Journal of Computing and Security

January 2018, Volume 5, Number 1 (pp. 35–49)

http://www.jcomsec.org

Towards a Model-Driven Framework for Simulating Interactive Emergency Response Environments

Samaneh Hoseindoost [a], Afsaneh Fatemi [a,*], Bahman Zamani [a],

[a] *MDSE Research Group, Department of Software Engineering, University of Isfahan, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: 30 July 2018
Revised: 02 September 2018
Accepted: 12 December 2018
Published Online: 26 December 2018

Keywords:
ERE-ML 2.0, Model-Driven Framework, Multi-Agent Systems, Interactive Systems, Emergency Response Environments, Model Driven Development.

ABSTRACT

Due to the increasing occurrence of unexpected events and the need for pre-crisis planning to reduce risks and losses, modeling emergency response environments (ERE) is needed more than ever. Modeling may lead to more careful planning for crisis-response operations, such as team formation, task assignment, and doing the task by teams. ERE-ML is a model-driven framework which allows a crisis manager to model an ERE, and to automatically generate the executable code of a multi-agent system (MAS) for that environment. However, the application generated by ERE-ML lacks the capability of supporting interactions among the agents and the organizations involved in the crisis management. In this paper, we propose ERE-ML 2.0 as an upgrade of the previous framework. The ERE-ML 2.0 framework supports the interactions by adding new features to the ERE-ML language, modifying the transformation code, and extending the platform. To evaluate the upgraded framework, the Plasco Tower Collapse incident is modeled, and then the model is transformed into the executable code of a MAS to visualize the run-time scenarios.

© 2018 JComSec. All rights reserved.

1 Introduction

Nowadays, disasters cause many human and financial losses. In managing the disasters, several agents should cooperate to cope with the crisis. Due to the critical situations and the need for a quick and timely response in facing disasters, using several tools or methods has been suggested.

For example, using emergency response systems (ERS) for supporting communications, data gathering and analysis, and decision-making will be helpful. In other words, these systems assist organizations for responding to an emergency situation by guiding responders to perform the best response actions. Also, some of the systems facilitate communications between the various responding groups and managers [1].

Another way for improving the planning and responding in emergency situations is using modeling and simulation methods. This method can be implemented in a real-world situation [2, 3] or can be implemented as a software application (Emergency Response Application) [4–15]. For example, in a study [2] a series of workshops in increasingly realistic settings have been organized where different state-of-the-art systems and devices were demonstrated and tried out to get a better understanding of the technological possibilities in an emergency response. These workshops were beginning with a stretcher in a lab and one of the researchers acting as car accident victim and ending with a two-day workshop at an emergency response

* Corresponding author.
Email addresses: s.hoseindoost@eng.ui.ac.ir (S. Hoseindoost), a_fatemi@eng.ui.ac.ir (A. Fatemi), zamani@eng.ui.ac.ir (B. Zamani)

https://dx.doi.org/10.22108/jcs.2018.112297.1008
ISSN: 2322-4460 © 2018 JComSec. All rights reserved.

training ground, with three teenagers acting like victims and eight incident situations being enacted by ten professionals using the mock-ups and prototypes. Through these workshops, seven challenges in designing interactive systems for emergency response were identified that describe the major issues to be addressed in designing support for emergency response. Then the design visions and principles addressing these challenges were developed to move towards new ICT support for emergency response. Finally, the 'families' of mock-ups and prototypes were developed, where each family explored one of the visions.

Another research [3] that has focused on simulating an emergency response environment (ERE) in a real-world situation aimed to find the patterns of group information seeking in a simulated ERE. For this purpose, 22 Groups worked on two separate emergency response-related cases (11 groups for each case).

> Each group had five participants: one group coordinator, who served as a facilitator, and four who represented the emergency services of the Police Department, Fire Department, Medical Officer, and Chemical Advisor. The group's task was to make decisions about how to allocate available resources in order to meet the goals of the response [3].

The goal of the research was investigating three hypotheses that were drawn from a series of studies on group decision making in simulated emergency response scenarios. These are as follows [3]:

> H1: Groups with decision support engage in more information-seeking activities than do groups without decision support.
> H2: Novice groups engage in more information-seeking activities than do expert groups.
> H3: The increasing of information-seeking activities with the availability of decision support is higher in novices groups than that in expert groups.

Another research [4] states that "the personnel and artifacts involved in a response operation forms an emergency response system" and has been modeled the system as a social network to analyze the interactions between the personnel engaged in the response operations. The nodes of the model are the personnel who were involved in the emergency response, and the edges represent the interactions between them. For building the model, documentation from an accident scenario was used to identify the different organizations and personnel participating in the operation, and then a questionnaire was published on the internet in which the agents in question were asked to provide information regarding which other agents they had contact with during the response operation.

In the case of modeling and simulation of EREs, several software applications are also developed to simulate the behavior of the involved agents [5–14]. The simulation results can be used to perform the necessary operations and reactions more effectively during a critical situation. Most of these tools have focused on a specific aspect of an emergency situation and do not provide a picture of the overall emergency incident response to planners, trainers, and responders. These tools need to be integrated together to reduce the time and effort for their use. In this regard, Jain and McLean [15] have proposed a framework to show that modeling and simulation tools can be systematically integrated together to address the overall response, but no implementation of the framework is done in this paper.

Despite these tools, the problem is that EREs are dynamic environments in which several entities with different behaviors exist; the victims, rescuers, organizations, and individuals observing an incident or a natural disaster, to name a few. The dynamic environment including these entities and their interactions forms a complex system [16]. Developing a simulation application to simulate a complex system is both difficult and complicated. In this regard, using new software development approaches such as model-driven engineering (MDE) is helpful. Using MDE, one can model a system at a high level of abstraction and then transform it automatically into the code. The generated code may be the partial code of a system and then is completed as a full system [17].

In our previous work [18], a model-driven framework named ERE-ML has been built which is used to develop a multi-agent system (MAS) in an ERE. This framework includes a domain-specific modeling language (DSML) named ERE-ML, the corresponding tool, the transformation code for automatic code generation from the model, and a platform for executing the code. However, ERE-ML lacks the capability of generating interactive systems to support the interactions between the involved agents and the organizations.

In this paper, we propose ERE-ML 2.0, as an upgrade of the previous framework ERE-ML with the aim of having the executable code of an interactive system. In this research by using MDE and DSML the convenient way of enhancing and extending ERE-ML framework is provided that has benefits such as [19]:

- Increasing level of abstraction for the system developer such that he/she doesn't need any programming knowledge
- Saving time and decreasing the cost of the development process because of using the DSML for ERE domain
- Increasing productivity in massive software production due to automatic generation of executable code and no need for coding of a MAS.

The contributions of this paper include adding new features to ERE-ML language, extending the transformation code, as well as extending the platform. To validate the ERE-ML2.0 framework, the Plasco tower collapse incident [20] is modeled using ERE-ML 2.0 modeling tool and then the model is transformed into a Java application. Running this application shows that how the user, agents, and organizations can interact with each other to reduce causalities in the incident scene.

This paper is organized as follows. Section 2 presents the related work which has addressed the problem of model-driven development (MDD) of MAS and ERE. Section 3 describes the necessary interactions among the agents and organizations in an ERE. Section 4 explains the ERE-ML 2.0 framework and elaborates the extensions on the original framework. For this purpose, we describe how ERE-ML and ERE-MLTool (EMT) are extended and then describe the platform extension and the written transformations, respectively. Section 5 evaluates the proposed framework using the Plasco case study, and finally, Section 6 provides conclusions and suggestions for future work.

2 Related Work

In this section, the related works that lie in the intersection of MDD, MAS, and ERE contexts are explained and compared with the ERE-ML 2.0 framework.

Garcia et al. [21] developed the first software application for simulating EREs using model-driven approaches. This study aimed to create an effective interaction pattern among the affected citizens and rescuers in critical situations [22]. In this way, first, the interaction pattern was modeled using a modeling tool called IDK (INGENIAS Development Kit). In this model, groups of agents (mainly citizens, doctors and firefighters) were taken into account with different capabilities and goals. Then, a module named IAF (embedded in the IDK tool) was used for automatic code generation from the model. Finally, the code was completed and is transformed into an executable program by the intervention of the programmer. To evaluate the proposed approach, it has been shown how the resulting application can be used in different scenarios of crisis management. Also, the application was tested and optimized for performance enhancement [23]. The only goal of the research [23] was to provide an efficient interaction pattern between the agents in critical situations, therefore, other aspects of crisis management including team formation, task allocation, and task performance were not considered.

Mustafa et al. [24] presented an organizational methodological framework for modeling and simulation of natural disasters. In this work, the model-driven architecture (MDA) approach was used for system development. First, a platform-independent model (PIM) based on the Conceptual Role Organizational Model (CROM) was created, and then this model is enhanced to the Conceptual Agent Organizational Model (CAOM). In this regard, the models were transformed into models depending on the platform. However, the framework is a theoretical framework and no implementation of that has been proposed.

To the best of our knowledge, the most recent research on the MDD of MAS in EREs is the ERE-ML framework [18]. ERE-ML is a model-driven framework which includes four main components (1) a domain-specific modeling language named ERE-ML for modeling EREs (2) a modeling tool named EMT (3) transformation code for automatic transformation of the model into the code, and (4) a platform for executing the generated code. The framework is an Eclipse plugin with the following capabilities:

(1) Modeling an ERE including disasters, organizations, tasks, and agents.
(2) Checking the designed model against predefined constraints. These constraints are checked to answer the following questions:
    - Are the teams organized properly in the relevant organizations?
    - Are the tasks properly assigned to each team?
    - Are the required resources provided to the teams to perform tasks?
(3) Automatic code generation from the designed model.
(4) Executing the generated code. By executing the Java project, the geographical map of the environment with disasters, organizations, and agents are shown, and the crisis manager can see the team formation, tasks assignment and perform tasks by choosing Organize Team, Assign Tasks and Perform Tasks options, respectively.

Despite the capabilities that the ERE-ML framework gives to the designer, there are deficiencies in the framework as follows:

(1) Initiating the team formation, task allocation, and task performance do not rely on the help request from the agents. Instead, by running the program, the user issues team formation command in all organizations by choosing the "Organize Team" option.
(2) The framework does not support communication between agents and organizations. Hence, the agents are not able to send any help request to the organizations and received the related responses.

(3) The framework does not support interactions between users and agents, as well as users and organizations.

In this paper, we will eliminate the aforementioned weaknesses in the upgraded model-driven framework by adding some new features to ERE-ML language, modifying the transformation code and updating the platform.

3 Interactions Among the Agents and Organizations in an ERE

This section describes the necessary interactions between the agents and organizations that are needed in an ERE.

- When a disaster occurs, the agents that are injured or/and are near to the disaster situation should be able to send a help request to the related organization.
- Each organization should handle these requests such that for each request the related tasks are performed and an appropriate message is sent to the requester.
- For this purpose, when an organization received a message, checks whether the response teams are available or not and sends an appropriate message to the requester.
- If the teams are available in the organization, a command is issued for creating teams and assigning tasks to them. Else a message with contents of "please wait..." is sent to the requester and the organization wait until the related teams become available.
- By assigning tasks to each team, they get ready to perform them and finally perform these tasks.
- After performing the tasks, an appropriate message is sent to the organization about successfully or failingly performing task.

Figure 1 shows the possible interactions among the agents, organizations, and environment by considering the above activities.

4 The ERE-ML 2.0 Framework

Figure 2 shows different parts of the ERE-ML framework [18]. For supporting interactions in the ERE-ML 2.0 framework, three main components of this framework should be extended. Therefore in this section, First, in Section 4.1 we explain how ERE-ML modeling language and its related tool, EMT, will be extended. Then the proposed platform will be described in Section 4.2, and finally, in Section 4.3 the extensions to the transformation code will be explained.

4.1 Extending the Modeling Language and the Tool

ERE-ML [18] is a modeling language that contains MAS and ERE concepts to support modeling an ERE situation. This language has been extended MAS-ML [26] by adding ERE concepts including Disaster, Task, EmergencyManagementTeam, Capability, and Resource. The main ERE concepts are given from Disaster Management Meta-model (DMM) [27]. MAS-ML is a UML profile for modeling MAS that extends UML concepts [28] using the concepts that are related to MAS including Agent, AgentRole, Organization, and Environment. Therefore ERE-ML 2.0 includes UML, MAS-ML and ERE-ML concepts that are shown in Figure 3.

To eliminate the first deficiency of the ERE-ML which was mentioned in Section 2, the agents who are close to the location of a disaster or accident location should be able to send a help request. In other words, the agents who observe a disaster or are victim agents should be identified. For this purpose, a feature named "sight radius" is added to the Agent concept in ERE-ML. Also, the agents that are affected by a task are considered as victims. Since the victim agents, hereafter, could not be a team member, a new constraint is added to the EMT to check it. This constraint has been written with OCL [29] for EmergencyManagementTeam concept in ERE-ML [18]. The constraint has been defined as follows:

Context EmergencyManagementTeam inv:
$self.Member st- > excludesAll(Agent.allInstances()t- > select(a : Agent|a.TaskAffectt- > isEmpty() = false))$

4.2 Extending the Platform

To enable communications between agents and organizations as well as the interactions between users, agents and organizations in the system, extensions are required for the platform of the ERE-ML framework. Accordingly, for each agent and organization in the system, a user interface (UI) as shown in Figure 4 and Figure 5 has been developed.

The user interface which is created for each organization includes the organization name, its geographical location and the following options provided to the crisis manager:

- Can Help: for sending a response to an agent
- Organize Teams: for team formation
- Assign Tasks: to assign tasks to teams
- Perform Tasks: to perform tasks

Furthermore, for each agent, the agent's name, its geographical location, and its status are displayed. During execution, an agent may be in one of the following situations: None, NeedHelp or BeingHelped.

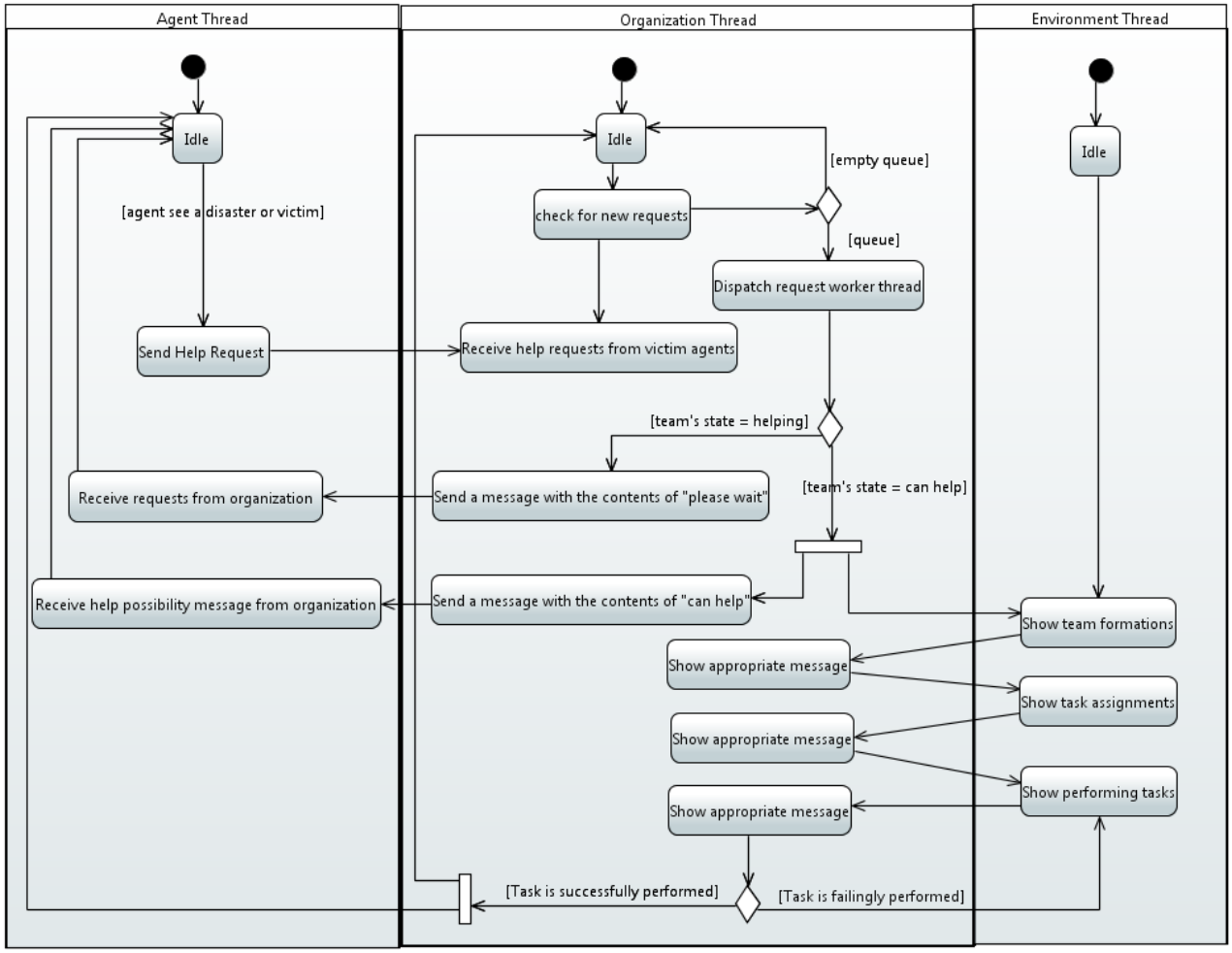

Figure 1. The Possible Interactions Among the Agents, Organizations, and Environment

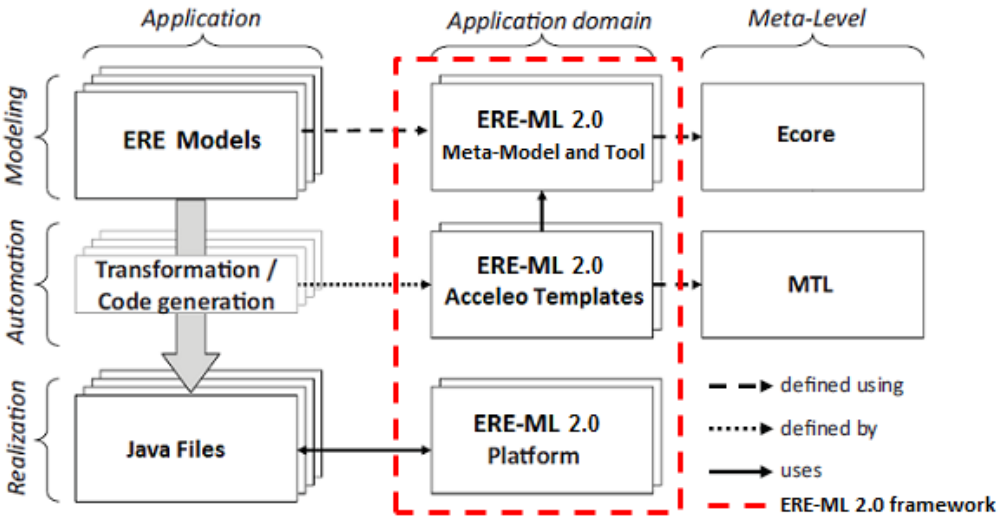

Figure 2. The Different Parts of the ERE-ML 2.0 Framework (Adapted From [25])

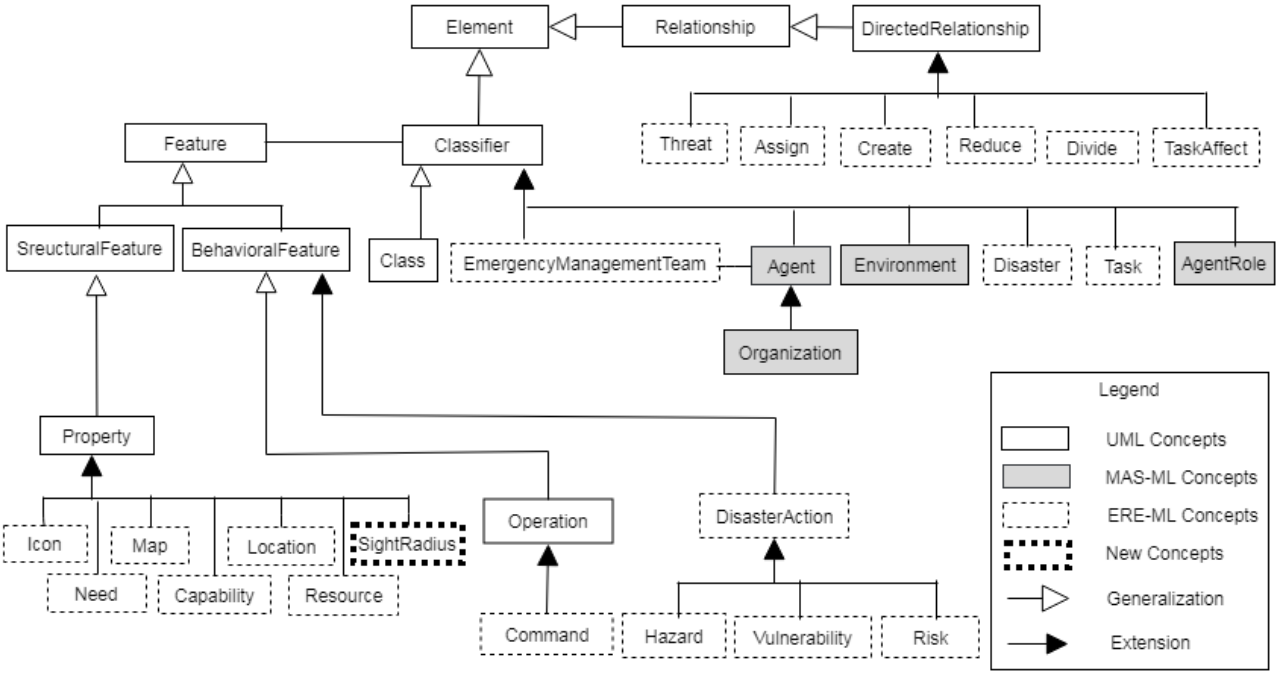

Figure 3. The ERE-ML 2.0 Meta-model

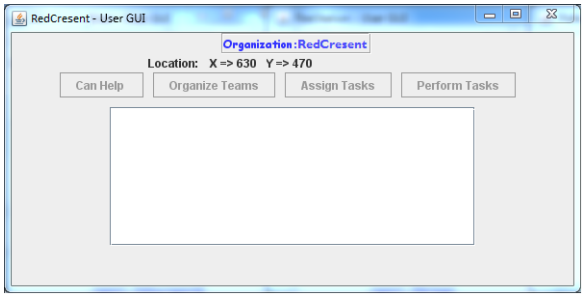

Figure 4. The Crisis Managers' UI

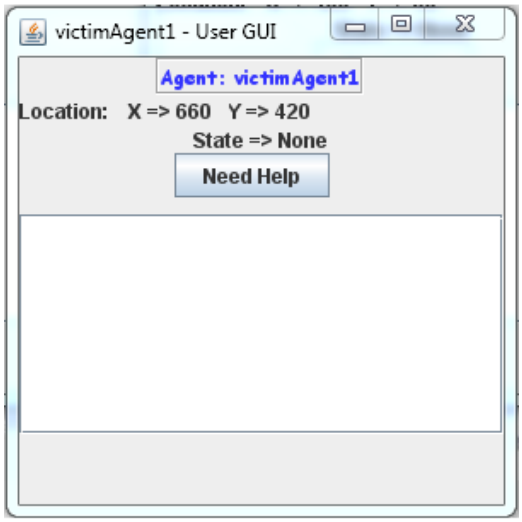

Figure 5. The Agents' UI

Moreover, on the user interface for each agent, a NeedHelp option is considered. This option is enabled only for victim and observer agents. These agents are identified by transformation codes that will be described in Section 4.3.

By running the application, all agents are initially in None status. Choosing the NeedHelp option, the agent's status changes from None to NeedHelp and finally changes from NeedHelp to BeingHelped if his requested tasks are successfully performed.

Each team may be in one of the CanHelp and Helping states, initially in CanHelp. As soon as performing a task begins by a team, the team status changes from CanHelp to helping. Finally, after successfully performing the task, the status returns from helping to CanHelp.

Each organization should be able to receive help request message from agents that are close to the incident location and send the appropriate message to them. Also, if the relevant teams were not engaged in performing tasks, issue commands for team formation, tasks assignment and perform tasks, respectively. In this case, if the teams are available (CanHelp state), the CanHelp option is enabled for the organization. By choosing this option, a message is sent to the agent that shows an organization can help the agent. Then the OrganizeTeam option is enabled which by choosing it, how the teams are formed is shown on the map. After the teams were formed, the AssignTask option is enabled and by choosing it, the relevant tasks will

be assigned to the teams. Finally, the PerformTask is enabled which by choosing it, the command relevant to the executing task is sent to the teams. Otherwise, if the teams are engaged in an operation (helping state) a message with the contents of "The team is doing another operation, please wait..." is sent to the requester agent.

4.3 Extending the Transformation Codes

In the ERE-ML framework [18], for each of the main concepts in the modeling language (including Environment, Agent, Organization, Disaster, and Task) a transformation file is written using Acceleo [30] and MTL (Model to Text Language) [31]. By executing these transformations, each of the elements in the model is transformed into a Java class. In the following, we extend the transformation file relevant to agents (Agent.mtl).

Each agent in the model is transformed into a Java class by executing Agent.mtl file. As mentioned in Section 4.2, the NeedHelp option is enabled only for victim and observer agents. The observer agents are identified based on the Euclidean distance between the agent and the disaster location by considering whether the distance is less than the sight radius of the agent or not. Listing 1 shows how the transformation code in the Agent.mtl file performs this task.

```
[if(anAgent.TaskAffect->isEmpty()=false or
 (Class.allInstances()->exists(cl:Class | cl.
     TaskAffect->isEmpty()=false and
 ((anAgent.Location_x-cl.Location_x)*(anAgent.
     Location_x-cl.Location_x)) +
 ((anAgent.Location_y-cl.Location_y)*(anAgent.
     Location_y-cl.Location_y)) <=
 (anAgent.SightRadius*anAgent.SightRadius)))
 or (DisasterClass.allInstances()->exists(dng:
     DisasterClass | dng.TaskAffect->isEmpty()=
  false and ((anAgent.Location_x-dng.Location_x)*(
      anAgent.Location_x-dng.Location_x)) +
  ((anAgent.Location_y-dng.Location_y)*(anAgent.
      Location_y-dng.Location_y)) <=
  (anAgent.SightRadius*anAgent.SightRadius))))]
  userApp.needHelpB.setEnabled(true);
```

Listing 1: The precondition to enable NeedHelp option for an agent (in MTL language)

By selecting the NeedHelp option, the relevant tasks (which are defined in the model and affected by an agent, object or disaster [18]) are created and added to the environment. Then, the organization that is responsible for the task is identified (through the assigned relationship [18]) and sent to each of them a NeedHelp message. For this purpose, the transformation code shown in Listing 2 is appended to Agent.mtl file.

```
[for(ta: TaskAffect | anAgent.TaskAffect)]
 [if (ta.affectSource.TargetDivision->isEmpty()=true)
      ]
  [let t:Task = ta.affectSource]
   if(env.getTask("[t.name/]") == null){
    org = env.getOrganization("[t.ownedAssign.
        AssignTarget.name/]");
    msg1.addReceiver(new AID(org.getLocalName(), AID.
        ISLOCALNAME));
    // Tasks
    Task [t.name/] = new [t.name/]("[t.name/]", env,
        new File("[t.icon/]"),org);
    try {
     env.addTask("[t.name/]", [t.name/], org.Location.
         x-40, org.Location.y+50);

    } catch (IOException e1) {
     e1.printStackTrace();
    }
    CallTasks.put("[t.ownedAssign.AssignTarget.name
        /]", [t.name/]);
   }
  [/let]
 [else]
  [let t:Task = ta.affectSource.TargetDivision.
      divisionSource->first()]
   if(env.getTask("[t.name/]") == null){
    org = env.getOrganization("[t.ownedAssign.
        AssignTarget.name/]");
    msg1.addReceiver(new AID(org.getLocalName(), AID.
        ISLOCALNAME));
    // Tasks
    Task [t.name/] = new [t.name/]("[t.name/]",env,
        new File("[t.icon/]"),org);
    CallTasks.put("[t.ownedAssign.AssignTarget.name
        /]", [t.name/]);
   }
  [/let]
 [/if]
[/for]
```

Listing 2: Transformation code for identifying the organization responsible for performing tasks

Furthermore, to prevent sending duplicate messages, before identifying the organization and sending messages to it, it is checked whether the relevant task has already been in the environment. If the task exists in the environment, it means that an agent has already requested to the relevant organization to do the task. Therefore, it prevents sending the message by the same agent or other agents to the organization. As a result, the network traffic, especially in large MAS with a large number of agents is reduced significantly.

After writing the transformation codes, a facility should be provided for the user such that he/she can easily run the codes on a model and get the result as a Java project. Therefore, a plugin is created using Acceleo as a context menu on the ERE-ML diagram files. Using this menu, the user can transform the model into a single executable Java project.

5 Case Study

To evaluate the extended framework, the scenario of Plasco Tower collapse is investigated as a case study. In this section, after a short description of this event, the scenario is modeled and validated using ERE-ML tool. In the next step, the model is automatically transformed into a Java application using predefined transformation codes. Finally, the program execution results are presented.

5.1 Plasco Tower Collapse

On Thursday, 19 January 2017, a huge fire took place in the Plasco tower, in the center of Tehran, the capital of Iran. The incident was very intensive that the tower collapsed completely after about four hours of fire and firefighters' efforts to extinguish the fire, and caused sixteen firefighters and ten civilians to be killed [20].

The Plasco tower collapse incident (from now on referred to as the "Plasco incident") is an example of incidents with extreme urgency that many organizations should work together to control it. For example, at the first minutes of the fire, the Red Crescent and fire-fighting stations were ready to take timely and quick response at the incident location, to quench of the fire and on-time relief to the causalities. Also, when the tower collapsed, new tasks were required to be performed in the environment including the removal of the debris and the search for the missing victims. To perform these tasks successfully, organizations such as municipality and police, carried out the relevant tasks as quick as possible. Proper team organization and task allocation played an important role in this process.

5.2 Modeling and Validation of the Model

In this section, the Plasco incident is modeled as an example scenario using the ERE-ML 2.0 tool [1] . Due to the space limitations, in this section, we only explain the part of models that are relevant to the new features of the ERE-ML 2.0 and EMT 2.0.

The organizations involved in the Plasco incident are fire organization, Red Crescent organization, municipality organization and police organization. These organizations along with the defined teams are shown in Figure 6. As an example, the contributing factors of Team 1 in the Red Crescent Organization are shown in this figure. In this step, if instead of Rescuer1, victim1 is placed as one of the team members, an error message is displayed for the user, as shown in Figure 7.

[1] The complete class diagram of Plasco Tower Collapse disaster is available at http://mdse.ui.ac.ir/project/ere-ml2-0-framework/

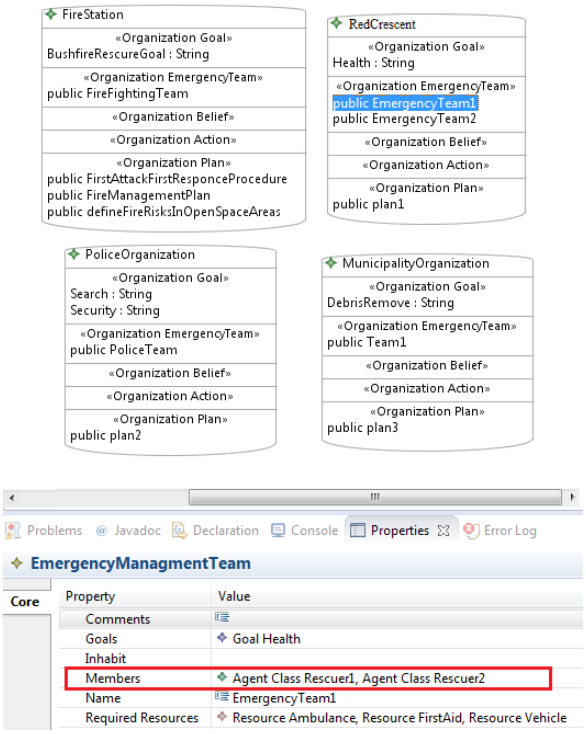

Figure 6. Modeling the Involved Organizations

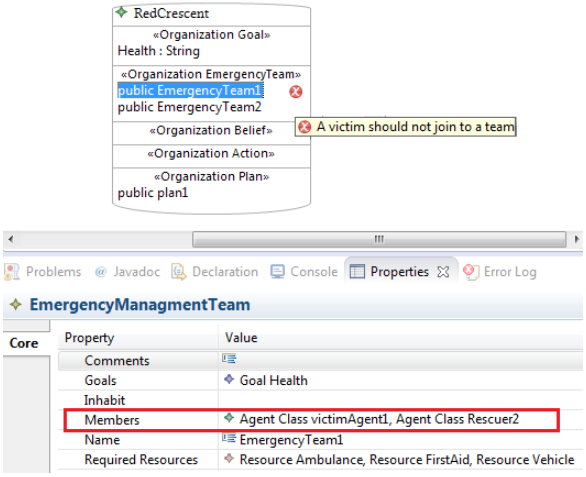

Figure 7. Checking New Constraint Added to ERE-ML

To determine the team members, ten agents including a firefighter, a pilotage, two police agents, four rescuers, and two crane drivers are designed in the model. For each agent, the Sight Radius feature is determined by the modeler. As an example, the rescuer agent characteristics are shown in Figure 8.

5.3 Automatic Code Generation

After the model design and validation in ERE-ML 2.0 tool, the model can be transformed into executable code. For this purpose, by adding the implemented plugin into the Eclipse, the user can right click on the model file and select the Acceleo Model to Text >

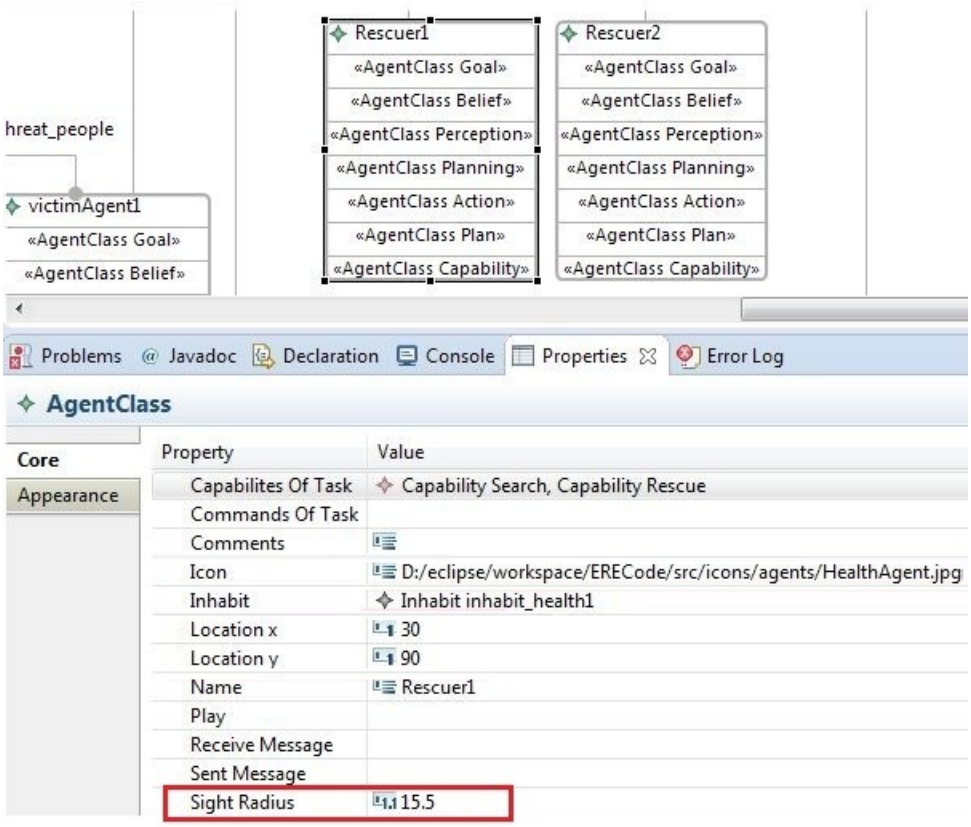

Figure 8. Features of a Rescuer Agent

Generate Java Application. That way, a Java project with the same model name is created, and the Java files are created in its src folder.

As shown in Figure 9, for the case study, 24 Java files are generated automatically for the classes designed in the model. For example, Aid1 and Aid2 Java files are related to rescue tasks 1 and 2. The PilotageAgent, RescuerAgent, FirefighterAgent and VictimAgent classes show the pilots, rescue agents, firefighters, and victim agents, respectively. Also as shown in Listing 3, line 1, the NeedHelp option has been activated for victimAgnet1 since it is near to the disaster scene. Lines 15-17 show that when victimAgent1 chooses the NeedHelp option, a message will be sent to Red Crescent organization for performing the Aid1 task.

5.4 Running the Generated Code for the Plasco Incident

After creating the Java project, the user should add jade.jar and EREML2.0_Platform.jar (the extended platform) libraries to the Java project and then right click on the default package and select Run as Java application. As a result, the main file of the program (*i.e.*, the Tehran.java class) is executed, and instances of all elements existing in the environment are created.

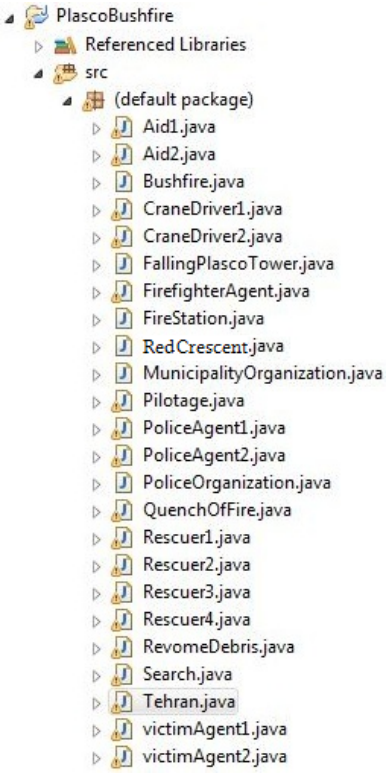

Figure 9. Java Files Generated for Plasco Tower Collapse Case Study

```
1  userApp.needHelpB.setEnabled(true);
2  userApp.needHelpB.addActionListener(new
       ActionListener() {
3   public void actionPerformed(ActionEvent arg0) {
4    userApp.needHelpB.setEnabled(false);
5    AgentState oldState = agentState;
6    agentState = AgentState.Need_Help;
7    AgentState newState = agentState;
8    userApp.showLog("Agent state changed from " +
         oldState + " to " + newState + ".");
9    userApp.showState(newState);
10
11   String requesterID = getLocalName();
12   ACLMessage msg1 = new ACLMessage(ACLMessage.INFORM)
         ;
13   msg1.setOntology("NCHelpFF");
14   Organization org;
15
16   if(env.getTask("Aid1") == null){
17    org = env.getOrganization("RedCrescent");
18    msg1.addReceiver(new AID(org.getLocalName(), AID.
         ISLOCALNAME));
19    // Tasks
20    Task Aid1 = new Aid1("Aid1", env, new File(
21       "D:/eclipse/workspace/ERECode/src/icons/tasks/
             Aid.jpg"),org);
22    try {
23     env.addTask("Aid1", Aid1, org.Location.x - 40,
           org.Location.y + 50);
24
25    } catch (IOException e1) {
26     e1.printStackTrace();
27    }
28    CallTasks.put("RedCrescent", Aid1);
29   }
```

Listing 3: A Partial Code of victimAgent1.java

By executing the program, first, a part of the Tehran geographical map is displayed. Figure 10 shows the icons used to represent the entities in the environment (Icon attribute which was determined during the modeling of each entity). Also, for each agent and organization in the environment, a user interface for user and system interaction is created.

Figure 11 shows a view of the application at the beginning of the execution. As shown in this figure, the NeedHelp option is only activated for agents 1 and 2 because the disaster has occurred in the sight radius of these agents (It was specified during the modeling).

When the user chooses the NeedHelp option, a help request is sent to Red Crescent, municipality, fire station, and police department (Figure 12). Upon receiving the message by the organizations, it is checked whether the relevant teams are available or not. Considering that all teams are available at the beginning of the program, the CanHelp option will be activated for all four organizations (Figure 13).

Firstly, by choosing the Can Help option by the Red Crescent and fire stations, some messages (as

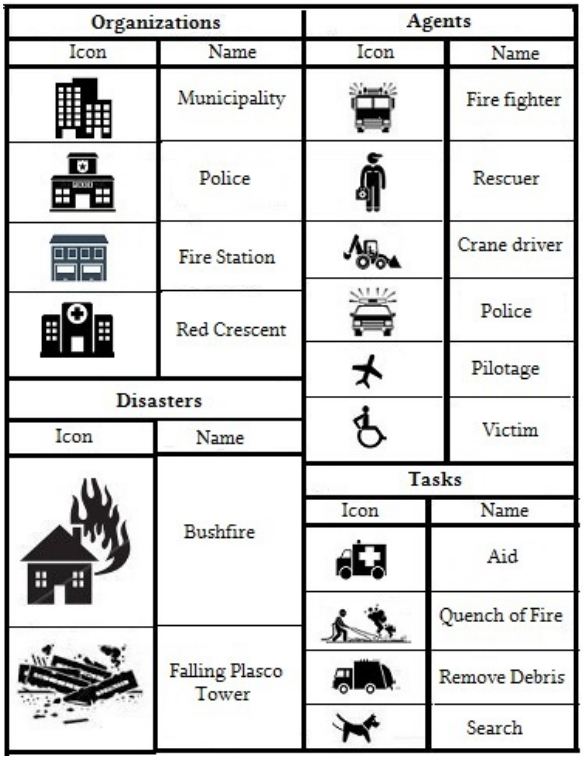

Figure 10. Icons Used for Visualizing the Plasco Tower Collapse Scenario

shown in Figure 14) are represented by the relevant organizations, and then the OrganizeTeam option is activated. It also sends a message to the requester agent for possible help from the organization, as shown in Figure 15.

By choosing the OrganizeTeam option, the team formations of respective organizations are shown. In Figure 16, a Firefighting team in the fire station and an emergency team in the Red Crescent organization are formed. The Firefighting team consists of two members: a firefighter and a pilotage. Also, the emergency team consists of two rescuers: rescuer1 and rescuer2. Each team formation makes the relevant messages to be represented to the organizations (Figure 17). Similarly, teams are formed by the municipality and police organizations for removing debris and searching.

After forming the teams, the AssignTask option is enabled. By choosing it, the tasks are assigned to the relevant teams (based on what has been designed during modeling). Here, two Quench of Fire and Aid tasks are assigned to the firefighting and emergency teams, respectively. The task assignments are shown in Figure 18.

After assigning tasks to each team, the PerformTask option becomes active. By choosing this option, the team's state changes instantly to Helping, and the CanHelp option becomes inactive. To perform tasks, the teams move toward the target (the entity that the

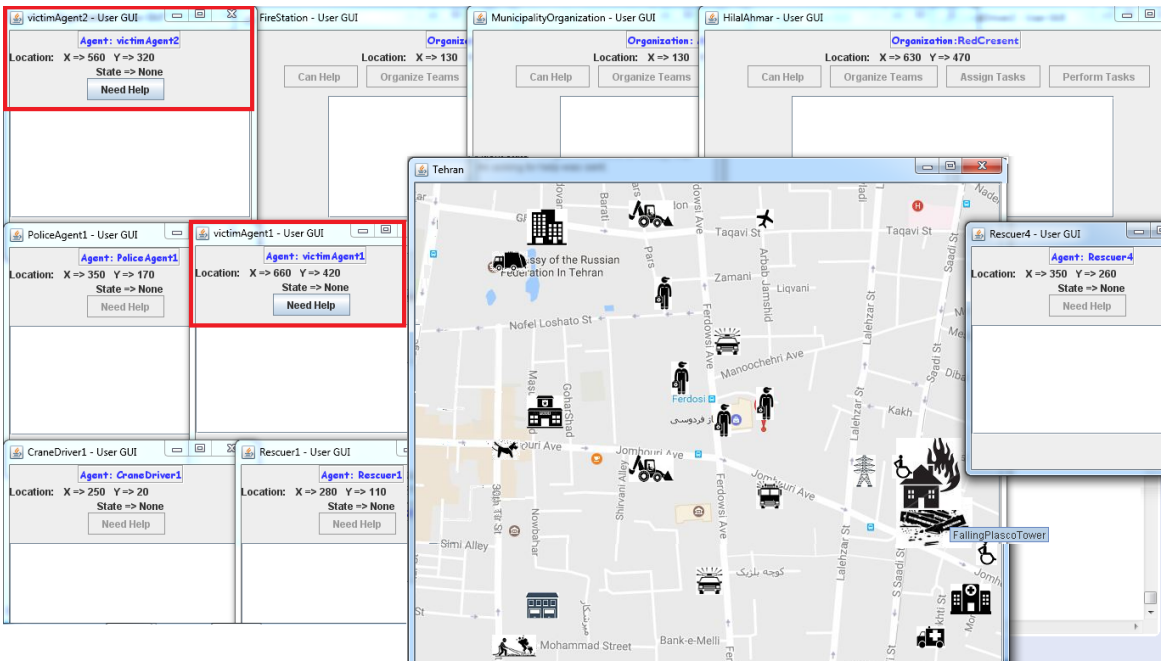

Figure 11. A View of the Generated Application

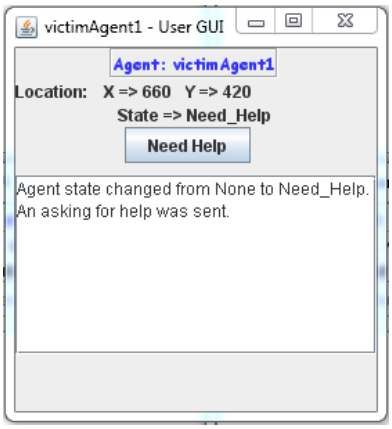

Figure 12. Victim Agent Sends Help Request

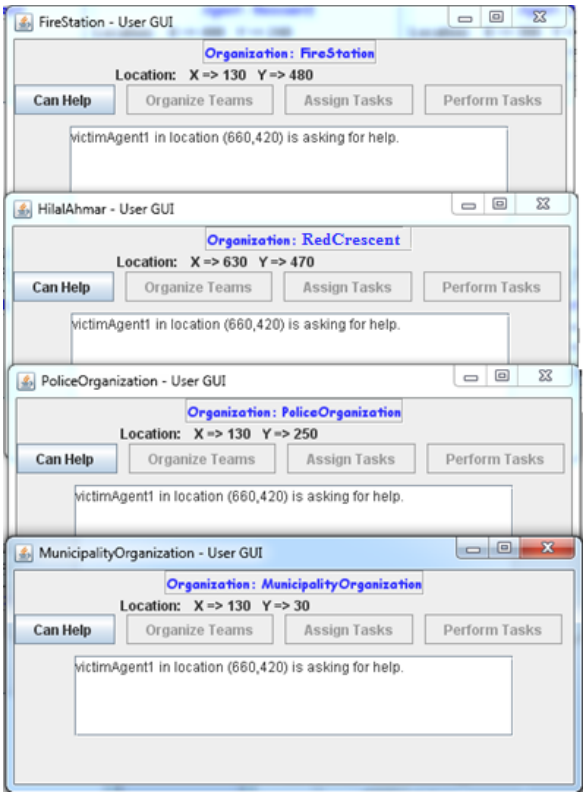

Figure 13. Organizations Receive Help Requests from Victim Agents

task affects it). According to the modeling, Quench of fire task will affect the bushfire and Aid task will affect the victim agent. Therefore the members of each team move toward the target location. By performing these tasks, the agent and disaster will be removed from the environment, and their icons are replaced by the task icons. It means that the firefighter and emergency teams have performed their tasks successfully. Finally, the state of the teams is changed to CanHelp again, and the state of the requester agents is changed to Being Helped. It also presents a message to the relevant organizations about successfully performing the tasks. Figure 19 shows a view of the program after performing Quench of Fire and Aid tasks.

6 Conclusion and Future work

Because of the urgency situations in an ERE and the need for a rapid and timely response, modeling and simulation of the agent's behavior in these environ-

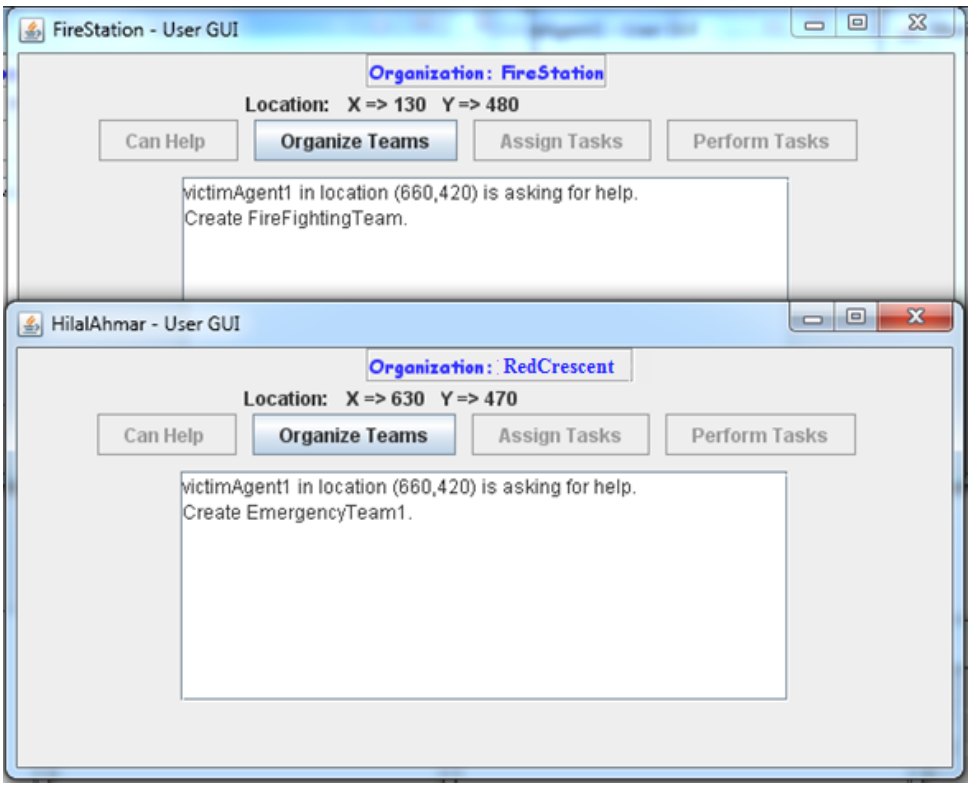

Figure 14. An Organization Forms an Emergency Team, After Receiving Help Request From an Agent

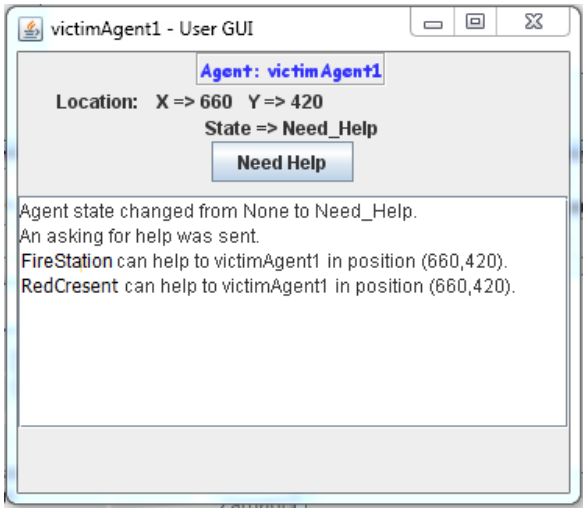

Figure 15. victimAgent1 Receives Help Possibility Messages From Organizations

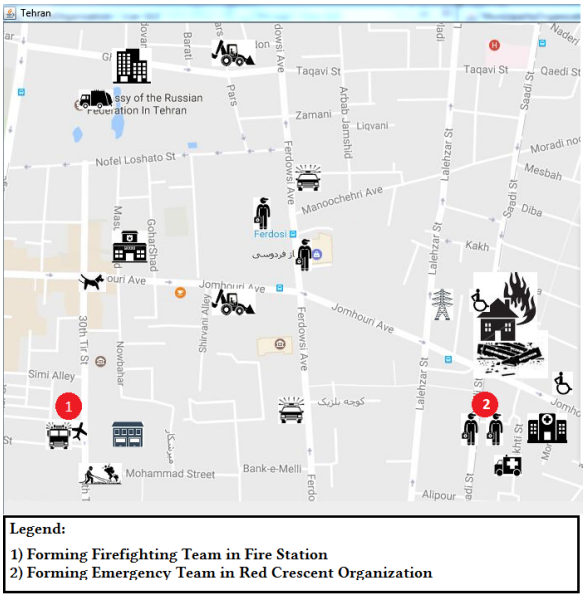

Figure 16. A View of the Generated Application After Creating Firefighting and Emergency Teams

ments are valuable to the crisis managers. The results of these simulations can be used to effectively carry out operations and responses during an emergency situation. In this regard, crisis managers need a tool to model an emergency response scenario and observe the results of their modeling as an executable program.

In this study, a model-driven framework named ERE-ML 2.0 was proposed to move toward simulating interactive EREs. This framework is an extension of our previously proposed framework named ERE-ML, which did not support interactions between the agents and organizations. To improve ERE-ML framework, the modeling language, the transformation code, and the platform was extended. To evaluate the ERE-ML 2.0 framework, the Plasco tower collapse scenario was modeled and validated as a case study. Afterward, the

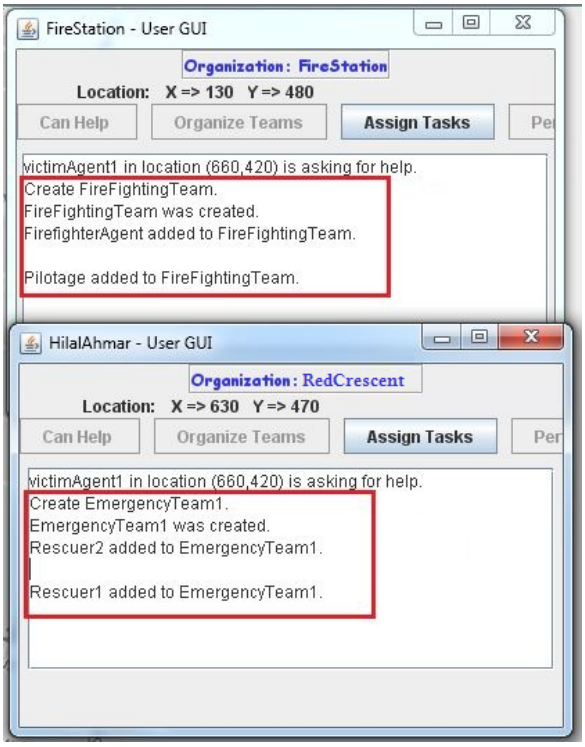

Figure 17. Messages About Firefighting and Emergency Team Formation

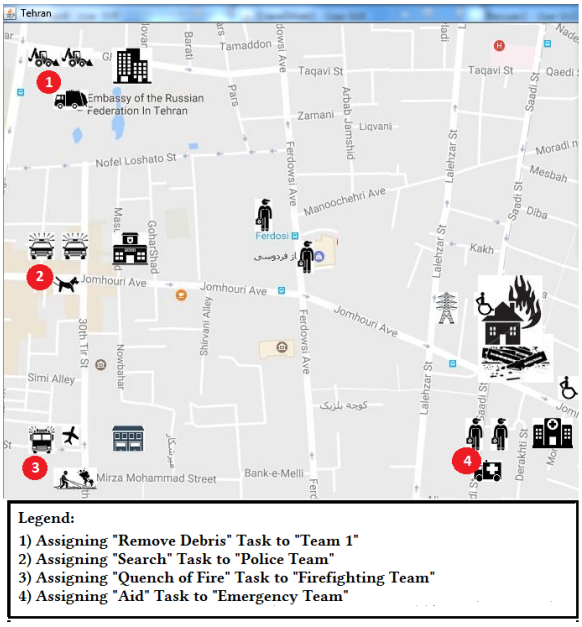

Figure 18. A View of the Generated Application After Assigning Tasks to Teams

designed model was automatically transformed to an executable Java code.

To enhance the ERE-ML 2.0 framework to simulate an interactive ERE, a suggestion is to make facilities for considering different policies and algorithms for team formation and task assignment. Furthermore, some important features, such as time, could be consid-

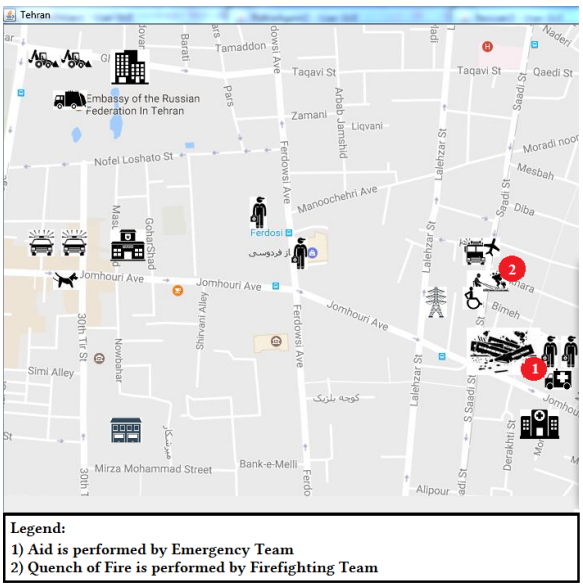

Figure 19. A View of the Generated Application After Performing Quench of Fire and Aid Tasks

ered to measure the success rate of each algorithm or policy. It can be more helpful for researchers or crisis managers to observe the effectiveness of different algorithms in different scenarios of emergency response and make better decisions. Also, adding capabilities such as learning and adaption to the environment for the agents in the modeling language can be helpful for modeling more realistic situations of EREs.

References

[1] Murray E Jennex. Modeling emergency response systems. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 22–22. IEEE, 2007.

[2] Morten Kyng, Esben Toftdahl Nielsen, and Margit Kristensen. Challenges in designing interactive systems for emergency response. In *Proceedings of the 6th conference on Designing Interactive systems*, pages 301–310. ACM, 2006.

[3] Qing Gu and David Mendonça. Patterns of group information seeking in a simulated emergency response environment. In *Proceedings of the 2nd international ISCRAM conference, Brussels, Belgium*, 2005.

[4] Christian Uhr, Henrik Johansson, and Lars Fredholm. Analysing emergency response systems. *Journal of Contingencies and Crisis Management*, 16(2):80–90, 2008.

[5] Tomoichi Takahashi, Ikuo Takeuchi, Tetsuhiko Koto, Satoshi Tadokoro, and Itsuki Noda. RoboCup-Rescue disaster simulator architecture. In *Robot Soccer World Cup*, pages 379–384.

Springer, 2000.

[6] Edouard Amouroux, Thanh-Quang Chu, Alain Boucher, and Alexis Drogoul. GAMA: an environment for implementing and running spatially explicit multi-agent simulations. In *Pacific Rim International Conference on Multi-Agents*, pages 359–371. Springer, 2007.

[7] Thanh-Quang Chu, Alain Boucher, Alexis Drogoul, Duc-An Vo, Hong-Phuong Nguyen, and Jean-Daniel Zucker. Interactive learning of expert criteria for rescue simulations. In *Pacific Rim International Conference on Multi-Agents*, pages 127–138. Springer, 2008.

[8] Venkatesh Mysore, Giuseppe Narzisi, and Bud Mishra. Agent modeling of a sarin attack in manhattan. In *Proceedings of the First International Workshop on Agent Technology for Disaster Management, ATDM*, pages 108–115, 2006.

[9] Giuseppe Narzisi, Joshua S Mincer, Silas Smith, and Bud Mishra. Resilience in the face of disaster: Accounting for varying disaster magnitudes, resource topologies, and (sub) population distributions in the plan c emergency planning tool. In *Holonic and multi-agent systems for manufacturing*, pages 433–446. Springer, 2007.

[10] Stephen Eubank. Scalable, efficient epidemiological simulation. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 139–145. ACM, 2002.

[11] Chris L Barrett, Stephen G Eubank, and James P Smith. If smallpox strikes Portland... *Scientific American*, 292(3):54–61, 2005.

[12] Khaled M Khalil, M Abdel-Aziz, Taymour T Nazmy, and Abdel-Badeeh M Salem. An agent-based modeling for pandemic influenza in Egypt. In *Handbook on Decision Making*, pages 205–218. Springer, 2012.

[13] Narjès Bellamine-Ben Saoud, Tarek Ben Mena, Julie Dugdale, Bernard Pavard, and Mohamed Ben Ahmed. Assessing large scale emergency rescue plans: an agent based approach. *The International Journal of Intelligent Control and Systems*, 11(4):260–271, 2006.

[14] Raffaele Giordano, Alessandro Pagano, Irene Pluchinotta, Rosa Olivo del Amo, Sonia M Hernandez, and Eduardo S Lafuente. Modelling the complexity of the network of interactions in flood emergency management: The Lorca flash flood case. *Environmental Modelling & Software*, 95: 180–195, 2017.

[15] Sanjay Jain and Charles McLean. Simulation for emergency response: a framework for modeling and simulation for emergency response. In *Proceedings of the 35th conference on Winter simulation: driving innovation*, pages 1068–1076. Winter Simulation Conference, 2003.

[16] Louise K Comfort, Yesim Sungu, David Johnson, and Mark Dunn. Complex systems in crisis: Anticipation and resilience in dynamic environments. *Journal of contingencies and crisis management*, 9(3):144–158, 2001.

[17] José M Gascueña, Elena Navarro, and Antonio Fernández-Caballero. Model-driven engineering techniques for the development of multi-agent systems. *Engineering Applications of Artificial Intelligence*, 25(1):159–173, 2012.

[18] Samaneh HoseinDoost, Tahereh Adamzadeh, Bahman Zamani, and Afsaneh Fatemi. A model-driven framework for developing multi-agent systems in emergency response environments. *Software & Systems Modeling*, pages 1–28, 2017.

[19] Daniel Amyot, Hanna Farah, and Jean-François Roy. Evaluation of development tools for domain-specific modeling languages. In *International Workshop on System Analysis and Modeling*, pages 183–197. Springer, 2006.

[20] Hamzeh Shakib, M Pirizadeh, S Dardaei, and M Zakersalehi. Technical and administrative assessment of Plasco building incident. *International Journal of Civil Engineering*, pages 1–13, 2018.

[21] Iván García-Magariño, Celia Gutiérrez, and Rubén Fuentes-Fernández. The INGENIAS development kit: A practical application for crisis-management. In *International Work-Conference on Artificial Neural Networks*, pages 537–544. Springer, 2009.

[22] Juan Pavón, Jorge J Gómez-Sanz, and Rubén Fuentes. The INGENIAS methodology and tools. In *Agent-oriented methodologies*, pages 236–276. IGI Global, 2005.

[23] Iván García-Magariño and Celia Gutiérrez. Agent-oriented modeling and development of a system for crisis management. *Expert Systems with Applications*, 40(16):6580–6592, 2013.

[24] Karam Mustapha, Hamid Mcheick, and Sehl Mellouli. Modeling and simulation agent-based of natural disaster complex systems. *Procedia Computer Science*, 21:148–155, 2013.

[25] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering*, 3(1):1–207, 2017.

[26] Viviane Torres Da Silva, Ricardo Choren, and Carlos JP De Lucena. MAS-ML: a multiagent system modelling language. *International Journal of Agent-Oriented Software Engineering*, 2(4): 382–421, 2008.

[27] Siti Hajar Othman and Ghassan Beydoun. Model-driven disaster management. *Information & Management*, 50(5):218–228, 2013.

[28] Object Management Group. Unified Modeling

Language (UML) - Version 2.5. http://www.omg.org/spec/UML/2.5/, Date Accessed: March 26, 2017.
[29] Object Management Group. Object Constraint Language - Version 2.4. http://www.omg.org/spec/OCL/2.4/, Date Accessed: March 26, 2017.
[30] Eclipse.org. Eclipse Model To Text (M2T) Project. http://www.eclipse.org/acceleo/, Date Accessed: February 4, 2017.
[31] Object Management Group. MOF Model to Text Transformation Language - Version 1.0. http://www.omg.org/spec/MOFM2T/1.0/, Date Accessed: March 4, 2017.

Samaneh Hoseindoost is a Ph.D. student at University of Isfahan. She received her B.Sc. in computer software engineering from University of Isfahan in September 2014. Then she studied at the M.Sc. degree in Software Engineering at the same University and graduated in April 2017. Her interests include Model Driven Software Engineering, Multi-Agent Systems, and crisis management systems. She is a member of Model-Driven Software Engineering Research Group (MDSERG) at University of Isfahan.

Afsaneh Fatemi received her B.Sc. from the Isfahan University of Technology, Isfahan, Iran, in 1995, and her M.Sc. and Ph.D from the University of Isfahan, Isfahan, Iran in 2001 and 2012, all in Computer Engineering (Software). Dr. Fatemi joined the Faculty of Computer Engineering at the University of Isfahan in 2013, as an assistant professor. Her main research interests lie in the fields of Complex Networks, Multi-Agent Systems, and Emergency-Response Systems' Modeling. She as been collaborating with the MDSE research group from 2014.

Bahman Zamani received his B.Sc. from the University of Isfahan, Isfahan, Iran, in 1991, and his M.Sc. from the Sharif University of Technology, Tehran, Iran in 1997, both in Computer Engineering (Software). He obtained his Ph.D. degree in Computer Science from Concordia University, Montreal, QC, Canada in 2009.
From 1998 to 2003, he was a researcher and faculty member of the Iranian Research Organization for Science and Technology (IROST) - Isfahan branch. Dr. Zamani joined the Faculty of Computer Engineering at the University of Isfahan in 2009, as an assistant professor. His main research interest is Model-Driven Software Engineering (MDSE). He is the founder and director of MDSE research group at the University of Isfahan.