses the commands they send. Their priority can be based on a priority list or on arrival time or could be random.
- **My Devices:** identifies the devices for each agent in the Requirements Document. For each device, we define the relative priority of the device for the agent (parameter), its type (attribute), if it can broadcast/multicast (attribute), and if it is half/full-duplex (attribute).
- **Prioritized Task:** indicates how the agent sorts the tasks to be performed during an emergency.
- **Answer Device Priority:** agent's priority to respond to commands based on the devices.



Id	Label
1	Nuclear Emergency Plan Director
2	PENCRA Director
3	NSC Inspector at the NPP
4	NSC President
5	Government Presidency
...	...

Source	Target	Label
16	323	Fax
16	324	Fax
16	325	Fax
16	326	Fax
325	368	Fax
325	369	Fax
1	2	Phone
1	3	Phone
1	4	Phone
1	5	Phone
...	...	...

```
<?xml version="1.0" ?>
<AgentBehavior>
  <Id>First Responder 20</Id>
  <Location>55D6</Location>
  <ReactionTime>00:00:10:000</
ReactionTime>
  ...
  ...
  ...
</AgentBehavior>
```

$$AM = < X, Y, S, ta, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda >$$
$$CM = < X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$$

Fig. 9. Scheme of the DAM definition for the NEP.

- **Answer Person Priority:** agent's priority to receive commands based on who sends them.
- **Send Command Priority:** each command on the list includes priority, destination, and content. In some cases, it is in the *Requirements and Assumptions Document* (e.g., the Director "Establishes emergency level 0"); in others, it is not specified.
- **Action Execution Priority:** indicates how the agent prioritizes the actions.
- **Communication Relations:** it is a list of the connections of the agent. Each connection has two attributes: target and device. It is translated directly from the Network model. In Table 2, we select the rows with *Source* equaling the Id in the ABM. Each row is a *Communication Relation* where the *target* attribute is the same as in the table, and *device* is the *label* attribute in the table.
- **Message Behavior:** defines the messages that the agent must send based on the received messages (defined in the *Requirements Document*). For each command or acknowledgment, there is a list of messages to be sent and a list of actions. Each message includes (1) destination, (2) content (e.g., "Tell individuals to stay home"), (3) a *mandatory/optional* tag, (4) a *broadcast/multicast* tag, (5) a mandatory device to use, and (6) if acknowledgment is needed.
- **Action Behavior:** defines the set of actions of the agent (*Requirements Document*). Each action includes (1) average execution time, (2) location, and (3) messages to send (with the same attributes in *Message Behavior*). This is an acknowledgment of action being completed.

4.4 Diffusion Abstract Model for the Nuclear Emergency Plan (NEP)

Once the Agent-Based model is completed, we define the *DAM* introduced in Section 3, and we apply it to the NEP. This process is summarized in Figure 9.

We use the NEP Network and the NEP Agent-Based models (top of the figure) to decide which DAM components are needed for the case study. We also choose the formal methodology to define the DAM. We then need to instantiate the generic DAM defined by ADPM (bottom right of the figure) for the specific application (bottom left of the figure). This is the NEP in our case. We will discuss how to convert the generic components into the proper instantiations.

The mapping between the components in the generic DAM and the names in the specific application is not automated. *Node* in the DAM corresponds to *Person* in the NEP (this information is obtained from the Agent and Network models). *Indirect Links* are converted into *Devices* (representing the devices that each person can use, obtained from the Agent model and from the labels in the Network model). *Links Connectors* are converted into NEP *Networks* (i.e., the Internet, the telephone network). In this case, such information is also defined as an object in the Agent-Based model. *Direct Link* is converted into a *Face-to-Face Connector* that represents *in situ* communications.

Nodes Updater is converted into *People in Location* models. For each location, this model calculates the list of persons present by taking the current location of each Person as input.

Indirect Links Updater is transformed into *Device States*, which define if a device is broken. We do the same for *Link Connector Updater—Network State*. Finally, *Diffusion Element Generator* is converted into *Command Generator*, which generates commands according to the *Requirements Document*. This triggers the diffusion process. We do not use *Direct Link Updater* as we do not want to modify the attributes of the *Face-to-Face Connector* at runtime (i.e., we could change environmental noise *in situ*, distance, etc., but such scenarios are beyond the scope of this study).

The next step is to formalize each component using the general DEVS specifications in Section 4. The DEVS parameterized atomic models are different for different applications; they must be available in a model library or be defined by a DEVS expert using the parameters in the ABM and Network Models. Once the atomic components are defined, the instantiation process of the *NEP DAM* is automated (using the Agent model and the parameters for devices and networks). As in our case there are hundreds of components, we automated this process by taking agents' XML files as inputs and generating a DEVS Computer Model (see Section 5.5).

In the rest of the section, we explain how to instantiate these DEVS coupled and atomic models.

Person-coupled model instantiation: we use the *Node* structure presented in Section 4.2. Our agents employ both direct (i.e., face-to-face) and indirect (i.e., devices) communications (Figure 10).

Behavior Rules is a coupled model that includes the actions that the agent performs to solve the emergency. It is instantiated for each *Person* model using the agent information in the XML file. It models how the person reacts to messages and how they execute tasks (based on the behavior defined in the XML, the active devices, the individuals around, and a to-do list).

Devices coupled model instantiation: we follow the *Indirect Link* structure in Section 4.3. We use the *MyDevices* attribute of the XML file and *devices* models employing the agent *Id*. The *Devices* coupled model includes *filters*, a *sink*, and as many *devices* as elements in *MyDevice* (Section 4.3).

Networks coupled model instantiation: we use the *Link Connectors* structure in Section 4.4. The model employs as many *Link Connectors* as networks in the Agent model. Each *Link Connector* is instantiated by a network parameterized model.

Other models: the other components in the NEP DAM (*Face-to-face Connector, Network State, Device State, Individuals in Same Location,* and *Command Generator*) are defined as atomic models. Their instantiation is like the atomic components of the previous examples.

4.5 Nuclear Emergency Plan (NEP) Diffusion Computer Model (DCM)

As we explained in Section 3, the *NEP DCM* is a Computer Model of the DAM. We built it using the Cadmium simulator introduced in Section 2. The model is based on the *NEP DAM* (i.e., the atomic

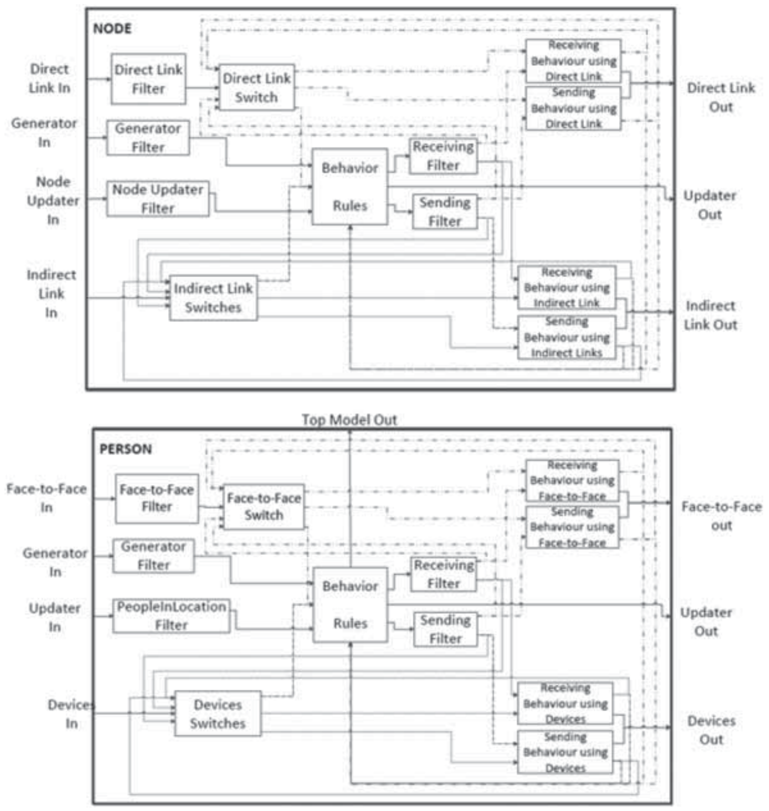


Fig. 10. Coupled model definition of a Node and its translation into a Person model for the NEP.

and coupled models we defined) and the agent XML, which are used to translate the *NEP DAM* into a Computer Model for Cadmium. Figure 11 shows a diagram of this process.

To implement the coupled models, we first instantiate the atomic models using functions that query the XML file and/or the parameters for the devices defined in the Agent model. The function uses the rules to generate all the required code. The top-level model is built by taking the XML files where agents are defined, reading them and transforming them into a structure to generate the parameters of all the previously explained functions. The output is a file with thousands of lines of code for Cadmium, including all the instantiated atomic and coupled models, which, once compiled, generates the *NEP DCM* ready to generate results. This process is automated.

This automation process is independent of the DEVS simulator that we use if the simulator allows the implementation of the parameterized DEVS atomic models that can be instantiated when the coupled model is defined.

4.6 Simulation Results

In this section, we show how to use our *Architecture for Diffusion Processes in Multiplex dynamic networks* and the model defined above. We employ two different case studies: the NEP leadership and the Health group to exemplify how we can respond to different questions suggested by decision makers.

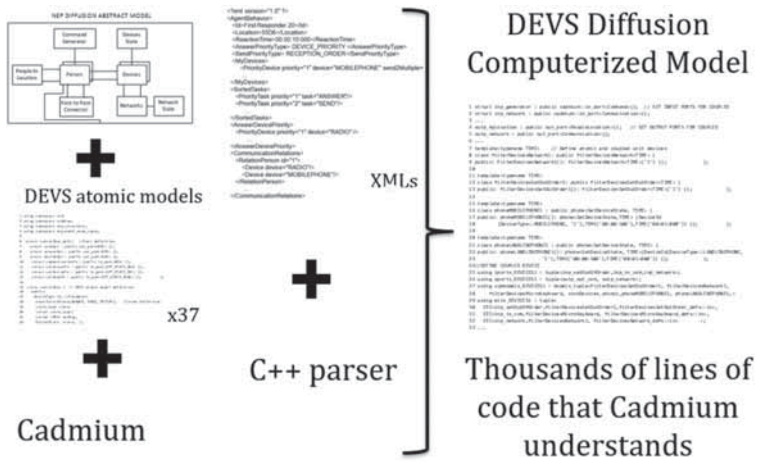

NEP DIFFUSION ABSTRACT MODEL
DEVS atomic models
x37
Cadmium
XMLs
C++ parser
DEVS Diffusion Computerized Model
Thousands of lines of code that Cadmium understands

Fig. 11. Scheme of the NEP DCM implementation.

```
00:00:00:000 “Establish Emergency Level 0” – “NEP Director”
00:25:00:000 “Establish Emergency Level 1” - “NEP Director”
00:50:00:000 “Establish Emergency Level 2” - “NEP Director”
02:00:00:000 “Establish Emergency Level 3” - “NEP Director”
```

Fig. 12. Commands issued by the Director.

4.6.1 Analysis of Nuclear Emergency Plan (NEP) Leadership. This case study presents the basic functionality of ADPM and shows some simple simulation results by focusing on employing ADPM. Leadership consists of 13 individuals and their respective communication devices.

The scenario we discuss is as follows: the NEP Director issues four commands at four separate times (with various levels of severity; see Figure 12). In addition, the communication device used by person 13 in the Radiological Group (a Leadership member) fails at time 00:15:00:000 (i.e., 15 minutes after the emergency started).

The simulation output log shows the tasks performed by each agent, including the communication mechanisms used. We transformed the log file in a summary report (*Results Analysis Report* in Fig. 1) using PowerBI [45], a tool for big data analyses.

We use this scenario to illustrate how to answer the following questions:

- What are the most widely used devices? To do so, we study the usage of the various communications devices.
- Are the message transmission rules properly defined in the NEP? To do this, we count the individuals who receive the messages when both networks and devices work as expected, and we compare them with the individuals who are assumed to receive messages (according to the NEP specifications).
- Who are the busiest individuals? We identify how many tasks are performed by each person and how many messages they receive.

a) What are the most widely used devices?

From the simulation log, we check the device that each person uses for each communication, and we count the number of times that each one was used. Table 4 summarizes in this scenario that the fax is used 36 times, followed by face-to-face communications and the Radiological Group

Table 4. Simulation Results: Use of Devices

Communication Device	#Uses
FAX	36
IN_PERSON	12
RADIOLOGICAL_GROUP_DEVICE (RGD)	10

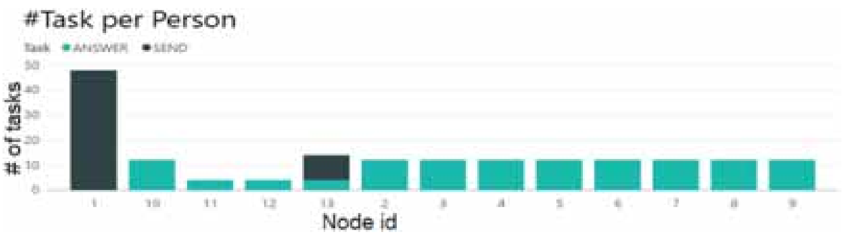


Fig. 13. Number of tasks per person, classified by sent and answered tasks.

Device (RGD). The emergency plan specifies that some individuals can use email, cell phones, and so forth. However, it is hard to predict which will be the most employed as both users' preferences and the availability of the devices will affect the result. Thanks to the simulation model, we can observe the emergent behavior of using devices, which may differ from the desired one.

By identifying the most widely used devices, we can find the networks that are most critical in the event of disruptions. Knowing the most widely used devices allows us to simulate scenarios in which they fail and to decide if they are critical or if there are other ways to transmit commands. The results in this specific scenario suggest that we need to focus on RGD and Fax communications.

b) Are the message transmission rules properly defined in the NEP?

To analyze if the message transmission rules are properly defined in the NEP, we study the number of individuals who receive each command. If all the devices and networks work as expected, knowing how many individuals receive the command allows us to identify if the message transmission rules are well defined. To do this study, we need to identify how many individuals were expected to receive the command, and how many received it in our simulation scenario. In this specific case, 12 individuals received each command. As the Leadership includes 13 individuals and the Director is the person who generates all the commands, we can see that everyone received all the commands. Moreover, the failure of the device used by person 13 did not affect transmission. We can conclude that the rules to transmit commands are well defined.

c) Who are the busiest individuals?

The simulation results also allow us to identify the number of tasks performed by each person and to therefore identify the individuals with more workload. Figure 13 shows the tasks by each person classified as two types: ANSWER (i.e., responding to incoming communications from another person) and SEND (i.e., transmitting a command). In our case, *Person 1* (the NEP Director) is the most active and only sends commands. The others (except person 13) only receive commands. This shows that it can be a promising idea to include a new person at the head of the NEP to help the Director to transmit commands to other individuals.

4.6.2 *Analysis of the Health Group inside the NEP.* We extended the NEP leadership to include the Health Group to show that we can simulate different scenarios by merely updating the Agents XML to automatically generate a new DCM. We instantiated the NEP DAM with Leadership and

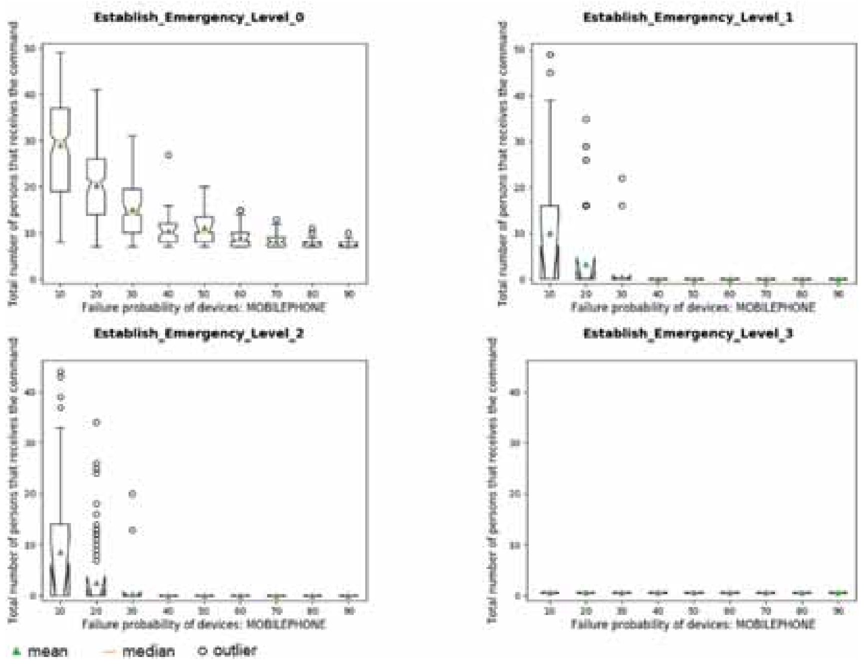


Fig. 14. Individuals who receive command "Establish Level 0, 1, 2, and 3" in the Health Group.

the Health group (107 individuals and their devices). We can see that both the number and behavior of agents differ from the NEP leadership, as well as the connections in the network model.

In this case study, we focus on finding out which communications mechanisms are critical. To answer this question, we analyze the individuals receiving commands based on different failure rates for devices.

a) Is the mobile phone a critical communication mechanism in the Health Group?

We studied what happens when the Director issues different emergency levels. Mobile phones fail with different probabilities. The simulation was repeated until a 95% confidence interval was obtained. Figure 14 shows the number of individuals who receive commands "Establish Emergency Level 0, 1, 2, and 3" when mobile phones fail with different probabilities (i.e., failure probability from 10% to 90%). To analyze the results, we consider that the NEP establishes that 49 individuals in the Health Group should receive commands for these levels. The group can use beepers, mobiles, and landline phones. However, it does not clarify who can use beepers, and first responders can only use mobile phones. We made a conservative assumption, and no one used a beeper.

We see that around eight individuals always receive command "*Establish Emergency Level 0,*" regardless of the failure probability. By analyzing the simulation log files, we found out that they belong to Leadership. Although they always receive commands, even 10% of failure probability prevents the message from being sent. We also see that for a 10% failure probability, the mean and median are around 30 (and 49 individuals should receive the command). Both the mean and median decrease as failure probability increases. When failure probability is over 50%, the results are uniform, and both the mean and median are fewer than 10 individuals.

We can see that when failure probability is over 30%, nobody receives the other commands. As individuals send the messages in FIFO order, and they are not limited to a number of attempts to transmit the command, the information transmission process blocks if a device with only one communication mechanism fails.

This study indicates that the mobile phone communication mechanism is critical, and we cannot afford a failure rate of even 10% in the Health group because, in that case, more than 75% of individuals do not receive the command. We can also conclude that the behavior we studied for individuals is not efficient because the information transmission process is easily blocked.

This study shows the importance of reviewing the communications defined in the NEP in the Health Group. We also found that the number of attempts to send a command before assuming lost communication is critical. Future analyses need to consider how this attribute affects the diffusion process. One advantage of ADPM is that we can extend the model by incrementally adding new features. In this case, we would need to update the XML files where the behavior of agents is defined and by only updating one DAM component: Behavior Rules (Figure 10).

b) Expert Validation

Although we usually conduct operational validation and compare the results with real data, this was not feasible in this case study because there are no real-world data available about what happens when a nuclear emergency occurs. In this case, we used expert validation. We presented the results to the experts in the emergency plan and we conducted interviews to validate them.

Figure 15 shows how many times each device was used based on the failure probability of mobile phones (i.e., from 10% to 90% failure probability). These results can be used to identify the most critical communication mechanisms, and to discuss the validity of the results with field experts. Figure 15 (in_person) shows that the median for *in situ* communications is equal for all the failure probabilities. However, when the failure probability of cell phones is low (10% and 20%), variability is wider. Only in these cases (i.e., when the communication process is not blocked) are other commands (i.e., "Establish emergency level 1, 2, 3") transmitted. Figure 15 (ladlinephone) shows an increasing trend in the mean and median in using landline phones and Figure 15 (mobilephone) depicts a decreasing trend for mobiles. When the mobile phone fails, the individuals with access to a landline phone stop using mobiles. Moreover, since there are individuals who only have access to mobiles, the probability to establish communication is lower. As individuals do not know why there is no response, they keep trying to communicate.

In Figure 15 (beeper), we see that beeper is not used (the mean, median, and quartiles are all zero). We obtained the same results for fax, e-mail, private landline phone, REMAR, REMER, satellite phone, and TrankiE. As we assumed that anybody in the Health group has access to a beeper and the specification document (detailed in Section 4.1), this means that the other devices are not used by the Health group, so we can conclude that the previous results are correct. In Figure 15 (radiological_group_device) and (tranki_gc), we can see that data distribution is uniform for TrankiGC and RGD when we simulate failures in mobiles. The number of attempts to establish communication leads to variability. These results validate the model based on NEP specifications, which states that the Health group does not use these two devices. These restrictions justify the plots in Figure 15 (beeper), (radiological_group_device), and (tranki_gc), which are uniform for the different failure probabilities.

We can also use other scenarios for expert validation; for example, Figures 16 and 17 show the results in the Health Group when the NEP Director sets Emergency Level 0 and 1 and the fax and the satellite phone fail with different failure probabilities (i.e., from 10% to 90% failure probability).

As these communications mechanisms are not used in the Health Group, we expect 49 individuals to receive commands. In the figures, we can see that the commands are transmitted as expected according to the NEP specifications. These scenarios were anticipated by the field experts.

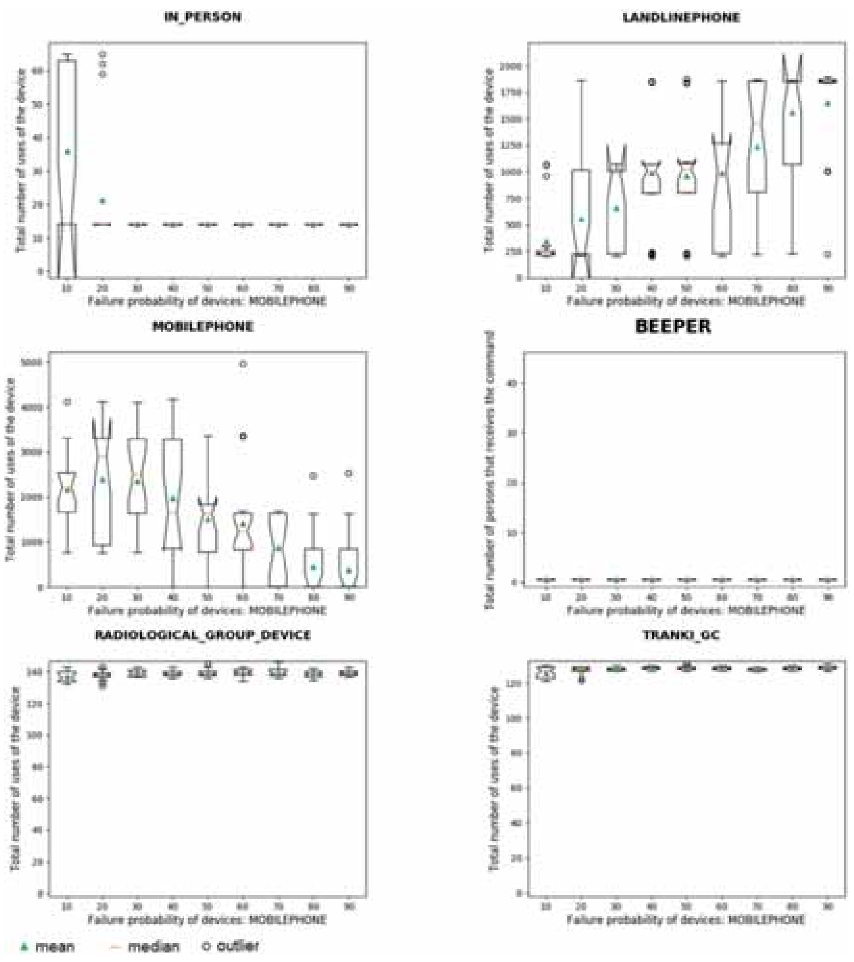


Fig. 15. Number of activations of different devices when cell phones fail with different probabilities.

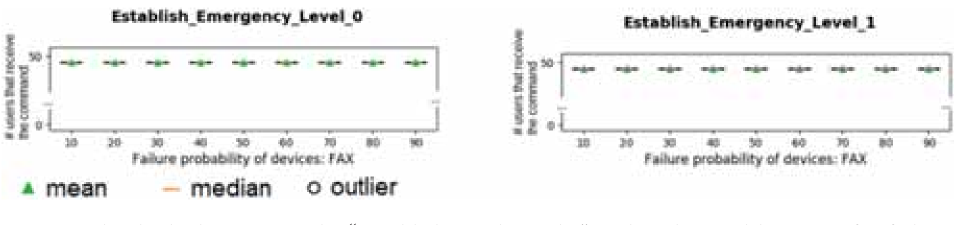


Fig. 16. Individuals that receive the "Establish Level 0 and 1" within the Health Group, fax failures.

5 BENEFITS OF ADPM

As discussed earlier, the aim of this research is to define (1) an *Architecture* to study Diffusion Processes in Multiplex dynamic networks and (2) a systematic *Process* to define, implement, and simulate diffusion processes over such networks. To do so, we provide a method to simulate diffusion processes over networks by systematically maintaining separations of concerns. The main advantages of ADPM and the presented process are:

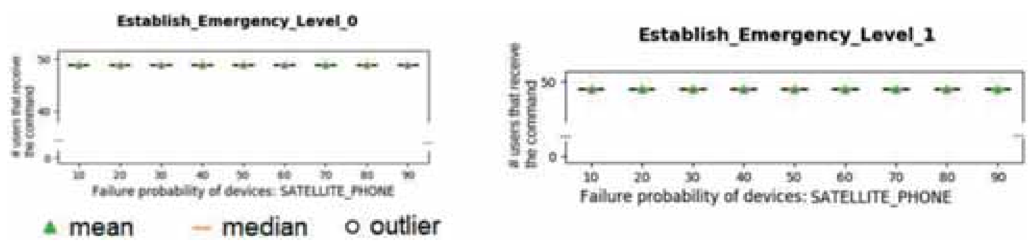


Fig. 17. Individuals who receive "Establish Level 0 and 1" in the Health Group, satellite failures.

- Reusability
- Complex agents with varied behaviors
- Configurability and ease to run experiments
- Captures the change over time of network characteristics

In this section, we discuss these advantages using the results of the case study and other specific examples.

5.1 Reusability

ADPM provides separation of concerns. The modeling, implementation, and experimentation phases of diffusion processes in multiplex networks are separated. As seen in Section 4, we first define a formal model that can be analyzed prior to any software implementation and fix any modeling issues found. Then we provide an executable model on a specific simulator (which can be independent of the model specification). This simulated model can be experimented in different scenarios, as shown in Section 4.6. Experiments are independent of the model definition and execution, and the entire process is automated.

This separation of concerns helps with early validation. The formal model can provide valuable information on early validation prior to implementation. We do not need the model implementation to start validation. As we exemplify in Section 3.4, with the model defined in Appendix A (Generator Filter), we can study if we defined the correct behavior for the state variable md^b, which accumulates the messages that have passed the filter. We can do these early checks for every single model that we define.

We can also reuse the models defined. Once a DEVS model is defined, it can be stored in a model library and be reused to study similar problems. More specifically, we could reuse the models of different devices and communication networks to study how rumors are spread in a community. We would need to redefine the behavior of the individuals (i.e., nodes), but we could reuse the models of the devices. We reused these models in the context of a new project focused on the resiliency of communications with a drone-based IoT platform [40]. The reusability concept is applicable both at the model definition level (as the same model can be implemented on different platforms) and at the implementation level.

5.2 Complex Agents with Varied Behaviors

In ADPM, there is no restriction to the complexity of the behavior of the agents. As such behavior is defined using ABM techniques and, in our case, implemented with an XML file, any behavior can be modeled in the Agent component of ADPM. The behavior of agents presented in the case study in Section 4 offers an example in which for every single message that agents receive, their response differs. Additionally, agents can have different behaviors. In the case study presented in Section 4, each agent responds in a unique way to every single message. For our case study, all the agents are defined with the same set of parameters, but different agents can be defined with different sets of parameters. We will merely need to consider this when formally defining the behavior rules component of the node, as explained in Section 3. In that case, we have as many parameterized behavior rule models as agent "types."

5.3 Configurability and Ease to Run Experiments

ADPM allows us to simulate diffusion processes where the behavior changes without modifying the model definition and implementation. If an agent's behavior changes but the parameters that define this behavior are the same, we update the corresponding value in the XML Agent. For example, the definition of an agent's behavior can include, as a parameter, "devices to be used." In a specific scenario, the agent can use a phone and a radio, or only the Internet and a phone in a different scenario. The value of the parameter changes, but the parameter in the specification is the same. These values can be updated in the XML Agent and the model is regenerated automatically. The model now differs, but no modifications to the formal definition or the actual implementation are necessary (i.e., the high-level metamodel remains unchanged but is instantiated differently). This is also useful if we want to modify the number of agents in the scenario as we did in our case study. We analyzed the Radiological and the Health groups without making any modifications in the formal model definition or implementation: simply updating the behavior of the agents in the XML Agent file suffices.

It is also easy to update the models' parameters. Different scenarios and network configurations can be run by simply updating the model parameters. As mentioned above, by changing the relations in the ABM (i.e., XML in our case), we can study different scenarios for different network configurations. We can also easily include and remove agents (i.e., nodes) by adding or deleting XML files from the agent-based model.

5.4 Captures Changes in Network Characteristics Over Time

The topology and characteristics of the network can change over time in a specific scenario. ADPM uses four models to update the properties/parameters of the DAM components during a simulation run. As we showed in our case study, this allows us to modify the state of the devices at runtime. We can also update the state of the networks. In this way, we can simulate a network collapse and recovery without storing all the network configurations.

5.5 Limitations

A limitation of the present version of our approach is that the modeler needs to know the formal modeling methodology followed (DEVS in our example). Another limitation is that if new parameters are added to the ABM, the definition of the atomic DAM components should be updated accordingly. However, thanks to the modular design of DAM, we only need to modify the affected components. The same occurs with any new additional components (i.e., Link Types) that need to be added.

6 CONCLUSIONS

We propose ADPM, an architecture and development process based on a formal M&S methodology to simulate diffusion processes in multiplex networks. We used Network Theory to define a Network model, and Agent-Based Modeling to define an Agent model. Both models were utilized to develop the *Diffusion Abstract model.* ADPM provides several advantages:

- Different scenarios and network configurations can be run by simply updating the model parameters. There is no need to make changes in the model design.

- There is no restriction to the complexity of the behavior of the agent. Any behavior can be modeled in the Agent component of ADPM.
- Different agents can have quite different behavior.
- We improve reusability (as the behavior of the agents and objects are separate, we can reuse these models to study other problems).
- Using four models to update the properties of the components allows us to simulate diffusion processes in which the topology or characteristics of the network change over time. We can update the network topology and the behavior of both the nodes and the links at runtime. Moreover, we do not need to store the whole model again with the new properties.

We showed how to apply ADPM to build and simulate an information diffusion process in a multiplex network, specifically an emergency plan in Spain. ADPM follows a formal model development approach and implementation for diffusion processes in multiplex networks, which is a novel approach. It also improves the model definition. Using an independent simulation engine simplifies the verification process and experimentation. As the model definition and implementation are separate, we can start validating the model as soon as it is formally defined.

Future research includes automating the analysis and visualization of the simulation results, allowing data to be shared with decision makers by easily improving the communication with stakeholders.

APPENDICES

A DIFFUSION ABSTRACT MODEL: FORMAL DEFINITION USING DEVS

The *Diffusion Abstract Model* (DAM), presented in Figure 3, is formally defined using DEVS as follows:

$$DAM = \langle X, Y, D, \{M_d | d \in D\}, \; EIC, EOC, IC \rangle$$

where

$$X = \emptyset; \quad Y = \emptyset$$

$$D = \left\{ \begin{array}{c} Node_1,\ Node_2\ , \ldots , Node_n, \\ IndirectLink_1, IndirectLink_2\ , \ldots , IndirectLink_n, \\ DirectLink,\ LinkConnectors,\ DiffusionElementGenerator, \\ NodeUpdater, IndirectLinkUpdater, DirectLinkUpdater, \\ LinkConnectorsUpdater \end{array} \right\}$$

$$M = \left\{ \begin{array}{c} M_{Node1}, M_{Node2}, \ldots , M_{Noden} \\ M_{IndirectLink1}, M_{IndirectLink2}, \ldots , M_{IndirectLinkn} \\ M_{DirectLink}, M_{LinkConnectors}, M_{DiffusionElementGenerator} \\ M_{NodeUpdater}, M_{IndirectLinkUpdater}, M_{DirectLinkUpdater}, \\ M_{LinkConnectorsUpdater} \end{array} \right\}$$

$$EIC = \emptyset; \quad EOC = \emptyset$$

IC=

$$\left\{\begin{array}{c}
((DiffusionElementGenerator, Out), (Node_1, InitialDiffusionElement_{In}))\\
\ldots\\
((DiffusionElementGenerator, Out), (Node_n, InitialDiffusionElement_{In}))\\
((NodeUpdater, Out), (Node_1, PropertyUpdate_{In})),\\
\ldots\\
((NodeUpdater, Out), (Node_n, PropertyUpdate_{In})),\\
((IndirectLinkUpdater, Out), (IndirectLink_1, PropertyUpdate_{In}))\\
\ldots\\
((IndirectLinkUpdater, Out), (IndirectLink_n, PropertyUpdate_{In}))\\
((DirectLinkUpdater, Out), (DirectLink, PropertyUpdate))\\
((LinkConnectorsUpdater, Out), (LinkConnectors, PropertyUpdate))\\
((Node_1, PropertyUpdate_{Out}), (NodeUpdater, In))\\
\ldots\\
((Node_n, PropertyUpdate_{Out}), (NodeUpdater, In))\\
((Node_1, DiffusionElementDirect_{Out}), (DirectLink, DiffusionElement_{In}))\\
\ldots\\
((Node_n, DiffusionElementDirect_{Out}), (DirectLink, DiffusionElement_{In}))\\
((Node_1, DiffusionElementIndirect_{Out}), (IndirectLink_1, NodeDiffusionElement_{In}))\\
\ldots\\
((Node_n, DiffusionElementIndirect_{Out}), (IndirectLink_n, NodeDiffusionElement_{In}))\\
((DirectLink, DiffusionElement_{Out}), (Node_1, DiffusionElementDirect_{In}))\\
\ldots\\
((DirectLink, DiffusionElement_{Out}), (Node_n, DiffusionElementDirect_{In}))\\
((IndirectLink_1, NodeDiffusionElement_{Out}), (Node_1, DiffusionElementIndirect_{In}))\\
\ldots\\
((IndirectLink_n, NodeDiffusionElement_{Out}), (Node_n, DiffusionElementIndirect_{In}))\\
((IndirectLink_1, ConnectorDiffusionElement_{Out}), (LinkConnectors, DiffusionElement_{In}))\\
\ldots\\
((IndirectLink_n, ConnectorDiffusionElement_{Out}), (LinkConnectors, DiffusionElement_{In}))\\
((LinkConnectors, DiffusionElement_{Out}), (IndirectLink_1, ConnectorDiffusionElement_{In}))\\
\ldots\\
((LinkConnectors, DiffusionElement_{Out}), (IndirectLink_n, ConnectorDiffusionElement_{In}))
\end{array}\right\}$$

$n = \#nodes\ in\ the\ Network\ Model$

The rest of the coupled models inside the DAM are defined similarly to the above model.

B GENERATOR FILTER: FORMAL DEFINITION USING DEVS

The formal definition of the Generator Filter atomic model is as follows:

$$GeneratorFilter(Id) = \langle X^b, Y^b, S, ta, \delta_{ext},\ \delta_{int},\ \delta_{con},\ \lambda\rangle$$

$$X^b = \{(\text{“In”}, d^b)\} \quad Y^b = \{(\text{“Out”}, d^b)\} \quad S = \{md^b\}$$

$$ta(S) = \left\{\begin{array}{l} md^b = \{\} \ \rightarrow\ \infty \\ md^b \neq \{\} \ \rightarrow 0 \end{array}\right\}$$

$$\delta_{ext}(S,e,X) = \delta_{con}(S,e,X) = \left\{ \begin{matrix} \forall\ d\ in\ X^b \\ if\ d_0 = Id \\ then\ md^b += d \end{matrix} \right\}$$

$$\delta_{int}(S) = \{md^b = \{\}\} \quad \lambda(S) = \{md^b\}$$

Here "In" and "Out" are the names of the input and output ports, respectively; $d^b = md^b = \{d \in \mathbb{N}^4\}$ are bags of natural numbers; d_0 represents the receiver of the message; d_1 represents the sender of the message; d_2 represents the transmitted information; and d_3 represents the network layer used to transmit the message.

C GENERATOR FILTER: IMPLEMENTATION IN CADMIUM

```
struct generatorFilter_defs{  //Declaration of the ports in the atomic
    struct out: public out_port<DiffusionElement> {};
    struct in: public in_port< DiffusionElement > {};
};
template<typename TIME>   //Atomic model definition
class generatorFilter {
    using defs= generatorFilter_defs;
    public:
    using input_ports=tuple<typename defs::in>; //Input ports definition
    using output_ports=tuple<typename defs::out>; //Output ports
     definition
    string id; //Model parameter
    struct state_type{ //Model state declaration
      vector<DiffusionElement> messagesPassingFilter;};
    state_type state; //Model state definition
    generatorFilter (string Id) noexcept { //Constructor & state
     initialization
      id=Id;
      state. messagesPassingFilter.clear();
    }
    void internal_transition() {//Internal transition
      state.messagesPassingFilter.clear();}
    void external_transition(TIME e,typename make_message_bags
     <input_ports>::type mbs){
      for (const auto &x: get_messages<typename defs::in>(mbs))
       {6 //Atomic model definition
         if(x.destinatary == id) state.messagesPassingFilter.
          emplace_back(x);
     }
    }
    void confluence_transition(TIME e,typename make_message_bags
     <input_ports>::type mbs){
      internal_transition();
      external_transition(TIME(), move(mbs));
    } //Confluence transition
    typename make_message_bags<output_ports>::type output() const
      {//Output function
      typename make_message_bags<output_ports>::type bags;
        for (int i = 0; i < (state. messagesPassingFilter.size()); i++){
          get_messages<typename defs::out>(bags).push_back(state.
            messagesPassingFilter[i]);}
      return bags;
    }
TIME time_advance() const {//Time advance function
      return (state.messagesPassingFilter.empty() ? numeric_limits<TIME>::
       infinity(): TIME("00:00:00:001"));}
}
```

The *Generator Filter* model filters the messages in the *in* port based on the model Id. When the messages pass the filter criteria, they are sent through the out port. We start by defining the input and output ports in the model. Then we implement the DEVS functions: internal transition, external transition, confluence, output, and time advance. To do so, we define the DEVS function for the filter. The internal transition function clears *msgPassingFilter*. The external transition function stores the messages received through the input port in the *msgPassingFilter* variable if the field "to" of the message matches the model's Id. The output function sends the messages stored in the *msgPassingFilter* variable through the output port. Finally, the time advance function passivates the model if there is nothing to send and sets a time advance of 1ms if there is something to send.

D DIFFUSION ABSTRACT COUPLED MODEL: IMPLEMENTATION IN CADMIUM

To implement the DAM, we translate all the components in the formal definition (Appendix A) to Cadmium syntax. First, we define the model's input and output ports of the model as a tuple. For the DAM, they are an empty tuple. Second, we define the subcomponents of the models using the keyword *models_tuple*, which includes the name of all the DAM components (both atomic and coupled) defined in M. Third, we define the external input and output couplings (EICs and EOCs) as tuples. In the DAM, they are empty. Then we define the internal couplings (ICs). The IC is a tuple that includes the IC specified in the formal definition. Finally, we define the DAM as a coupled model. The coupled model is defined as a tuple (i.e., coupled model) that contains all the elements previously implemented and a TIME-type parameter.

```
using iports_DAM = tuple<>; //Input ports
using oports_DAM = tuple<>; //Output ports
using submodels_DAM = models_tuple< //Components
      Node_1, Node_2, ..., Node_n,
      IndirectLink_1, IndirectLink_2, ..., IndirectLink_n,
      DirectLink, LinkConnectors, DiffusionElementGenerator,
     NodeUpdater, IndirectLinkUpdater, DirectLinkUpdater,
     LinkConnectorsUpdater >;
using eics_DAM = tuple< >;   //External Input Couplings
using eocs_DAM = tuple<  >;    //External Output Couplings
using ics_DAM = tuple<   //Internal Couplings
    IC<DiffusionElementGenerator, DiffusionElementGenerator::Out, Node_1,
        Node_1::InitialDiffusionElementIn>, ...,
    IC<NodeUpdater, NodeUpdater::Out, Node_1, Node_1::PropertyUpdateIn>, ...,
    IC<IndirectLinkUpdater, IndirectLinkUpdater::Out, IndirectLink$_1$,
        IndirectLink$_1$::PropertyUpdateIn>, ...,
    IC<DirectLinkUpdater,DirectLinkUpdater::Out, DirectLink,
      DirectLink::PropertyUpdateIn>, ...,
    IC<LinkConnectorsUpdater, LinkConnectorsUpdater::Out, LinkConnectors,
          LinkConnectors::PropertyUpdateIn>, ...,
    IC<Node$_1$, Node$_1$::PropertyUpdateOut, NodeUpdater, NodeUpdater::In>,
      ...,
    IC<Node$_1$, Node$_1$:: DiffusionElementDirectOut, DirectLink,
          DirectLink::DiffusionElementIn>, ...,
    IC<Node$_1$, Node$_1$:: DiffusionElementIndirectOut, IndirectLink$_1$,
          IndirectLink$_1$::DiffusionElementIn>, ...,
    IC<DirectLink, DirectLink::DiffusionElementOut, Node$_1$, Node$_1$::
     DiffusionElementDirectIn>, ...,
    IC<IndirectLink$_1$, IndirectLink$_1$::DiffusionElementOut, Node$_1$,
          Node$_1$::DiffusionElementIndirectIn, >, ...,
    IC<IndirectLink$_1$, IndirectLink$_1$::ConnectorDiffusionElementOut,
      LinkConnectors,
          LinkConnectors::DiffusionElementIn>, ...,
    IC<LinkConnectors, LinkConnectors::DiffusionElementOut, IndirectLink$_1$,
          IndirectLink$_1$::ConnectorDiffusionElementIn>, ...>;
template<typename TIME> //Coupled model
    struct DAM: public coupled_model<
      TIME, iports_DAM, oports_DAM, submodels_DAM, eics_DAM, eocs_DAM,
        ics_DAM>{};
```

REFERENCES

[1] A. Vespignani. 2012. Modeling dynamical processes in complex socio-technical systems. *Nature Physics* 8 (2012), 32–39.

[2] E. Abrahamson and L. Rosenkopf. 1997. Social network effects on the extent of innovation diffusion: A computer simulation. *Organization Science* 8 (1997), 289–309.

[3] M. Newman. 2010. *Networks: An Introduction.* Oxford University Press.

[4] S. Gómez, A. Díaz-Guilera, J. Gómez-Gardeñes, C. J. Pérez-Vicente, Y. Moreno, and A. Arenas. 2013. Diffusion dynamics on multiplex networks. *Physical Review Letters* 110, 2 (2013), 028701.

[5] H. Rahmandad and J. Sterman. 2008. Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models. *Management Science* 54, 5 (2008), 998–1014.

[6] T. Schelling. 1978. *Micromotives and Macrobehavior.* Norton, New York.

[7] R. Axelrod. 1997. The dissemination of culture—a model with local convergence and global polarization. *Journal of Conflict Resolution* 41, 2 (1997), 203–226.

[8] J. M. Epstein. 2006. *Generative Social Science: Studies in Agent-Based Computational Modeling.* Princeton University Press, Princeton, NJ.

[9] D. Levinthal and J. G. March. 1981. A model of adaptive organizational search. *Journal of Economic Behavior and Organization* 2, 4 (1981), 307–333.

[10] R. Axtell, J. Epstein, J. Dean, G. Gumerman, A. Swedlund, J. Harburger, S. Chakravarty, R. Hammond, J. Parker, and M. Parker. 2002. Population growth and collapse in a multiagent model of the Kayenta Anasazi in Long House Valley. *Proceedings of the National Academy of Sciences USA* 99, Suppl. 3 (2002), 7275–7279.

[11] L. Tesfatsion. 2002. Economic agents and markets as emergent phenomena. *Proceedings of the National Academy of Sciences USA* 99, Suppl. 3 (2002), 7191–7192.

[12] F. Bass. 1969. A new product growth model for consumer durables. *Management Science* 15, 5 (1969), 215–227.

[13] R. M. Anderson and R. M. May. 1991. *Infectious Diseases of Humans Dynamics and Control.* Oxford University Press, Oxford, UK.

[14] P. Samuelson. 1939. Interactions between the multiplier analysis and the principle of acceleration. *Review of Economic Statistics* 21 (1939), 75–79.

[15] H. Xiong, P. Wang, and G Bobashev. 2015. Multiple peer effects in the diffusion of innovations on social networks: A simulation study. *Journal of Innovation and Entrepreneurship* 7, 1 (2015), 2.

[16] B. Zeigler, H. Praehofer, and T. Kim. 2000. *Theory of Modeling and Simulation*. Academic Press.

[17] M. Newman, A.-L. Barabasi, and D. J. Watts. 2006. *The Structure and Dynamics of Networks*. Princeton University Press.

[18] A. Sole-Ribalta, M. De Domenico, S. Gómez, and A. Arenas. 2014. Centrality rankings in multiplex networks. In *Proceedings of the 2014 ACM Conference on Web Science*.

[19] S. Boccaletti, G. Bianconi, R. Criado, C. I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. 2014. The structure and dynamics of multilayer networks. *Physics Reports* 544, 1 (2014), 1–122.

[20] M. Kivela, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. 2014. Multilayer networks. *Journal of Complex Networks* 2, 3 (2014), 203–271.

[21] V. Capasso and G. Serio. 1978. A generalization of the Kermack-McKendrick deterministic epidemic model. *Mathematical Biosciences* 42, 1–2 (1978), 43–61.

[22] W. Wang, Q.-H. Liu, S.-M. Cai, M. Tang, L. A. Braunstein, and H. E. Stanley. 2016. Suppressing disease spreading by using information diffusion on multiplex networks. *Scientific Reports* 6, 7600 (2016), 29259.

[23] C. Granell, S. Gomez, and A. Arenas. 2013. Dynamical interplay between awareness and epidemic spreading in multiplex networks. *Physical Review Letters* 111, 12 (2013), 1–10.

[24] A. Khelil, C. Becker, J. Tian, and K. Rothermel. 2002. An epidemic model for information diffusion in MANETs. In *Proceedings of the 5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems GA*.

[25] O. Yağan and V. Gligor. 2012. Analysis of complex contagions in random multiplex networks. *Physical Review E - Statistical, Nonlinear, Biological, and Soft Matter Physics* 86, 3 (2012), 1–11.

[26] E. Cozzo, R. A. Baños, S. Meloni, and Y. Moreno. 2013. Contact-based social contagion in multiplex networks. *Physical Review E - Statistical, Nonlinear, Biological, and Soft Matter Physics*. 88, 5 (2013), 1–5.

[27] N. Gilbert. 2007. *Agent-Based Models*. Sage, London.

[28] N. R. Jennings, K. Sycara, and M. Wooldridge. 1998. A roadmap of agent research and development. *Autonomous Agents and Multi-agent Systems* 38 (1998), 7–38.

[29] B. Edmonds. 2000. *The Use of Models - Making MABS More Informative: International Workshop on Multi-Agent Systems and Agent-Based Simulation*. Springer, Berlin.

[30] J. M. Galán, L. R. Izquierdo, S. S. Izquierdo, J. I. Santos, R. Del Olmo, R. A. Lopez-Paredes, and B. Edmonds. 2009. Errors and artefacts in agent-based modelling. *Journal of Artificial Societies and Social Simulation* 12 (2009), 1.

[31] Y. Jiang and J. C. Jiang. 2015. Diffusion in social networks: A multiagent perspective. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 2 (2015), 198–213.

[32] J. Himmelspach and A. M. Uhrmacher. 2004. A component-based simulation layer for JAMES. In *Proceedings of the 18th Workshop on Parallel and Distributed Simulation*.

[33] J.-B. Filippi and P. Bisgambiglia. 2004. JDEVS: An implementation of a DEVS-based formal framework for environmental modeling. *Environmental Modelling & Software* 19, 3 (2004), 261–274.

[34] H. S. Sarjoughian and B. P. Zeigler. 1998. DEVSJAVA: Basis for a DEVS-based collaborative M & S environment. In *International Conference on Web-Based Modeling and Simulation*.

[35] L. Belloli, D. Vicino, C. Ruiz-Martin, and G. Wainer. 2019. Building devs models with the cadmium tool. In *Winter Simulation Conference (WSC'19)*.

[36] Y. Bouanan, G. Zacharewicz, B. Vallespir, J. Ribault, and S. Y. Diallo. 2016. DEVS based network: Modeling and simulation of propagation processes in a multi-layers network. In *The Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems 2016*.

[37] C. Ruiz-Martin, Y. Bouanan, G. Wainer, G. Zacharewicz, and A. Lopez-Paredes. 2016. A hybrid approach to study communication in emergency plans. In *Winter Simulation Conference*.

[38] Y. Bouanan. 2016. Contribution à une architecture de modélisation et de simulation à evénements discrets: Application à la propagation d'information dans les réseaux sociaux. *Bordeaux*.

[39] M. Bastian, S. Heymann, and M. Jacomy. 2009. Gephi: An open source software for exploring and manipulating networks. In *3rd International AAAI Conference on Weblogs and Social Media (ICWSM'09)*, 8, 361–362.

[40] W. De Nooy, A. Mrvar, and V. Batagelj. 2005. *Exploratory Social Network Analysis with Pajek*. Cambridge University Press.

[41] M. De Domenico, M. A. Porter, and A. Arenas. 2015. MuxViz: A tool for multilayer analysis and visualization of networks. *Journal of Complex Networks* 3, 2 (2015), 159–176.

[42] C. Nikolai and G. Madey. 2009. Tools of the trade: A survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation* 12 (2009), 22.

[43] C. Ruiz-Martin, G. Wainer, and A. Lopez-Paredes. 2018. Formal abstract modeling of dynamic multiplex networks. In *SIGSIM-Principles of Advanced Discrete Simulation*.

[44] R. Ihaka and R. Gentleman. 1996. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5, 3 (1996), 299–314.

[45] Microsoft. 2015. Power BI. Retrieved from https://powerbi.microsoft.com/es-es/.

[46] R. G. Sargent. 2013. Verification and validation of simulation models. *Journal of Simulation* 7 (2013), 12–24.

[47] C. Ruiz-Martin. 2013. *Modelo organizational para la gestión de emergencias.* Master Thesis, Universidad de Valladolid.

[48] C. Ruiz-Martin, M. Ramírez Ferrero, J. L. Gonzalez-Alvarez, and A. Lopez-Paredes. 2015. Modelling of a nuclear emergency plan: Communication management. *Human and Ecological Risk Assessment: An International Journal* 21, 5 (2015), 1152–1168.

[49] C. Ruiz-Martin, A. López-Paredes, and G. Wainer. 2018. What we know and do not know about organizational resilience. *International Journal of Production Management and Engineering* 6, 1 (2018), 11–28.

[50] C. Ruiz-Martin. 2019. Resilience analysis method and tool for humanitas systems. *NSERC Engage Project. Technical Report, Carleton University.*

Received August 2019; revised June 2020; accepted October 2020