Journal of Computing and Security

Research Article, 2026, Volume 13, Number 2 (pp. 23–48)

http://www.jcomsec.org

A Model-Driven Framework for Automatic Generation of Auction-Based Coordination Strategies for Crisis Response

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

[a] *MDSE Research Group, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: 12 December 2025
Revised: 24 February 2026
Accepted: 03 May 2026
Published Online: 03 August 2026

Keywords:
Auction-Based Strategies, Crisis Management, Task Allocation, Model-Driven Engineering, Domain-Specific Modeling Language.

ABSTRACT

Coordination strategies for crisis response mainly deal with task allocation. Auction-based coordination strategies are one of the most common strategies for task allocation in Emergency Response Environments (ERE), i.e., the environments in which the crisis occurs. Due to the emergency in EREs, coordination must be performed efficiently. To this end, before a crisis occurs, crisis managers try to benefit from modeling and simulation of coordination strategies. However, programming for the simulation of different coordination strategies is a difficult and time-consuming task for domain experts who usually do not have enough programming skills. To overcome this issue, in this paper, we propose a model-driven framework for modeling and automatic generation of auction-based coordination strategies for crisis response. The proposed framework consists of two main parts: (1) a Domain-Specific Modeling Language (DSML), named Auction-ML, with tool support for designing auction-based coordination strategies at a high level of abstraction, and (2) a transformation engine for automatic code generation from designed models. The generated code can be executed on ERE models to generate the output model that shows team formations, task allocations, resource allocation, and strategy performance. To show the applicability of the proposed framework, two inter- and intra-organizational auction-based coordination strategies are modeled using AuctionML. Then, the performance of the strategies is measured by the automatic generation of coordination strategies and execution of the strategies on the ERE models. The results confirm the applicability of the framework for enabling non-programmers to model, simulate, and evaluate auction-based strategies.

1 Introduction

Due to the urgency of the situation in Emergency Response Environments (ERE) and the limited time for quick and timely response, crisis management is an important issue that must be performed efficiently [1]. In general, crisis management includes four phases: mitigation, preparedness, response, and recovery [2].

* 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.2026.145977.1190
ISSN: 2322-4460

The mitigation and preparedness phases take place before the crisis. The response phase happens during the crisis, and the recovery phase is performed after the crisis. The mitigation phase aims to undertake necessary measures to prevent crises or minimize their adverse consequences. In the preparedness phase, the focus is on gathering information, planning, organizing, training, and supplying resources to facilitate effective responses during a crisis. During the response phase, emergency actions and services are executed with the primary goal of preserving lives and safeguarding property. In the fourth phase of crisis management, known as recovery, essential actions are taken to restore affected areas to their pre-crisis state and to address the physical, mental, and social well-being of those affected.

Since the purpose of the preparedness phase is to plan for effective responses during a crisis, one of the methods that can assist crisis managers in decision-making is modeling and simulating EREs. These methods help crisis managers effectively coordinate available resources (including equipment and human resources) to minimize casualties and reduce operation time [3]. Crisis management involves engaging individuals from various organizations (including police, Red Crescent, hospitals, and municipalities). As crises escalate, the number of involved organizations increases. Consequently, EREs consist of heterogeneous and autonomous organizations, each assigned specific tasks. These tasks are interconnected, necessitating constant interaction and collaboration among organizations, particularly at higher levels dealing with crises [4].

For example, consider a large-scale earthquake affecting a metropolitan city such as Tehran, which consists of multiple densely populated districts. Following the disaster, several critical response tasks must be carried out, including search and rescue operations in severely damaged areas. Due to differences in damage severity, population distribution, and available resources, these tasks have different priorities and resource requirements. Multiple rescue organizations are involved in the response, each operating autonomously with limited human and non-human resources. Under such conditions, coordinating an effective response requires appropriate mechanisms for allocating tasks and resources among organizations, while accounting for competing priorities, operational constraints, and time pressure.

In the response phase of crisis management, coordination primarily revolves around task allocation [5, 6]. Cooperation for task allocation is a very challenging research issue in disaster environments [7]. Various strategies have been proposed for task allocation within EREs. Among the most common strategies are auction-based, consensus-based, and optimization-based approaches [8]. Given the widespread applicability and popularity of auction-based coordination strategies in the domain of crisis management [8, 9], this research focuses primarily on this category of coordination strategies.

Task allocation in EREs must be performed efficiently. Using ineffective strategies for task allocation during crisis response includes high risks and may lead to severe damages, losses, and financial costs. To address this concern, crisis managers prefer to model and simulate the EREs and task allocation strategies to be able to evaluate them before a crisis occurs. Modeling and simulation assist crisis managers in the efficient allocation of available resources (including equipment and human resources) to minimize the number of casualties and reduce the time of operations [3].

Building simulations for various coordination strategies poses a challenge for domain experts, including crisis managers, strategists, and urban planning managers [10, 11]. To address this challenge, using Model-Driven Engineering (MDE) and Domain-Specific Modeling Languages (DSML) could be beneficial. These approaches simplify the implementation of such systems by raising the level of abstraction and bringing modeling concepts closer to the user's specialized domain. Consequently, productivity is enhanced by reducing the time and effort required for development [10, 12]. For this purpose, DSMLs are used to model coordination strategies, allowing crisis managers to design and evaluate systems without delving into implementation details. Instead, crisis managers can focus solely on designing their desired coordination strategies.

Until now, numerous software engineering modeling languages have been proposed to describe the behavioral aspects of algorithms and systems. Among them are Sequence Diagrams, Activity Diagrams, and Flow Charts. Although these languages are expressive and widely used, their general-purpose nature and the absence of domain-specific concepts make them complex and time-consuming for domain experts in crisis management to use. This is because modeling each part of a coordination strategy using these languages requires handling numerous nodes and edges with multiple features together to represent the algorithm associated with the desired strategy.

In summary, the main problem addressed in this research originates from the necessity of employing efficient strategies at the time of crisis occurrence and the difficulty in programming by domain experts to simulate coordination strategies. This leads to a main

problem, i.e., the lack of a specific language for modeling coordination strategies and generating the corresponding code automatically.

To address the problem, in this research, a model-driven framework has been proposed including two domain-specific languages:

(1) CoordinationML as a foundational modeling language for coordination strategies
(2) AuctionML as an extension of CoordinationML for modeling auction-based coordination strategies at a higher level of abstraction than programming languages

In addition, a transformation engine has been developed for automatic generation of code for the strategies from the AuctionML models.

To demonstrate the applicability of the proposed framework, two auction-based strategies for inter- and intra-organizational coordination were modeled using the AuctionML language. Subsequently, the Java code related to the strategies was generated, and the performance of the coordination strategies was measured. To show the performance of the strategies, the strategies were executed on ERE models designed by CoorERE [13]. CoorERE is an executable DSML with tool support for the graphical representation of the entities that are involved in an ERE, including organizations, tasks, resources, and persons. The code that is generated from AuctionML models will be the operational semantics of CoorERE. In our previous work [13], the semantics were implemented by programming. In this paper, the strategies will be modeled at a higher level of abstraction, and the code of the strategies will be generated automatically.

The main contributions of this paper are summarized as follows:

- Introducing AuctionML, a DSML that enables domain experts to specify auction-based coordination strategies at a higher level of abstraction than programming languages.
- Designing a metamodel-driven architecture that explicitly models coordination semantics and performance evaluation through dedicated meta-classes, thereby enabling extensibility and domain-specific customization of evaluation metrics.
- Developing an automated transformation engine that generates executable code directly from AuctionML models, eliminating the need for manual programming of coordination strategies.
- Demonstrating the applicability and practicality of the framework through inter- and intra-organizational case studies and integrating the generated code as the operational semantics of CoorERE.

In contrast to our previous work [13], where coordination semantics were implemented manually through programming, this paper provides a fully model-driven solution with automatic code generation.

The rest of this paper is structured as follows. In Section 2, the research background is reviewed. Section 3 presents the related work in comparison with the current research. Section 4 introduces the proposed framework for modeling and code generation of auction-based coordination strategies. Section 5 describes the evaluation of the proposed framework using the case study technique. Section 6 discusses the advantages and limitations of this research. Finally, Section 7 presents the conclusion and outlines the future works.

2 Background

This section reviews the research background that helps better understand the solution that is provided in this paper. Since the scope of these studies is limited to auction-based coordination strategies, in Section 2.1, these strategies are introduced. The proposed framework is based on the MDE concepts; therefore, in Section 2.2, MDE concepts are reviewed.

2.1 Auction-Based Coordination Strategies

In this section, the principles of auction-based coordination strategies are introduced. Generally, there are four common stages in an auction-based strategy [8, 14].

(1) Announcement: In this phase, one or a set of tasks are announced to a group of agents [14].
(2) Bidding Strategy: Individual agents that receive the announcement calculate the individual utility values or cost based on the objective/cost function and communicate this value to the auctioneer [14]. To this end, each agent sends a message to the auctioneer and informs him about the cost that he incurs for doing the task [8]. In the single auction mechanism, bidders can place bids on single items; in the combinatorial mechanism, the bidders can place bids on combinations of items [15].
(3) Winner Determination: After receiving all the bids from the agents, the job of the auctioneer is to determine the winning agent by assessing the received bids based on an optimization strategy [14].
(4) Contract: The task is then assigned to the winning agent for execution.

Different auction algorithms can be classified based

on the number of rounds, the number of items auctioned in each round, the number of items sold in each round, and the number of bids made by bidders for those items [16]. Accordingly, auction algorithms are categorized into six categories, as described in the following [17].

- In sequential auctions, for each item, a four-step auction process is executed. Therefore, there are m different auction rounds, and each participating agent submits one bid in each round.
- In parallel auctions, m items are auctioned simultaneously and only in one round. Upon receiving the announcement message, each participating agent immediately sends a message of length m containing the proposed prices for each item to the auctioneer.
- In the repeated parallel auction type, instead of a total of m items, n items are offered and sold in each round. Hence, $\lceil m/n \rceil$ rounds are needed for all items to be sold. Each agent can only win one item in each round, although they may win in multiple rounds and undertake multiple tasks.
- In the G-Prim algorithm, the auctioneer puts all unsold items up for auction in each round, and participating agents only submit bids for their best choice in each round.
- In the repeated G-Prim auction, participating agents bid for their top n choices in each round, not just their best choice. Therefore, n items are sold in each round, and $\lceil m/n \rceil$ rounds are necessary to sell all items.
- In combinatorial auctions, similar to parallel auctions, only one auction round is held, but each agent can bid for all possible combinations of items.

2.2 Model-Driven Software Engineering

Model-driven software engineering (MDSE) is a promising approach to cope with the complexity of software systems. Using this approach, a user can model the system at a high-level abstraction [10]. In MDSE, the main concepts are models and transformations [18]. In this approach, models are considered first-class citizens, and the development is driven by modeling artifacts [19]. A model is an abstract representation of a system that is consumed and processed more easily than documents. Generally, models are used by experts to understand and reason about a system, to communicate and argue with other actors, and as a support for decision-making [10]. Models are managed by modeling languages. A modeling language, also known as a metamodel, is a model that defines a language to specify conforming models [20]. A modeling language consists of three main aspects as follows [10].

(1) **Abstract syntax:** Describing the structure of the language, i.e., the key concepts and their relationships. This is created by using metamodeling languages (i.e., meta-metamodels), such as the standard MOF or Ecore [20].
(2) **Concrete syntax:** Describing specific representations of the modeling language, covering encoding and/or visual appearance issues. The concrete syntax can be either textual or graphical. Designers often use concrete syntax as a reference when engaging in modeling activities. When using a visual syntax, the output may consist of one or more diagrams.
(3) **Semantics:** Describing the meaning of the elements defined in the language and the meaning of the different ways of combining them. The semantics of a modeling language can be defined as static and dynamic semantics [21]. Static semantics are the constraint properties that are defined by using a (constraint) property modeling language, e.g., Object Constraint Language (OCL) [22]. Dynamic semantics (or executable semantics) define the behavior of a language. Dynamic semantics is used for model execution, interpretation, compilation (transformation), and simulation.

Modeling languages can be divided into two categories in terms of their scope of application: General Purpose Modeling Languages (GPMLs) and Domain-Specific Modeling Languages (DSMLs) [10]. GPMLs are used to model any type of system and address any problem, e.g., UML [23] and Petri-net [24], while DSMLs are used to model a specific system and address a specific problem. For instance, MAS-ML [25] is a language for modeling Multi-agent Systems (MAS), and ERE-ML [26, 27] is a language for modeling MAS in ERE.

A transformation is essentially a program defined by using a transformation language. These languages can be used to define transformation patterns [10]. For example, the MTL transformation language [28] is used to write model-to-code transformations. To use the Acceleo transformation language, a tool called Acceleo [29] has been developed as an Eclipse plugin. The workflow of this tool is as follows: it takes the input model along with the transformation pattern written in MTL language as input and generates the corresponding code as output.

It is worth mentioning that automatic code generation is one of the main features of MDE, distinguishing it from traditional software development methods. In model-to-code transformations, all or part of the code is automatically generated from the model. Many modern modeling tools can generate

code skeletons from the model. However, the ultimate goal of MDE is to achieve 100% automatic code generation [10].

3 Related Works

Modeling coordination in software systems has been extensively investigated using formal methods, organizational models, agent-oriented approaches, and model-driven techniques [30–32]. However, despite this extensive body of work, only a limited number of studies address coordination modeling at a high level of abstraction while remaining semantically close to domain concepts in EREs.

This section reviews prior work in two parts: Section 3.1 examines coordination modeling approaches within the ERE domain, while Section 3.2 discusses frameworks proposed for other environments. The discussion is structured analytically to highlight conceptual, methodological, and execution-level differences from the proposed framework. Section 3.3 then explicitly articulates the limitations of existing frameworks and establishes the distinctive contribution of our work.

3.1 Modeling Coordination Strategies for Crisis Response

Existing research on coordination modeling in EREs can be broadly categorized into three groups:

(1) **Knowledge-sharing based coordination frameworks.** Truptil and Lauras et al. [33, 34] proposed a UML/OWL-based metamodel to facilitate interoperability between crisis management information systems. Coordination is primarily achieved through knowledge sharing among actors. Although their approach provides structural modeling support and tool assistance, coordination strategies such as task allocation or team formation are not explicitly modeled as domain-level constructs.

(2) **Temporal or activity-dependency modeling approaches.** Franke et al. [35] introduced a framework for temporal coordination in dynamic crisis scenarios. Activities and their dependencies are modeled using activity diagrams and state machines. While the framework supports inter-organizational coordination and provides tool support for managing state conflicts, it relies on GPMLs and does not introduce domain-specific abstractions such as *Agents*, *Tasks*, or auction-based allocation strategies. Furthermore, the models are not executable in a way that enables systematic performance evaluation of coordination strategies.

(3) **Interaction-pattern and simulation-integration approaches.** Le et al. [36] combined business process modeling with multi-agent models and used Petri nets for coordination design. Although expressive, Petri-net concepts are not semantically aligned with ERE domain abstractions, making them less accessible to domain experts. Moreover, coordination logic is manually specified and not represented as a first-class domain-specific construct.

Xiong Li et al. [37] proposed the Agent Action Diagram (AAD) to bridge conceptual and simulation models in crisis management. Their approach integrates model-driven architecture and automated transformation toward simulation models. While this work shares similarities with the present study in terms of model transformation, its primary focus is on modeling interaction patterns among agents (e.g., Commander-Broker patterns). Explicit modeling of task allocation algorithms as domain-level coordination strategies is not the focus of that work.

Synthesis. As reflected in Table 1, although existing ERE-related approaches provide modeling support, tool assistance, or transformation mechanisms, none of them explicitly model auction-based task allocation algorithms as domain-specific, executable constructs at the metamodel level. In most cases, coordination is either knowledge-driven, interaction-pattern-based, or manually implemented, rather than formally specified and automatically executed within a dedicated DSML tailored to ERE.

3.2 Coordination Modeling in Other Environments

Coordination modeling has also been studied extensively in non-ERE domains. These works typically focus on organizational processes, general multi-agent systems, or service-oriented architectures.

Vreede and Eijck [38] proposed conceptual viewpoints for modeling coordination in organizational settings and provided simulation support. However, their approach is model-based rather than model-driven; transformations to executable artifacts are performed manually. Moreover, it is not tailored to ERE or crisis response contexts.

Van Aart [39] introduced a capability-based conceptual framework for modeling coordination in MAS. While the framework provides rich abstractions for reasoning about agent capabilities, it operates in a static context and does not support domain-specific task allocation modeling for crisis scenarios.

Sudeikat and Renz [40] developed a DSML for

Table 1. Comparison of the current study with related work in the ERE domain.

#	Reference	ERE domain	Coordination through		Modeling language		Explicit task allocation modeling	Tool Support	Performance measurement
			Task allocation	Other methods	General Purpose	Domain Specific			
1	Truptil, Lauras et. al [33, 34]	+	–	+	+	–	–	+	–
2	Franke et. al [35]	+	+	–	+	–	–	+	–
3	Le et. al [36]	+	–	+	+	–	–	+	+
4	Xiong Li et al [37]	+	–	+	–	+	–	+	+
5	The proposed framework	+	+	–	–	+	+	+	+
Legend: + Supported – Not Supported									

application-independent inter-agent coordination patterns. Their approach separates coordination strategies from implementation details and supports design-level abstraction. Nevertheless, it does not target ERE and does not address auction-based coordination strategies

Nieves et al. [41] proposed a multi-layer architecture integrating organizational theories with model-driven development in service-oriented systems. Although automated transformations and tool support are provided, coordination is treated at a general architectural level rather than as an executable, domain-specific strategy tailored to emergency response.

Synthesis. As explicitly indicated in Table 2 through the "ERE domain" dimension, the frameworks are designed for general-purpose coordination modeling and are not tailored to the ERE. While they provide valuable abstractions, DSML support, or automated transformations, they do not embed ERE-specific coordination concepts such as task allocation strategies directly into the metamodel. Consequently, they do not provide an executable, domain-specific coordination modeling framework for ERE.

3.3 Identified Research Gap

The analysis of existing studies reveals three main limitations:

(1) **Lack of explicit modeling of task allocation algorithms** as first-class, domain-specific constructs within ERE-focused frameworks.
(2) **Reliance on general-purpose or pattern-based coordination mechanisms,** rather than formalized, executable auction-based strategies modeled at the metamodel level.
(3) **Absence of an integrated model-driven pipeline** that simultaneously supports domain-specific modeling, automated transformation into executable artifacts, and systematic performance evaluation of coordination strategies in ERE.

The proposed framework addresses these limitations by introducing a DSML tailored to ERE, explicitly modeling auction-based task allocation strategies at the metamodel level, and enabling their automated execution and performance evaluation within a unified model-driven environment.

4 The Proposed Framework

In this section, we present the proposed framework for the automatic generation of auction-based coordination strategies for crisis response. The framework is designed based on the principles of MDSE [10] and is developed through a systematic and incremental process. As illustrated in Figure 1, the development process begins with a domain analysis phase. In this phase, the requirements of general coordination strategies are analyzed to identify common and predominant concepts. Based on this analysis, a requirement specification for coordination strategies is defined, leading to the design and development of CoordinationML, a DSML that captures the essential concepts of coordination.

Subsequently, a separate requirement specification focusing on auction-based coordination strategies is conducted. Since the completeness and quality of automatic code generation strongly depend on how closely the modeling language concepts align with the coordination process, AuctionML is introduced as an extension of CoordinationML to incorporate auction-specific concepts. This extension enables more precise modeling of auction-based coordination mechanisms.

To support the practical use of AuctionML, a modeling tool consisting of graphical and textual editors is developed. In parallel, model-to-code transformations are defined to enable automatic generation of executable coordination strategies. These transformations specify how each element of AuctionML is mapped into executable code. Finally, the generated components including the modeling tool and the transformation engine are integrated into a unified

Table 2. Comparison of the current study with related work in non-ERE domains.

#	Reference	ERE domain	Coordination through		Modeling language		Tool Support	Performance measurement
			Task allocation	Other methods	General Purpose	Domain Specific		
1	Vreede and Eijck [38]	–	–	+	–	+	–	–
2	Van Aart [39]	–	–	+	–	+	–	+
3	Sudeikat and Renz [40]	–	+	–	–	+	+	+
4	Nieves et al. [41]	–	+	–	–	+	+	–
5	The proposed framework	+	+	–	–	+	+	+
Legend: + Supported – Not Supported								

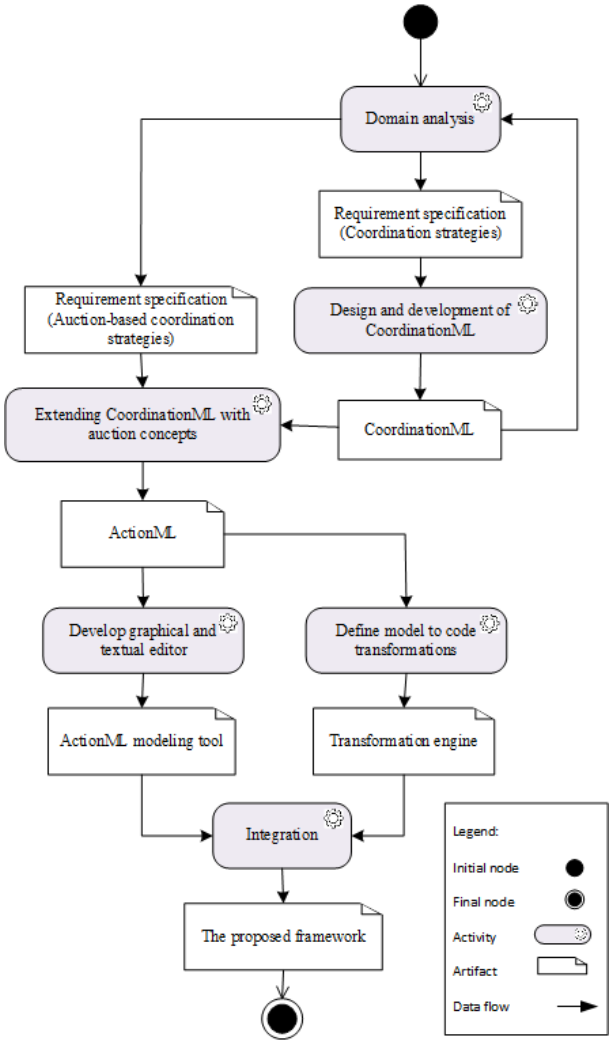


Figure 1. Process of design and development of the proposed framework.

framework.

The overall architecture and execution flow of the proposed framework are shown in Figure 2. First, a coordination strategy model is created using AuctionML. The defined model conforms to the corresponding metamodel and serves as the input for the

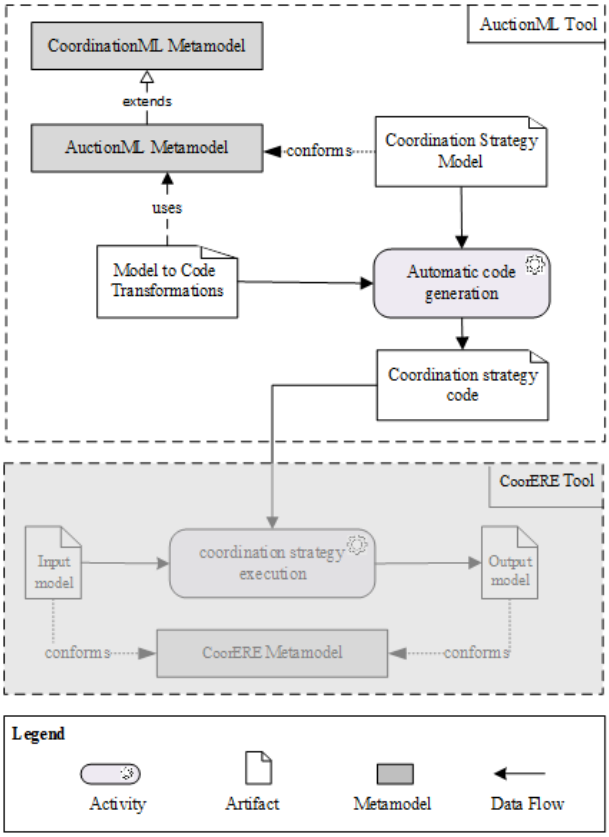


Figure 2. The architecture of the proposed framework.

transformation engine. By executing the model-to-code transformations, the coordination strategy Java code is automatically generated.

The generated code is designed to be executable on the target platform, which in this study is the CoorERE tool [13]. Using the facilities provided by CoorERE, the generated coordination strategy is executed on the input model to produce the output model. The input model represents the entities involved in an ERE, including organizations, tasks, resources, and persons. The output model reflects the results of strategy execution, such as task allocation to organizations, resource allocation, team formation,

and overall coordination strategy performance.

Then, according to the principle that the more closely the concepts in the modeling language align with the coordination strategy steps, the more complete code generation will be, AuctionML is presented as an extension of CoordinationML for modeling auction-based coordination strategies. To use AuctionML, a modeling tool consisting of graphical and textual editors for modeling the coordination steps has been developed.

To automatically generate code for coordination strategies, model-to-code transformations are required. To write these transformations, elements from the AuctionML are used, specifying how each of these elements should be transformed into executable code.

The components of the framework will be introduced as follows. In Section 4.1, the CoordinationML along with the AuctionML are presented. The graphical and textual editors are discussed in Section 4.2. In Section 4.3, model-to-code transformations are elaborated. Finally, Section 4.4 describes how end-users utilize the framework.

4.1 The Modeling Languages

In this paper, two modeling languages are presented: CoordinationML and AuctionML. To create the languages, first, the requirements and concepts of the coordination strategies need to be extracted by reviewing the literature and research background. These requirements are presented in Section 4.1.1. Then CoordinationML and AuctionML are explained in Sections 4.1.2 and 4.1.3 respectively.

4.1.1 Requirements Specification

We extracted nine requirements (R1 to R9) for the CoordinationML modeling language [5, 6, 16, 37, 42–45], and eight extra requirements (R10 to R17) that are specific to the AuctionML language [5, 8, 16, 46, 47].

(A) Requirements of CoordinationML

- R1: Modeling coordination strategy steps with the ability to extend it for specific strategies
- R2: Modeling the execution order of steps
- R3: Modeling control structures such as conditions and loops
- R4: Modeling mathematical and logical expressions with consideration for operator precedence
- R5: Ability to define auxiliary variables of various types
- R6: Modeling mathematical functions independently of coordination strategies
- R7: Modeling suspended or waiting states in a process
- R8: Modeling sorting of elements based on a specific criterion in ascending or descending order
- R9: Modeling performance measurement criteria for coordination strategies

(B) Extra requirements for AuctionML

- R10: Modeling the four stages of auction-based coordination strategy, including announcement, bidding, winner determination, and contract (resource allocation)
- R11: Modeling different states in the announcement stage, such as announcing all items, the top few items, or only one item per round
- R12: Defining different cost or valuation functions in the bidding stage
- R13: Modeling different states for agents to submit bids, such as bidding for all items or only the best or least expensive item
- R14: Modeling the winner determination strategy based on a specific criterion (e.g., the agent with the lowest cost or highest value offered)
- R15: Modeling functions related to resource allocation for task execution to organizations
- R16: Modeling the announcement of the winning individual to all participants
- R17: Modeling the sending of acknowledgment messages by the winning agent

4.1.2 CoordinationML

Figure 3 illustrates the CoordinationML metamodel for modeling and evaluating coordination strategies. This metamodel includes 16 meta-classes as follows.

The root of the metamodel, *CoordinationStrategy*, should have its name property set. Each *CoordinationStrategy* consists of several nodes and transitions. Each node is an abstract meta-class that can be of type *ControlNode* or *Step* in the coordination strategy. Transitions are used to move from one step to another. Each transition has exactly one source and one destination node. *ControlNodes* in this metamodel are divided into two categories: *DecisionNode* and *Loop*.

- **Decision Node:** The structure of each decision node is defined as *if* `<conditional expression>` *then* `<Node>` *else* `<Node>`. Therefore, this node includes a conditional expression in *EString* type, which must conform to the logical expressions' grammar provided in Appx. B.
- **Loop:** Loops are used to iterate over a single or a set of steps. The number of iterations in a loop can be implemented in three ways: (1) until reaching a specific number, (2) until a specific condition is met, or (3) based on the number of

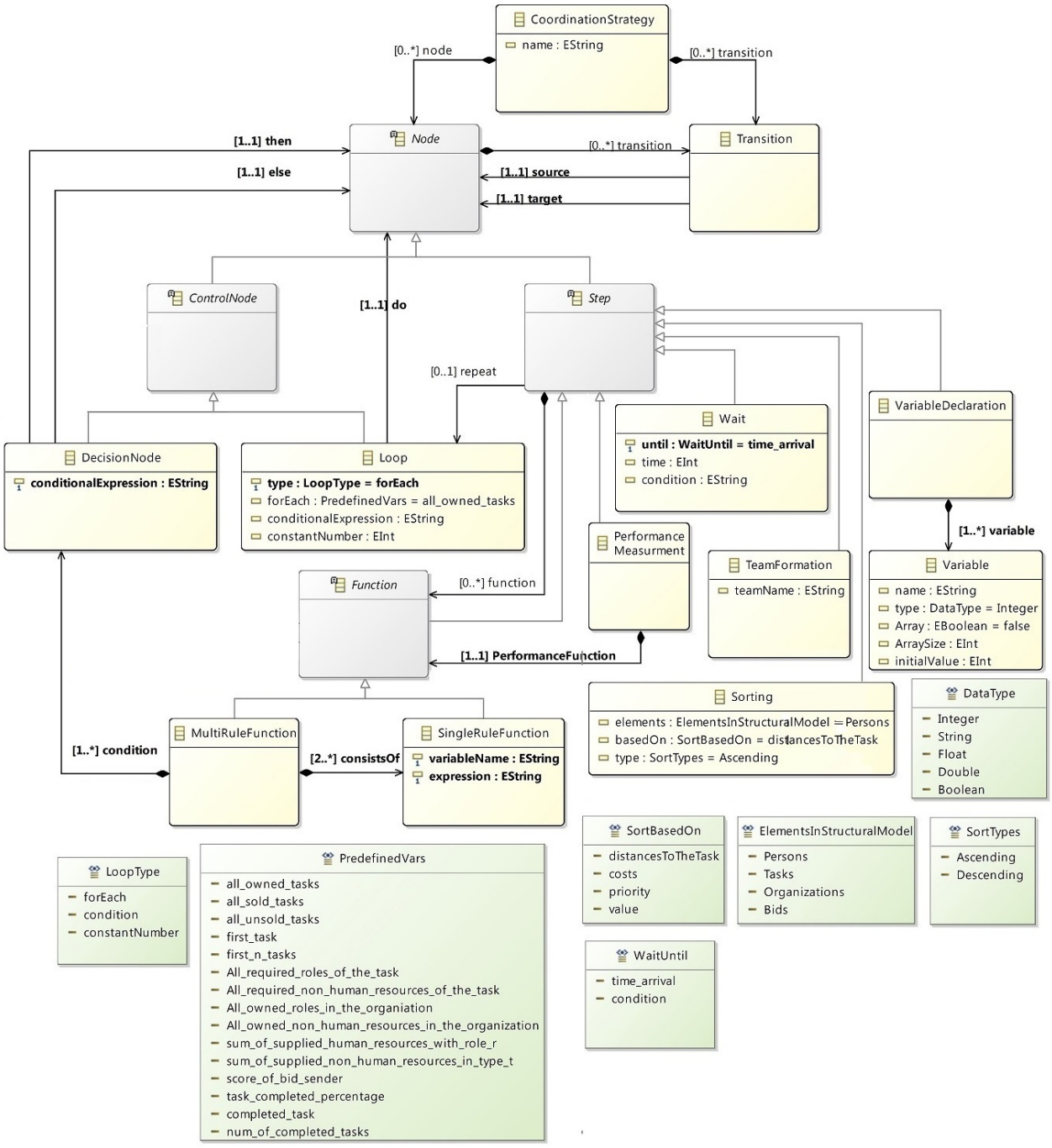


Figure 3. The CoordinationML Metamodel.

elements in the structural model. These elements are defined as an enumeration called *PredefinedVars*.

The *Step* is an abstract and extensible superclass (expanded for implementing specific concepts of auction-based coordination strategies in Section 4.1.3). Each step in a coordination strategy corresponds to an operation in the coordination algorithm. By studying the domain of coordination strategies, the most common shared operations among different types of coordination strategies are identified and incorporated into this model as steps. These include *Wait*, *TeamFormation*, *Sorting*, *Function*, *VariableDeclaration* and *PerformanceMeasurement*, which are explained as follows.

- **Wait:** This concept can be used when agents in the system must synchronize with each other at a certain time. The wait duration can be a specific number of seconds or until a specific condition is met. This condition follows the grammar of

logical expressions.

- **TeamFormation:** In many coordination strategies for crisis response, there is a team formation process. Using the *TeamFormation* element in the coordination model is equivalent to creating an instance of the Team class in the structural model at runtime. The process of adding members to this team varies depending on different strategies and can be modeled either by coding or by extending the framework for a specific type of strategy. In this study, the team formation process in auction-based coordination strategies is modeled by extending the framework.
- **Sorting:** Sorting operations can be performed for tasks based on their priority, for organizations or individuals based on their distance from the task, or for bids based on their cost or value. Sorting can also be done in ascending or descending order.
- **Function:** In the proposed metamodel, a function is an abstract concept that can be implemented as a single-rule function or a multi-rule function.
  - A single-rule function follows the pattern *"variableName = expression"* where the expression is written according to the mathematical expression grammar (explained in Appx. A).
  - A multi-rule function is a function composed of at least one condition and two single-rule functions.
- **VariableDeclaration:** Using this meta-class, one can define several variables. For each variable, properties such as name, type, array or non-array, array size, and initial value can be specified. The data types that can be defined for each variable are listed in an enumeration called *DataType*, including integers, floating-point numbers, booleans, and strings.
- **PerformanceMeasurement:** In the proposed metamodel, efficiency evaluation is represented through a dedicated meta-class responsible for performance measurement. This meta-class allows the domain expert to define a custom efficiency function tailored to system-specific evaluation criteria. Therefore, the framework does not impose a predefined metric; instead, it provides a flexible modeling construct through which the modeler specifies how efficiency should be computed. As an illustrative example, efficiency may be defined as the ratio of completed tasks to the total number of tasks defined within the system. However, alternative formulations can be modeled depending on the intended evaluation objectives.

4.1.3 AuctionML

In this section, the AuctionML metamodel is presented as an extension of the CoordinationML for modeling auction-based coordination strategies. Figure 4 illustrates the AuctionML metamodel. The part enclosed in the dashed box contains the concepts added to the CoordinationML to create the AuctionML. In the following, each of these concepts, along with their relationships, will be explained.

The root of the AuctionML is the *AuctionBasedTask Allocation* class, which inherits from the *Step* meta-class of the CoordinationML. As explained in Section 2.1, each auction-based coordination strategy consists of four main steps: announcement, bidding, winner determination, and contract. For each step of the strategy, a class with the same name has been considered, and they all inherit from the *Step* meta-class.

- **Announcement:** the auctioneer can announce one to n tasks for auction in each round. A feature named *announcedTasks* has been considered in this class, which can have values such as *first_task*, *first_n_tasks*, *all_owned_tasks*, or *all_unsold_tasks*. These values are defined in the enumeration *PredefinedVars*.
- **Bidding Strategy:** In the bidding strategy, first, a cost function must be defined. Therefore, there is a one-to-one relationship between the *BiddingStrategy* meta-class and the Function meta-class called *CostFunction*. In addition, it must be determined how many bids the bidders send for each task received in each round of the auction, e.g., bidders bid for all announced tasks or each bidder only bids for the task that has the lowest cost or the highest value for them.
- **Winner Determination:** This process involves sorting the bids, selecting the winner (which has its own specific strategy and can be part of the team formation process), and adding resources to the winning agent's possession for task execution. Therefore, there is a composition relationship between the *WinnerDetermination* meta-class and the *Sort*, *Winner Selection*, *TeamFormation*, and *SupplyingResourcesByWinner* meta-classes.
- **Contract:** In this research, the concept of a contract refers to task allocation to the winning agent. Depending on the coordination strategy, this process may also include allocating resources to the winning agent for task execution. In this case, the resource allocation process is performed according to a function called *ResourceAllocationFunction*. Therefore, there is a relationship with the same name between *Contract* and *Func-*

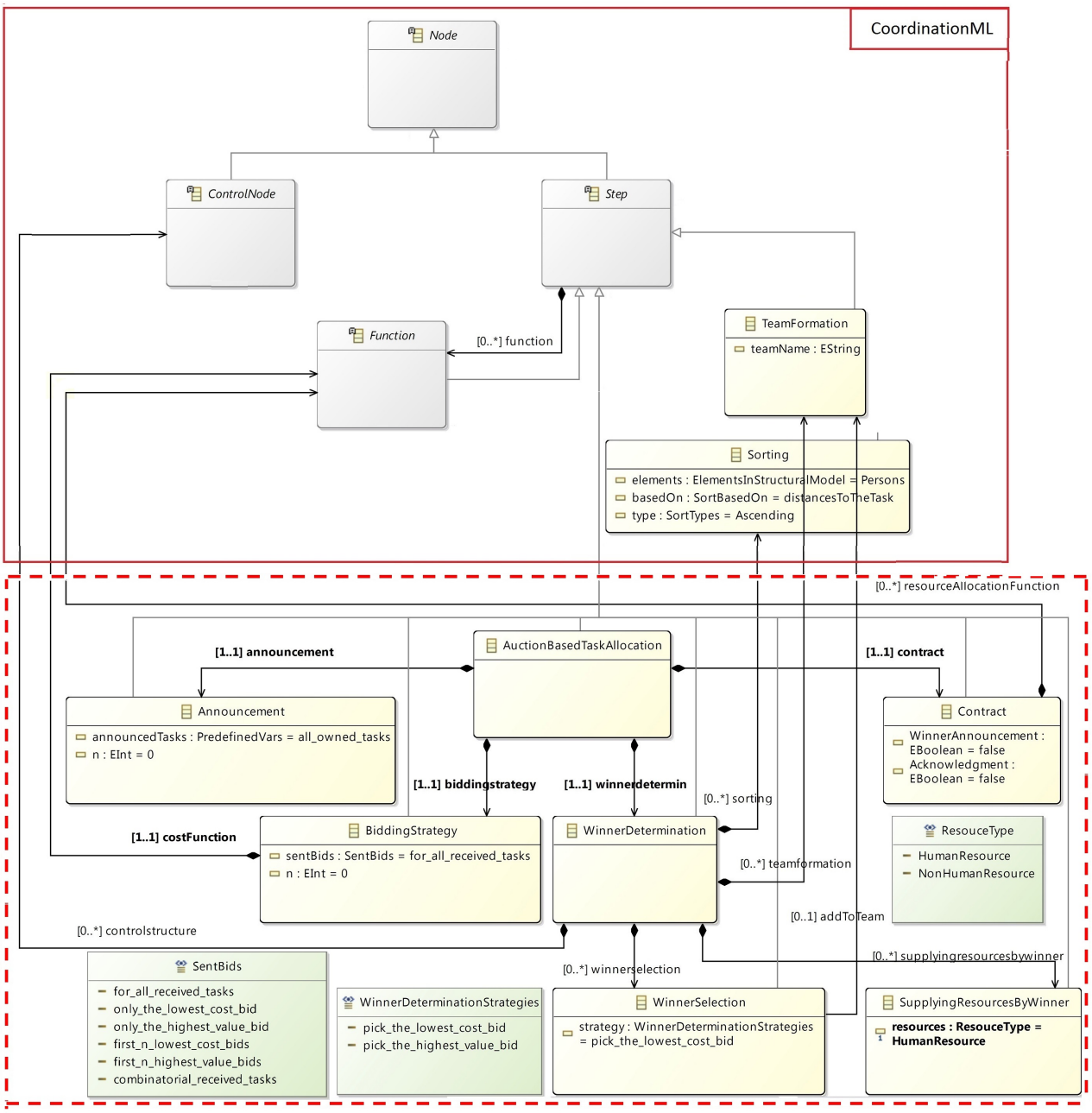


Figure 4. Concepts added to CoordinationML to create AuctionML.

tion meta-classes (existing in CoordinationML). Since separate allocation functions can be considered for each type of resource belonging to the winning agent, or there may be no such process in a coordination strategy at all, the multiplicity of this relationship is considered from zero to infinity. The contract process may also include operations such as *WinnerAnnouncement* and *Acknowledgement*, which respectively mean announcing the winning bidder to all participating agents in the auction and receiving a confirmation message from the winner.

4.2 Tool Support

To facilitate user interaction with the AuctionML modeling language, a graphical editor has been developed. This tool was built using Sirius, along with the Sirius/Xtext integration plugin [1]. Table 3 illustrates the concrete syntax (graphical symbols) designed for each of the elements present in the model. Numbers 1 to 10 correspond to nodes of type *Step*, numbers 11 and 12 correspond to *Control Nodes*, and numbers 13 to 15 correspond to the definition of *Edges*. It should

[1] https://altran-mde.github.io/xtext-sirius-integration.io/

be noted that only symbols 6, 7, 8, and 15 are specific to the AuctionML language, while the rest of the symbols are related to concepts existing in the CoordinationML model, which are defined in a general manner.

Figure 5 depicts the toolbox designed for the AuctionML graphical editor. This toolbox consists of three sections: *Steps*, *Control Nodes*, and *Edges*. Figure 6 illustrates the result of integrating the logical expressions grammar with Sirius in the AuctionML modeling tool. To write logical expressions in this editor, the user can view the list of expressions that can be placed in that section. If the expression written in this section of the logical expressions grammar does not comply, it will be indicated to the user with a red underline indicating the error. For example, in Figure 7, the user has started the logical expression with the '>' operator. Therefore, an appropriate error message is displayed to the user.

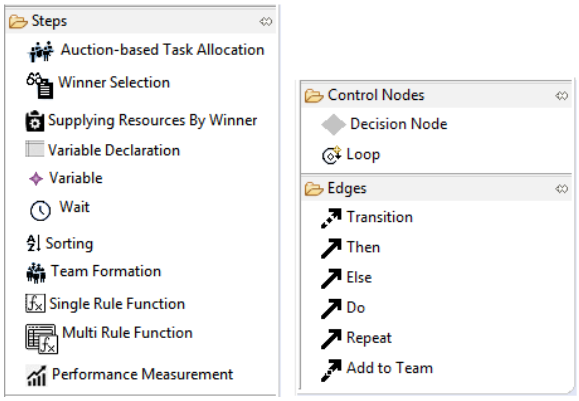


Figure 5. The AuctionML Toolbox.

The usage of mathematical expressions in the graphical editor is similar to logical expressions. Figure 8 illustrates the panel of defined properties for *SingleRuleFunction*. In this panel, there are two sections named *Variable* and *Expression*. In the *Variable* section, the user can view and select a list of predefined variables from the grammar along with other variables defined in the model. In the *Expression* section, the user can define a mathematical expression. This expression must adhere to the grammar of mathematical expressions. Otherwise, an appropriate error message will be displayed to the user.

4.3 Transformation Engine

After creating the coordination strategy model, the model must be transformed into executable code on the CoorERE platform. For this purpose, the model-to-code transformations are written using Acceleo and Model to Text Language (MTL) [28]. Figure 9 illustrates a code snippet of the transformation.

In Lines 43 to 57, classes related to CoorERE, created by EMF, are added. In Lines 202 to 225, the start node is identified. In this regard, a node is considered the start node if it has no incoming transitions. Therefore, for each node in the model, it is checked whether that node is the destination of any transition or not. If the node has no incoming transitions, it is identified as the start node (Line 211). The transformation operation continues by following the transitions sequentially. For example, if the start node is of type *Announcement* (Line 219) and the *announcedTasks* feature is set to *all_owned_tasks*, then the input model is recognized as a model for parallel auction, and accordingly, the code segment for processing tasks in parallel is written (Lines 222 to 225).

In addition to the transformation code project, an Acceleo UI Launcher project has been created, which provides the ability to execute transformations as a dropdown menu on behavioral models.

4.4 Operational Workflow of the Proposed Framework

After developing all required artifacts of the proposed framework, the framework is delivered to users as an Eclipse plug-in. By installing this plug-in within the Eclipse environment, users can perform the modeling process, automatic code generation, and code execution in a structured, step-by-step manner.

Figure 10 illustrates the end-user process for utilizing the proposed framework. As shown in the figure, the structural modeling and behavioral modeling activities can be carried out in parallel. These two phases are fully independent and have no mutual dependencies.

Once the models are created, the structural model must be validated, and the behavioral model must be transformed into executable coordination code compatible with the CoorERE platform. To generate the executable code, the user right-clicks on the behavioral model file (with the *.coordination* extension) and selects the option: Acceleo Model to Text → Generate coordination Java code. This operation automatically generates the corresponding Java implementation of the coordination strategy.

Subsequently, the user transfers the generated Java file to the src directory of the CoorERE tool and executes it within the CoorERE environment. It should be noted that the CoorERE tool has been previously introduced and published in our earlier work [13]. In this paper, CoorERE is reused as the execution platform, while the remaining components and the overall integration process illustrated in Figure 10 are newly developed as part of the proposed framework.

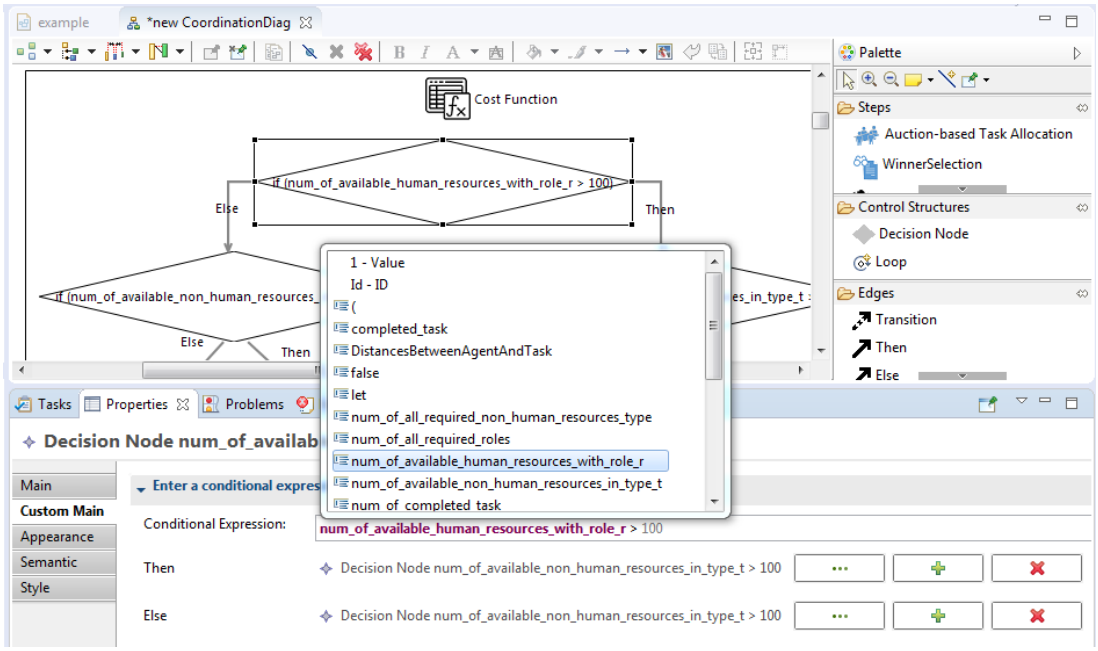


Figure 6. Logical Expression in AuctionML Tool.

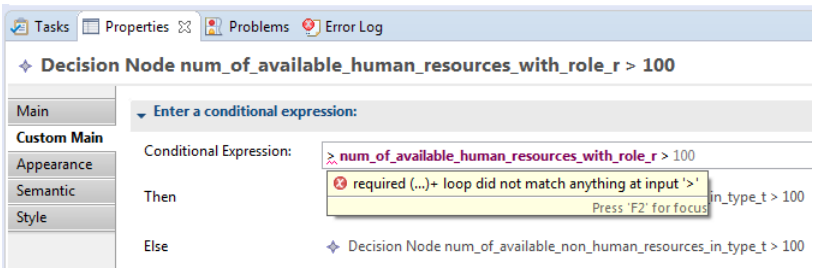


Figure 7. Logical Expression Validation.

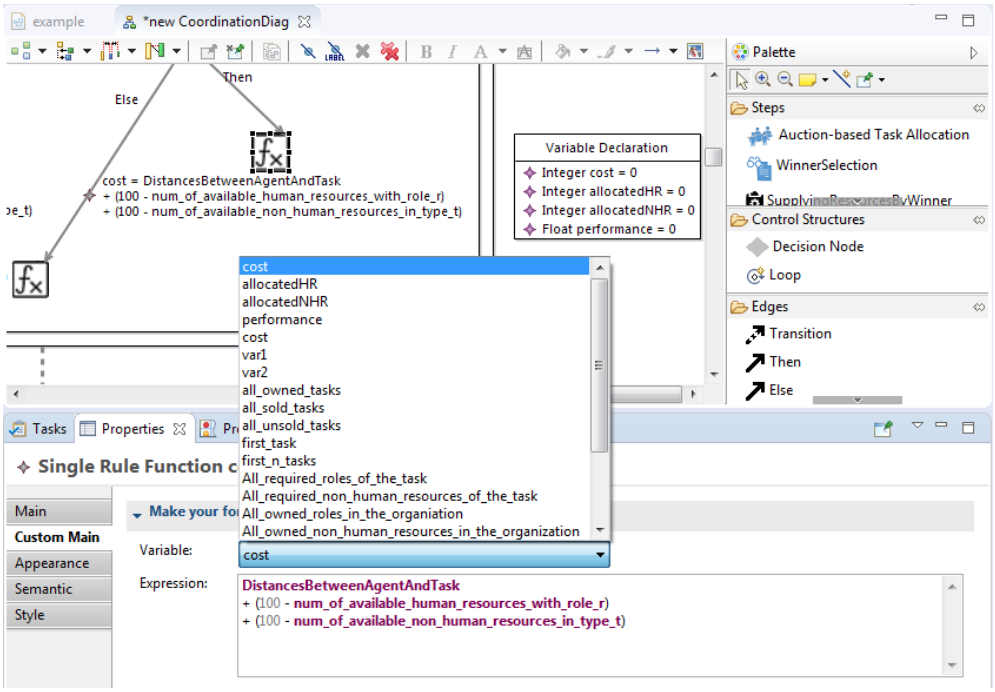
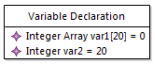
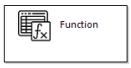
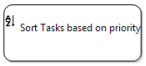
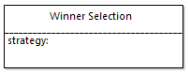
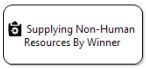
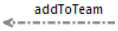
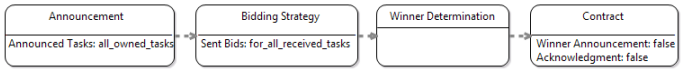


Figure 8. Mathematical Expression in AuctionML Tool.

Table 3. Concrete Syntax of AuctionML.

#	Concept	Icon	#	Concept	Icon
1	Variable Declaration & Variable		9	Single Rule Function	
2	Wait		10	Multi Rule Function	
3	Sorting		11	Decision Node	
4	Team Formation		12	Loop	
5	Performance Measurement		13	Transition	
6	Winner Selection		14	Control edges (do / repeat / then / else)	
7	Supplying Resources By Winner		15	Add to Team	
8	Auction-based Task Allocation				

5 Evaluation

To show the applicability of the proposed framework, two case studies related to organizational auction-based coordination for crisis response are modeled using AuctionML. In our previous work [13], we analyzed and designed two inter and intra-organizational auction-based coordination strategies as the operational semantics of CoorERE. In this paper, the same strategies are selected as the case studies to model at a higher level of abstraction than programming, and then the code of the designed strategies is automatically generated. The main reason for selecting the case studies is that those cover all requirements discussed in Section 4.1.1.

5.1 Modeling Inter-organizational Strategy

In inter-organizational strategy, the coordinator organizations are responsible for assigning tasks to their subordinates. An example of a coordinator organization is the Red Crescent Organization, which is responsible for coordinating relief and rescue stations in a city. The coordinator plays an auctioneer role, and its subordinates are bidders. In this example, the task that must be assigned to relief and rescue stations is search and rescue in different regions of the disaster area. The Red Crescent Organization should select the best organizations for doing each task. In the following, the inter-organizational strategy outlined in [13] is modeled according to the auction-based strategy phases.

(1) **Announcement:** First, the coordinator organization, e.g., Red Crescent Organization, auctions off tasks one by one in order of priority. Figure 11 illustrates the model created for the announcement phase. Initially, by selecting the *"Sort"* option from the toolbox, tasks are sorted in ascending order of priority. Then, for all tasks belonging to the coordinator organization, auction operations for each high-priority task are performed one by one.

(2) **Bidding Strategy:** Figure 12 illustrates how to model the bidding phase. First, in the *"Sent Bids"* section, the number of bids each agent sends is specified. Then, the cost function must be modeled. In this section, the desired cost function is modeled by *"MultiRuleFunction"*, *"Decision Node"*, and *"SingleRuleFunction"* el-

generateCoordination.mtl

```
[comment encoding = UTF-8 /]
[module generateCoordination('http://www.example.org/coordinationMM')]

[template public runCoorModule(aCoorStrategy : CoordinationStrategy)]

[comment @main/]
[file (aCoorStrategy.name + '.java', false, 'UTF-8')]

package coorERE.project.design;
      ...
import CoorERE.Agent;
import CoorERE.Bid;
import CoorERE.CoorEREFactory;
import CoorERE.Environment;
import CoorERE.MAS;
import CoorERE.Messeage;
import CoorERE.NonHumanResource;
import CoorERE.NonHumanResourceName;
import CoorERE.Organization;
import CoorERE.Person;
import CoorERE.Region;
import CoorERE.Role;
import CoorERE.RoleName;
import CoorERE.Task;
import CoorERE.Team;
     ...
[let flag:Boolean = false]
[for(node : Node | aCoorStrategy.node)]
    [if(flag=false)]
        [for(tr: Transition | node.transition)]
            [if(tr.target = node)]
                [flag = true/]  [comment it is not starting node/]
            [/if]
        [/for]
    [/if]
    [if(flag=false)] [comment it is starting node/]
        [if(node.oclIsTypeOf(Loop) = false)]
                    if(auctioneerOrg.getReceive().size() > 0) {
                        auctioneerOrg.getReceive().clear();
                    }
                    if(auctioneerOrg.getSend().size() > 0) {
                        auctioneerOrg.getSend().clear();
                    }
        [if(node.oclIsTypeOf(Announcement))] [comment it is Announcement node/]
            [let ann:Announcement = node.oclAsType(Announcement)]
                [if(ann.announcedTasks = PredefinedVars::all_owned_tasks)]
                    List<Task> taskList = auctioneerOrg.getOwnedTask();

                    //Parallel
                    taskList.parallelStream().forEach(task -> {
```

Figure 9. Code snippet of transformation rules written in MTL language.

ements.

(3) **Winner Determination:** Figure 13 depicts the modeling of the winner determination phase for supplying human and non-human resources. First, all received bids are sorted in ascending order based on their cost. Then, as long as there are bids in the queue, the best organization with the lowest cost bid is examined as the winner, and all human and non-human resources available in that organization are registered to participate in the crisis. If the organization does not have any resources, the next organization is chosen. This process continues until either all required human resources are supplied or all bids in the queue are exhausted. Finally, the task completion percentage in terms of the amount of supplied resources is calculated.

(4) **Contract:** In this phase, the available resources of the winning organizations are fairly allocated to perform the corresponding task. The model of the contract stage is depicted in Figure 14. In this stage, two multi-rule functions are used to model the allocation of human and non-human resources, respectively.

(5) **Performance Measurement:** At the end, the strategy performance is calculated as the ratio of the sum of the completion percentage of all tasks to the total number of tasks. Therefore, the performance will be a numerical value between 0 and 100. Figure 15 illustrates how to model the performance evaluation function according to this criterion.

5.2 Modeling Intra-organizational Strategy

The goal of the intra-organizational strategy is to determine the best individuals for deployment to the incident site. In the following, the modeling of the strategy that is outlined in [13] is described.

A. Announcement and Bidding Strategy: Figure 16 illustrates the model corresponding to the announcement and bidding phases. First, for each role required to perform each task, among the individuals playing that role, the organizations hold auctions. In this way, each participant in the auction sends a bid to the organization. This bid includes the cost of performing the task by the proposed individual. The cost depends on the body's strength, stamina, success rate of the person in previous operations, and their work experience.

(B) Winner Determination and Contract: As depicted in Figure 17, in the winner determination phase, bids are sorted in ascending order based on their cost. Then, as long as there is a bid available and the required number of individuals has not been met, the winner selection operation is performed based on the "minimum cost - best selection" strategy. In the contract phase, a message is sent to announce the winning individuals to all participants in the auction. Also, a confirmation message is sent from the winning individual to the organization. Therefore, the *Winner Announcement* and *Acknowledgement* attributes are set to true to enable the announcement of winners and acknowledgement of task fulfillment by the winning individual.

5.3 Code Generation and Execution

After modeling the inter-organizational and intra-organizational coordination strategies using the transformation execution plugin, the Java code of the coordination strategies is generated. By transferring the generated code to the CoorERE tool and executing it on the desired structural model, the user can observe the output model. The main results in the output model are team formations, task allocations, resource allocation, and strategy performance.

In the following, the designed coordination strategies are applied to the models related to Hurri-

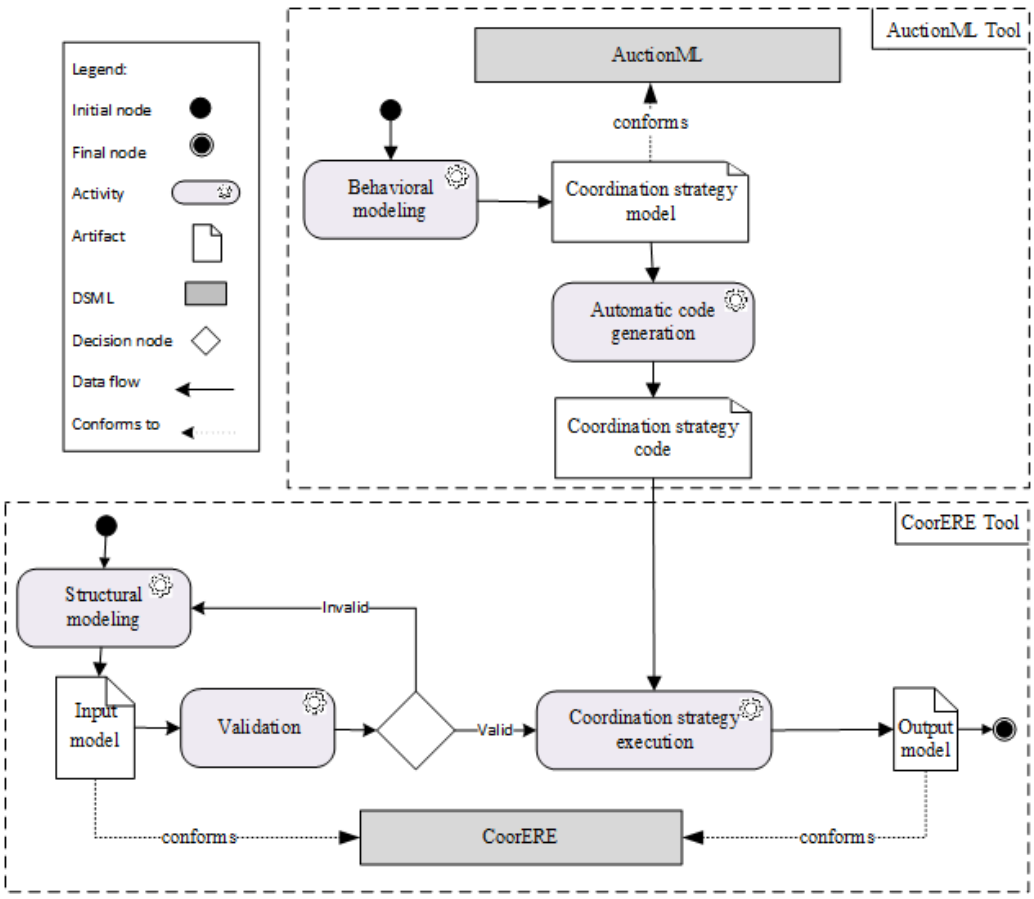


Figure 10. End-user Process for Utilizing the Proposed Framework.

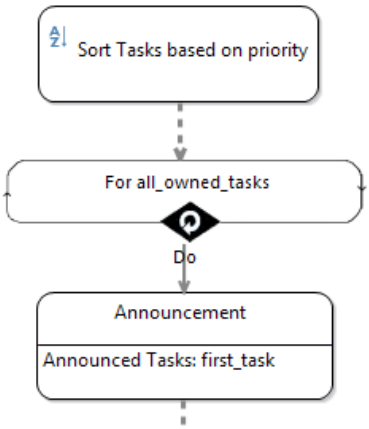


Figure 11. Modeling of the Announcement Phase.

cane Lisa [2] which affected four countries: Belize, Guatemala, Honduras, and Mexico. To manage this crisis, responsible organizations must collaborate internationally to reduce the damage caused by the disaster [48]. It is worth mentioning that the strategies were also applied to the Tehran earthquake model [13] and all expected results were obtained. The model included three search and rescue tasks and six responsible organizations with 66 employees in the organizations. Table 4 shows the features of the selected models.

As shown in this table, the hypothetical earthquake in Tehran has been implemented as a small-scale regional disaster with homogeneous tasks and organizations. In contrast, Hurricane Lisa is a large-scale disaster that involves heterogeneous organizations at the international level to carry out heterogeneous tasks. Therefore, the second model is more complex than the first. Additionally, in terms of the number of fixed model elements (i.e., all model elements except individuals, which are randomly generated), the Hurricane Lisa model is 3.5 times larger than the first case study.

In our previous work [13], the output results related to applying the coordination strategies in the Tehran earthquake model have been shown. Therefore, in the following, only the results obtained from the execution on the Hurricane Lisa model will be described. Figure 18 to 20 show these results.

As indicated in Figure 18, this model includes 20 organizations and a total of 1867 individuals em-

[2] https://reliefweb.int/disaster/tc-2022-000357-blz

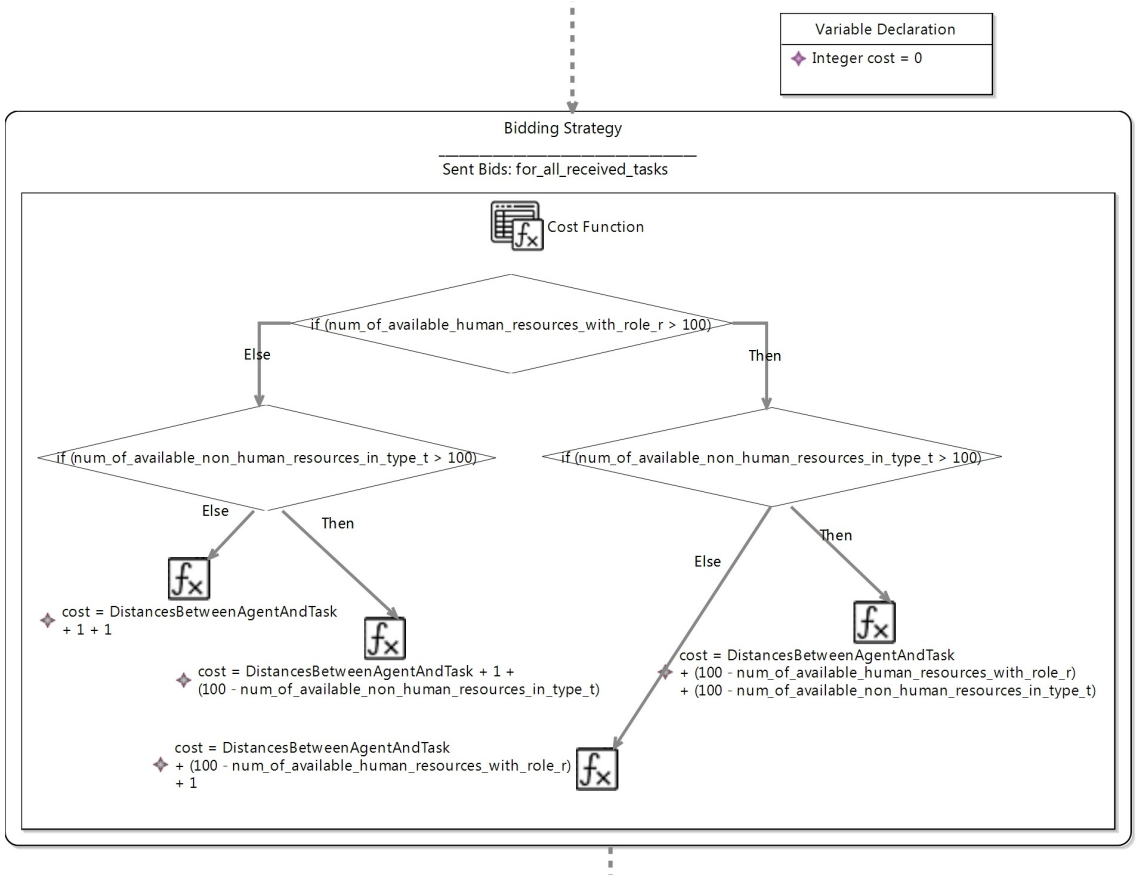


Figure 12. Modeling of the Bidding Strategy Phase.

Table 4. Comparison of the Two Selected Disasters.

Name	Scale	Type of Tasks (Homogeneous / Heterogeneous)	Type of Organizations (Homogeneous / Heterogeneous)	Number of Fixed Model Elements
Tehran Earthquake	Regional	Single Type (Rescue)	Single Type (Rescue Stations)	53
Hurricane Lisa	International	5 Different Types	5 Different Types	182

ployed within these organizations. The organizations are defined in three regions: Belize, Honduras, and Guatemala. Details related to the organizations defined in each region can be found in the modeling document [3]. As a result of implementing the coordination strategy in the aforementioned model, an efficiency rate of 81 percent has been achieved.

The winning organizations for each task and the percentage of resources provided by them are shown in Figure 19. For example, in rescue operations and the provision of food and shelter, Guatemala and Honduras collaborate with Belize and share their resources with them.

Figure 20 provides an example of the outcome of the intra-organizational coordination strategy in the Hurricane Lisa model. This figure shows part of the view of Fire Station No. 2 in Belize City, where a subgroup has been assigned the task of debris removal. Based on this, two teams, consisting of six and eight crane operators respectively, are dispatched to two areas of Belize City for debris removal, while six other operators remain available within the organization.

[3] https://mdse.ui.ac.ir/Tools/CoorERE/HurricaneLisa%20Case%20Study.pdf

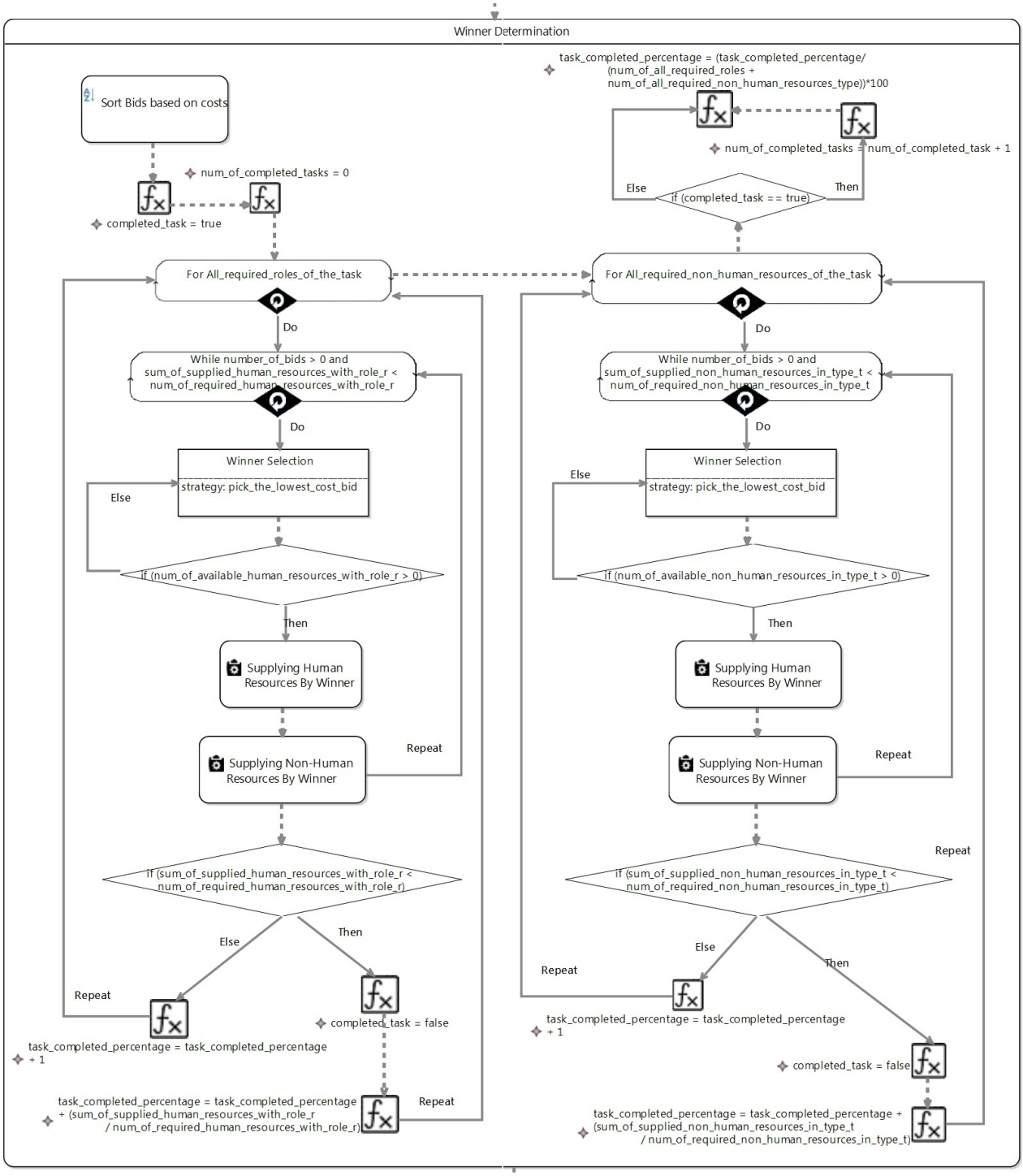


Figure 13. Modeling the Winning Determination Phase.

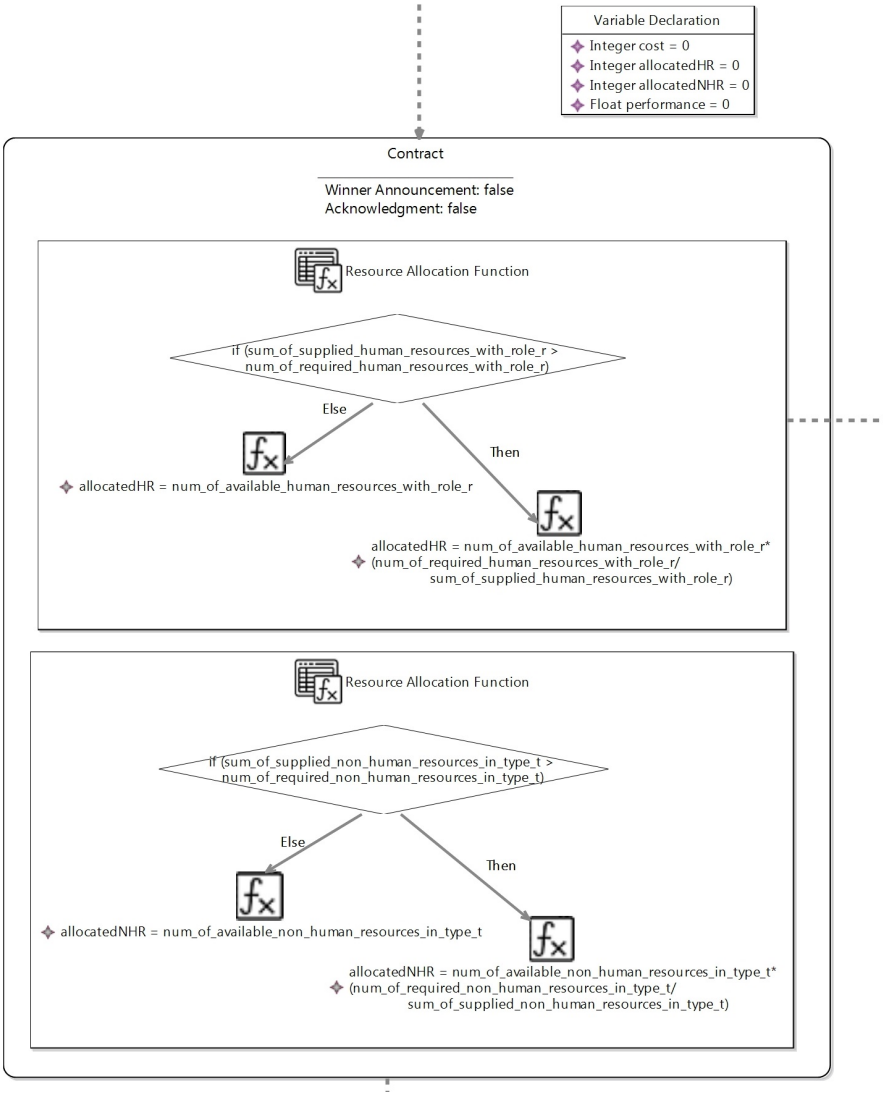


Figure 14. Modeling of the Contract Phase.

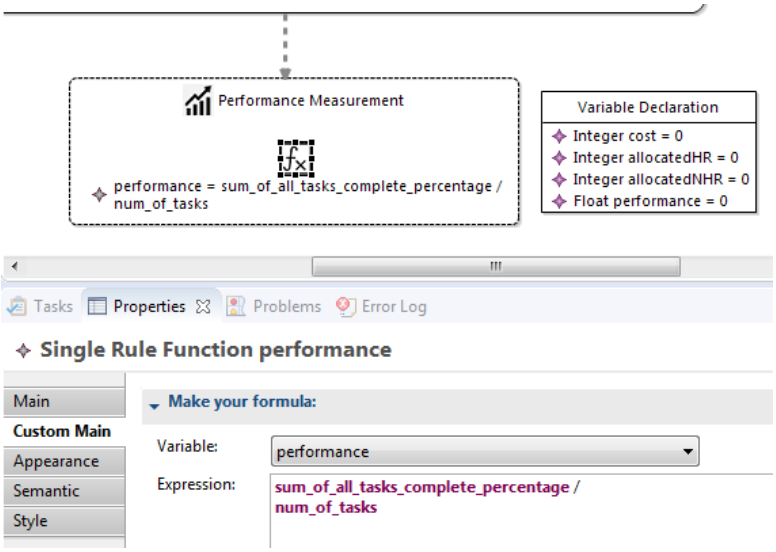


Figure 15. Modeling of the Performance Measurement.

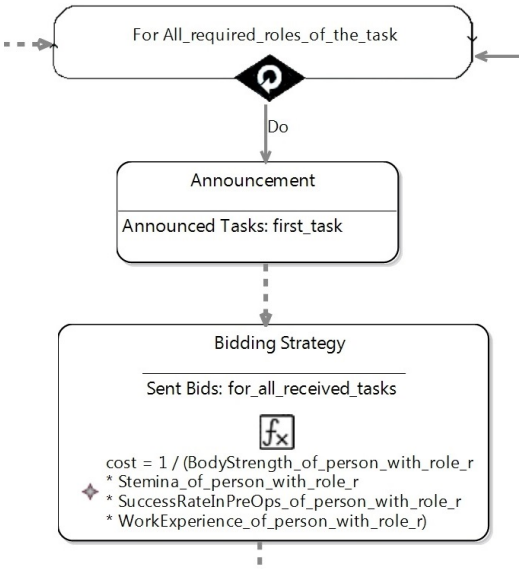


Figure 16. Intra-organizational Strategy Model - Part 1.

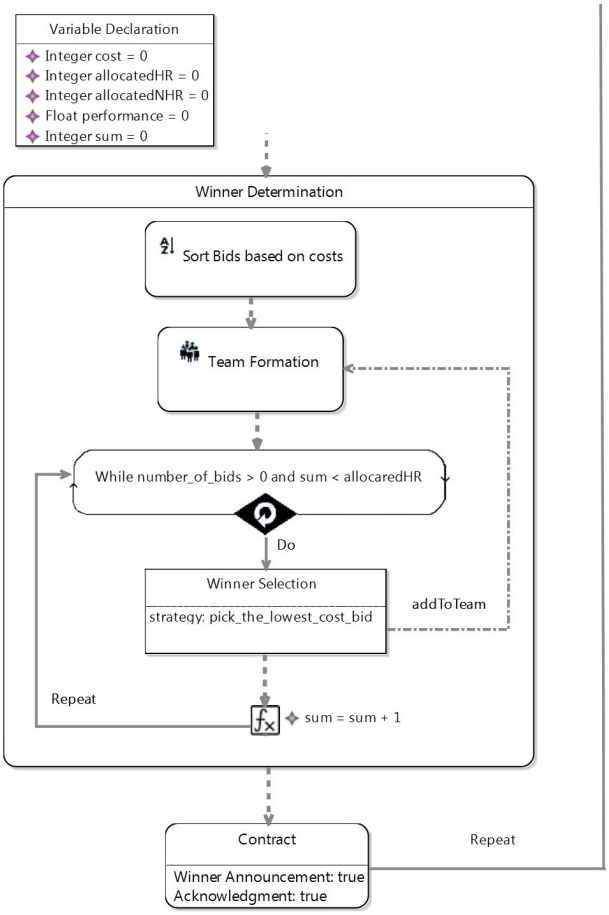


Figure 17. Intra-organizational Strategy Model - Part 2.

5.4 Results

The conducted evaluations demonstrate that the proposed framework, particularly AuctionML and its supporting tool, can successfully model and execute auction-based coordination strategies in different crisis response scenarios. The designed strategies were fully modeled using AuctionML and automatically transformed into executable code through the implemented model-to-code transformations.

The generated code was integrated into the CoorERE modeling environment and executed on ERE models representing two distinct case studies: the Tehran earthquake and Hurricane Lisa. Despite differences in scale, organizational structures, and resource distributions, the strategies were executed without requiring structural modifications to the metamodels. This confirms the independence of CoordinationML and AuctionML from specific disaster types and organizational settings.

Furthermore, the framework supports systematic and quantitative evaluation of coordination strategies through domain-defined performance functions. By varying auction parameters such as the number of rounds and the number of tasks allocated at each stage, different execution outcomes were obtained and analyzed. This enabled comparative assessment of strategy configurations within the same operational environment.

The results indicate that the proposed framework enables consistent modeling, automatic code generation, and executable evaluation of auction-based coordination strategies across heterogeneous crisis scenarios.

6 Discussion

This section analyzes the advantages, limitations, and broader implications of the proposed framework beyond the experimental evaluation.

6.1 Advantages

The evaluation results demonstrate that the proposed framework enables systematic modeling and quantitative assessment of auction-based coordination strategies in ERE. Its main strengths include:

(1) Domain-specific abstraction. Auction-based coordination strategies are modeled using domain-aligned constructs such as agents, tasks, and allocation steps, making the framework accessible to domain experts.

(2) Explicit modeling of coordination semantics. Unlike approaches where coordination logic is embedded directly in source code, the proposed framework

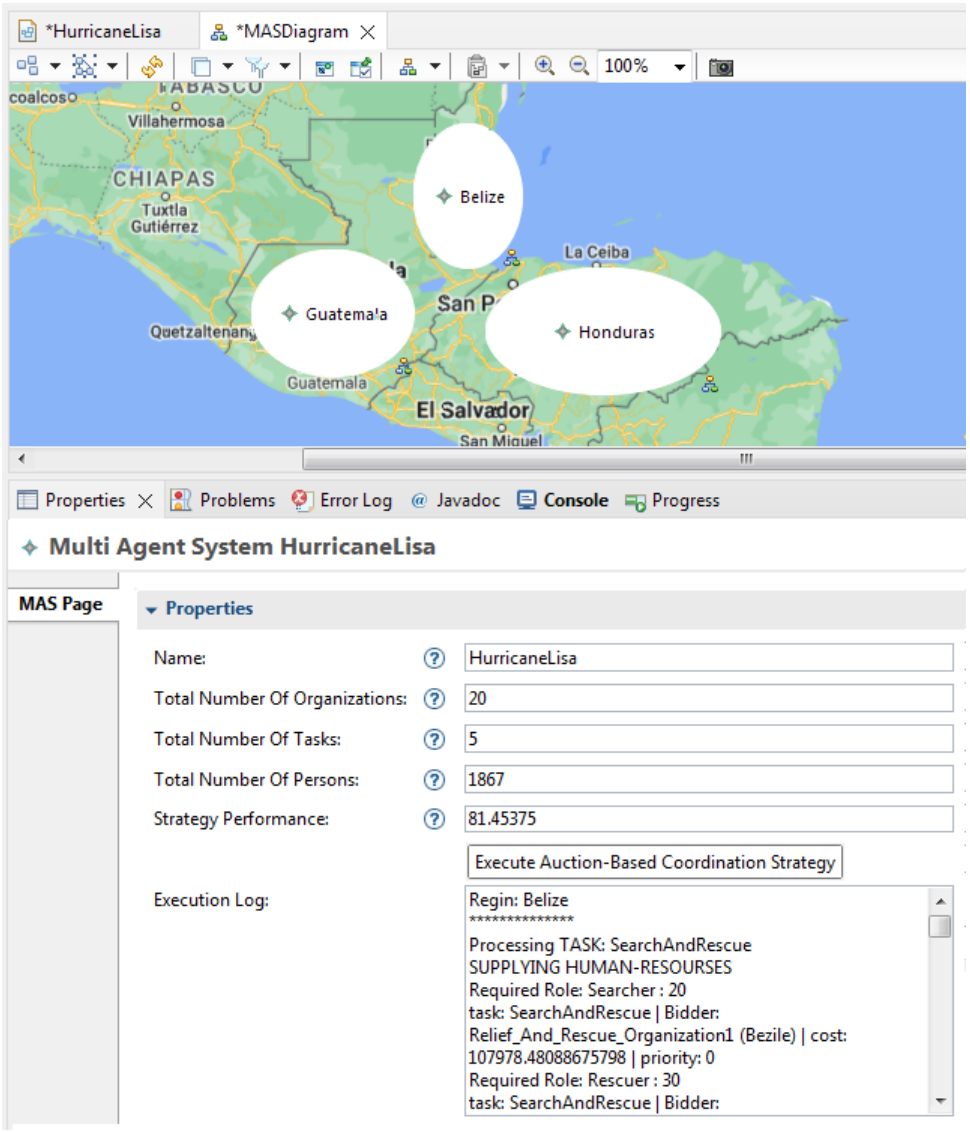


Figure 18. Execution Log and Strategy Performance in the CoorERE Tool.

	Completed percentage	Winner organizations
Task RemoveDebris	0.25	[Fire_Department2 (Belize), Fire_Department1 (Belize)]
Task SearchAndRescue	0.23466666	[Relief_And_Rescue_Organization1 (Bezile), Relief_And_Rescue_Organization (Guatemala), Relief_And_Rescue_Organization1 (Honduras)]
Task TransferringInjuredPeopleToHospitals	0.25	[Relief_And_Rescue_Organization2 (Honduras)]
Task TreatmentsInHospitals	0.35	[Hospital1 (Belize), Hospital3 (Belize), Hospital2 (Belize)]
Task SupplyingFoodAndShelter	0.2719375	[Relief_And_Rescue_Organization1 (Bezile), Relief_And_Rescue_Organization2 (Bezile), Relief_And_Rescue_Organization (Guatemala), Relief_And_Rescue_Organization1 (Honduras),

Figure 19. Winning organizations after executing inter-organizational coordination strategy.

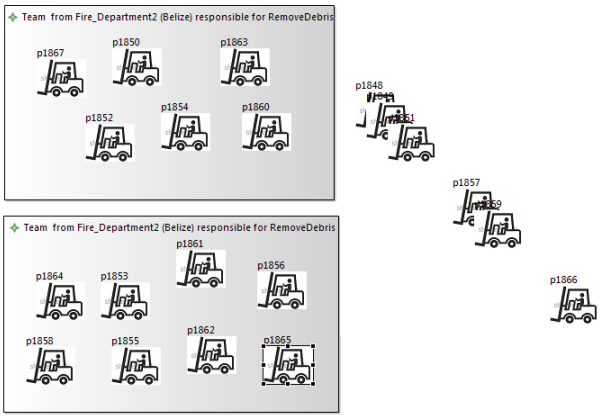


Figure 20. Team formation for debris removal at Fire Station No. 2 in Belize.

represents task allocation strategies as first-class metamodel constructs. This improves traceability and conceptual clarity.

(3) Integrated model-driven pipeline. Modeling, transformation, execution, and performance evaluation are unified in a single workflow. This reduces implementation inconsistencies and enhances reproducibility.

(4) Extensibility. The CoordinationML metamodel can be extended to support additional coordination strategies by expanding core meta-classes such as *Step*. This design supports gradual evolution of the framework.

6.2 Limitations

Despite its strengths, the framework has several limitations:

- The current implementation focuses primarily on auction-based coordination strategies. Other coordination paradigms require additional metamodel extensions.
- Automatic code generation is most effective when DSML concepts closely align with coordination logic. For highly abstract or loosely defined domains, partial manual refinement may still be necessary.
- Large-scale empirical validation in real-world crisis management infrastructures remains future work.

6.3 Theoretical Implications

From a theoretical perspective, this work contributes to bridging coordination theory and model-driven engineering. It demonstrates that coordination mechanisms, specifically auction-based task allocation, can be formalized as executable domain constructs within a DSML rather than treated merely as algorithmic implementations.

This contributes to research on executable modeling languages and domain-specific coordination semantics in socio-technical systems.

6.4 Practical Implications

Practically, the framework enables emergency planners to experiment with and evaluate coordination strategies before crisis occurrence. For example, resource shortages can be identified during simulation, allowing organizations to take preventive measures.

Although Large Language Models (LLMs) can generate code from textual prompts, they do not inherently ensure formal domain consistency, traceability, or deterministic transformation. In contrast, the proposed framework guarantees structural correctness through metamodel constraints and rule-based transformations. Nevertheless, future extensions could integrate specialized LLMs as intelligent assistants, for example, to support DSML model creation from natural language descriptions while preserving the formal model-driven backbone of the system.

7 Conclusions and Future Works

This paper presented a domain-specific, executable model-driven framework for modeling and evaluating auction-based coordination strategies in ERE. By embedding coordination semantics into a DSML and enabling automated model-to-code transformation, the framework bridges high-level domain modeling and executable coordination behavior.

Finally, the suggestions for future work in continuation of this research include:

- Verification of the designed coordination strategies using formal or semi-formal methods
- Extending the proposed framework to cover other task allocation algorithms
- Addressing optimization problems in resource allocation, task assignment, and team formation, and proposing algorithms in these areas.
- Investigating the integration of AI-assisted modeling techniques

References

[1] 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.

[2] Mari Olsen, PerAnders Oskarsson, Niklas Hallberg, Magdalena Granåsen, and Johan Nordström. Exploring collaborative crisis

management: a model of essential capabilities. *Safety Science*, 162:106092, 2023. doi:10.1016/j.ssci.2023.106092.

[3] Mehdi Hashemipour, Steven MF Stuban, and Jason R Dever. A community-based disaster coordination framework for effective disaster preparedness and response. *Australian Journal of Emergency Management*, 32(2):41–46, 2017.

[4] Inès Thabet, Mohamed Chaawa, and Lamjed Ben Said. A multi-agent organizational model for a snow storm crisis management. In *Proceedings of the International Conference on Information Systems for Crisis Response and Management in Mediterranean Countries*, volume 196, pages 143–156. Springer, 2014. doi:10.1007/978-3-319-11818-5_13. URL http://link.springer.com/chapter/10.1007/978-3-319-11818-5{_}13.

[5] Jian Tang, Kejun Zhu, Haixiang Guo, Chengzhu Gong, Can Liao, and Shuwen Zhang. Using auction-based task allocation scheme for simulation optimization of search and rescue in disaster relief. *Simulation Modelling Practice and Theory*, 82:132–146, 2018. doi:10.1016/j.simpat.2017.12.014.

[6] Myriam Abramson, William Chao, Joseph Macker, and Ranjeev Mittu. Coordination in disaster management and response: A unified approach. In *Massively Multi-Agent Technology*, Lecture Notes in Artificial Intelligence, pages 162–175. Springer, 2008. doi:10.1007/978-3-540-85449-4_12.

[7] Xing Su, Minjie Zhang, and Quan Bai. Coordination for dynamic weighted task allocation in disaster environments with time, space and communication constraints. *Journal of Parallel and Distributed Computing*, 97:47–56, 2016. doi:10.1016/j.jpdc.2016.06.010.

[8] Navid Hooshangi and Ali Asghar Alesheikh. Agent-based task allocation under uncertainties in disaster environments: An approach to interval uncertainty. *International journal of disaster risk reduction*, 24:160–171, 2017. doi:10.1016/j.ijdrr.2017.06.010.

[9] Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019. doi:10.1145/3303848.

[10] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-driven software engineering in practice*. Morgan & Claypool Publishers, 2017. doi:10.1007/978-3-031-02549-5.

[11] Tansu Zafer Asici, Baris Tekin Tezel, and Geylani Kardas. On the use of the analytic hierarchy process in the evaluation of domain-specific modeling languages for multi-agent systems. *Journal of Computer Languages*, 62:101020, 2021. doi:10.1016/j.cola.2020.101020.

[12] Martin Fowler. *Domain-Specific Languages*. Addison-Wesley Professional, 2010. ISBN 9780321712943.

[13] Samaneh Hoseindoost, Afsaneh Fatemi, and Bahman Zamani. An executable domain-specific modeling language for simulating organizational auction-based coordination strategies for crisis response. *Simulation Modelling Practice and Theory*, 131:102880, 2024. doi:10.1016/j.simpat.2023.102880.

[14] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. *Cooperative robots and sensor networks 2015*, pages 31–51, 2015. doi:10.1007/978-3-319-18299-5_2.

[15] Peter Cramton, Yoav Shoham, and Richard Steinberg. Introduction to combinatorial auctions. In *Combinatorial Auctions*, pages 1–13. MIT Press, 2006. doi:10.7551/mitpress/9780262033428.003.0001.

[16] Michael W. Otte, Michael J. Kuhlman, and Donald Sofge. Auctions for multi-robot task allocation in communication limited environments. *Autonomous Robots*, 44(3–4):547–584, 2020. doi:10.1007/s10514-019-09828-5.

[17] Ignacio Palacios-Huerta, David C. Parkes, and Richard Steinberg. Combinatorial auctions in practice. *Journal of Economic Literature*, 62(2): 517–553, 2024. doi:10.1257/jel.20221679.

[18] Shane Sendall and Wojtek Kozaczynski. Model transformation: The heart and soul of model-driven software development. *IEEE Software*, 20 (5):42–45, 2003. doi:10.1109/MS.2003.1231150.

[19] Markus Völter, Thomas Stahl, Jorn Bettin, Arno Haase, and Simon Helsen. *Model-driven software development: technology, engineering, management*. John Wiley & Sons, 2013.

[20] Object Management Group. Meta Object Facility (MOF) Specification 2.5. http://www.omg.org/spec/MOF/2.5, Date Accessed: January 3, 2024.

[21] Barrett R Bryant, Jeff Gray, Marjan Mernik, Peter J Clarke, and Robert B France. Challenges and Directions in Formalizing the Semantics of Modeling Languages. *Computer Science and Information Systems*, 8(2):225–253, 2011. doi:10.2298/csis110114012b.

[22] Object Management Group. Object constraint language (ocl), version 2.4. https://www.omg.org/spec/OCL/2.4, Date Accessed: January 3, 2024. OMG Document formal/14-02-03.

[23] Object Management Group. Unified Modeling Language (UML) - Version 2.5. http://www.omg.org/spec/UML/2.5/, Date Accessed: Jan-

uary 2, 2024.

[24] Petri Nets Steering Committee. Petri nets. www.petrinets.info/, Date Accessed: January 4, 2024.

[25] Viviane Torres Da Silva, Ricardo Choren, and Carlos J P De Lucena. MAS-ML: a multiagent system modelling language. *International Journal of Agent-Oriented Software Engineering*, 2(4):382–421, 2008. ISSN 17461383. doi:10.1504/ijaose.2008.020138.

[26] Samaneh Hoseindoost, Tahereh Adamzadeh, Bahman Zamani, and Afsaneh Fatemi. A model-driven framework for developing multi-agent systems in emergency response environments. *Software and Systems Modeling*, 18(3):1985–2012, 2019. doi:10.1007/s10270-017-0627-4.

[27] Samaneh Hoseindoost, Afsaneh Fatemi, and Bahman Zamani. Towards a model-driven framework for simulating interactive emergency response environments. *Journal of Computing and Security*, 5(1):35–49, 2018. doi:10.22108/JCS.2018.112297.1008.

[28] Object Management Group. MOF Model to Text Transformation Language Specification Version 1.0. https://www.omg.org/spec/MOFM2T/1.0, Date Accessed: January 6, 2024.

[29] Eclipse.org. Eclipse Model To Text (M2T) Project. https://eclipse.dev/acceleo, Date Accessed: January 4, 2024.

[30] Paolo Ciancarini. Coordination models and languages as software integrators. *ACM Computing Surveys (CSUR)*, 28(2):300–302, 1996. doi:10.1145/234528.234732.

[31] George A Papadopoulos and Farhad Arbab. Coordination models and languages. In *Advances in Computers*, volume 46, pages 329–400. Elsevier, 1998. doi:10.1016/s0065-2458(08)60208-9.

[32] Marjan Sirjani, editor. *Coordination Models and Languages*, volume 7274 of *Lecture Notes in Computer Science*, 2012. Springer. ISBN 978-3-642-30828-4. doi:10.1007/978-3-642-30829-1.

[33] Sébastien Truptil, Frédérick Bénaben, Pierre Couget, Matthieu Lauras, Vincent Chapurlat, and Hervé Pingaud. Interoperability of information systems in crisis management: Crisis modeling and metamodeling. In *Enterprise Interoperability III: New Challenges and Industrial Approaches*, Proceedings of IESA 2008, pages 583–594. Springer, 2008. doi:10.1007/978-1-84800-221-0_46.

[34] Matthieu Lauras, Sébastien Truptil, and Frédérick Bénaben. Towards a better management of complex emergencies through crisis management meta-modelling. *Disasters*, 39(4):687–714, 2015. doi:10.1111/disa.12122.

[35] Jörn Franke, François Charoy, and Paul El Khoury. Framework for coordination of activities in dynamic situations. *Enterprise Information Systems*, 7(1):33–60, 2013. doi:10.1080/17517575.2012.690891.

[36] Nguyen-Tuan-Thanh Le, Chihab Hanachi, Serge Stinckwich, and Tuong-Vinh Ho. Discovering crisis models to help assess coordination plans: A case study of tsunami response plan given by ho chi minh city, vietnam. *Vietnam Journal of Computer Science*, 4(2):97–110, 2017. doi:10.1007/s40595-016-0078-9.

[37] Xiong Li, Wei Pu, and Xiaodong Zhao. Agent action diagram: Toward a model for emergency management system. *Simulation Modelling Practice and Theory*, 94:66–99, 2019. doi:10.1016/j.simpat.2019.02.004.

[38] Gert-Jan De Vreede and Daniel TT Van Eijck. Modeling and simulating organizational coordination. *Simulation & Gaming*, 29(1):60–87, 1998. doi:10.1177/1046878198291006.

[39] Chris J. van Aart. *Organizational Principles for Multi-Agent Architectures*. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser, Basel, 2005. ISBN 978-3-7643-7213-2. doi:10.1007/b137137.

[40] Jan Sudeikat and Wolfgang Renz. Masdynamics: Toward systemic modeling of decentralized agent coordination. In Klaus David and Kurt Geihs, editors, *Kommunikation in Verteilten Systemen (KiVS 2009)*, Informatik Aktuell, pages 79–90. Springer, 2009. doi:10.1007/978-3-540-92666-5_7.

[41] Juan Carlos Nieves, Julian Padget, Wamberto Weber Vasconcelos, Athanasios Staikopoulos, Owen Cliffe, Frank Dignum, Javier Vázquez-Salceda, Siobhán Clarke, and Chris Reed. Coordination, organisation and model-driven approaches for dynamic, flexible, robust software and services engineering. In Schahram Dustdar and Fei Li, editors, *Service Engineering: European Research Results*, pages 85–115. Springer, 2011. doi:10.1007/978-3-7091-0415-6_4.

[42] Navid Hooshangi and Ali Asghar Alesheikh. Developing an Agent-Based Simulation System for Post-Earthquake Operations in Uncertainty Conditions: A Proposed Method for Collaboration among Agents. *International Journal of Geo-Information*, 7(1):27–49, 2018. doi:10.3390/ijgi7010027.

[43] Mattijs Ghijsen, Wouter Jansweijer, and Bob Wielinga. Designing a search and rescue simulation environment for studying the performance of agent organizations. In *The 24th Benelux Conference on Artificial Intelligence (BNAIC 2012)*, pages 115–120. Citeseer, 2012.

[44] Mehdi Hashemipour, Steven Stuban, and Jason Dever. A disaster multiagent coordination simulation system to evaluate the design of a first-response team. *Systems Engineering*, 21(4):322–344, 2018. doi:10.1002/sys.21437.

[45] Yoosef Abushark, Tim Miller, John Thangarajah, Michael Winikoff, and James Harland. Requirements specification via activity diagrams for agent-based systems. *Autonomous Agents and Multi-Agent Systems*, 31(3):565–614, 2017. doi:10.1007/s10458-016-9327-7.

[46] Muhammad Irfan and Adil Farooq. Auction-based task allocation scheme for dynamic coalition formations in limited robotic swarms with heterogeneous capabilities. In *2016 International Conference on Intelligent Systems Engineering (ICISE)*, pages 210–215. IEEE, 2016. doi:10.1109/intelse.2016.7475122.

[47] Jieke Shi, Zhou Yang, and Junwu Zhu. An auction-based rescue task allocation approach for heterogeneous multi-robot system. *Multimedia Tools and Applications*, 79(21–22):14529–14538, 2020. doi:10.1007/s11042-018-7080-4.

[48] OCHA.UNCT-Belize. Belize: Hurricane Lisa, Situation Report No.1. Technical report, Date Accessed: January 2, 2024.

Appendix

In this section, the mathematical and logical expression grammar used in modeling coordination strategies is introduced, respectively.

A Mathematical Expression Grammar

As mentioned in Section 4.1.2, the *expression* feature in the *SingleRuleFunction* meta-class can include a mathematical expression. For this purpose, the grammar for mathematical expressions has been defined. Listing 1 illustrates the defined grammar by the Xtext language, where operator precedence is observed.

```
grammar org.xtext.example.mathexp.MathExp with
org.eclipse.xtext.common.Terminals

generate mathExp
"http://www.xtext.org/example/mathexp/MathExp"

MathExp:
 exps+=NamedExp+;

NamedExp:
 exp=Exp;

Exp returns Expression:
 Factor (({Plus.left=current} '+' |
 {Minus.left=current} '-') right=Factor)*;

Factor returns Expression:
 Primary (({Mult.left=current} '*' |
 {Div.left=current} '/' |
 {Mod.left=current} '%') right=Primary)*;

Primary returns Expression:
 Operands | Literal | Number | Parenthesis |
 VariableBinding | VariableUse;

VariableUse returns Expression:
 {Var} id=ID;

VariableBinding returns Expression:
 {Let} 'let' id=ID '=' binding=Exp 'in' body=Exp 'end';

Parenthesis returns Expression:
 '(' Exp ')';

Number returns Expression:
 {Num} value=INT;

Literal returns Expression:
 {Literal} value=("true" | "false");

Operands:
 name=myOperands;

enum myOperands:
 DistancesBetweenAgentAndTask |
 num_of_tasks |
 num_of_available_human_resources_with_role_r |
 num_of_required_human_resources_with_role_r |
```

```
num_of_all_required_roles |
num_of_available_non_human_resources_in_type_t |
num_of_required_non_human_resources_in_type_t |
num_of_all_required_non_human_resources_type |
sum_of_supplied_human_resources_with_role_r |
sum_of_supplied_non_human_resources_in_type_t |
number_of_bids |
score_of_bid_sender |
task_completed_percentage |
sum_of_all_tasks_complete_percentage |
completed_task |
num_of_completed_task |
BodyStrength_of_person_with_role_r |
Stemina_of_person_with_role_r |
SuccessRateInPreOps_of_person_with_role_r |
WorkExperience_of_person_with_role_r;
```

Listing 1: Mathematical expressions grammar implemented by Xtext language

The specific feature of the grammar defined in this paper is that it includes the necessary keywords for modeling coordination strategies (Lines 44 to 64 in Listing 1). These keywords can be used as operands in a mathematical or logical expression and can be extended for specific coordination strategies. This feature enables the modeling of coordination strategies to be completely independent of structural modeling. The value of these operands is obtained from the structural model at runtime. However, during the modeling phase, strategies can be modeled independently of the structural model.

B Logical Expression Grammar

As mentioned in Section 4.1.2, the *conditionalExpression* feature in the classes *Wait*, *Loop*, and *DecisionNode* can include a logical expression. A logical expression in the model consists of mathematical expressions along with logical operators (`such as and, or, <, !=`). Listing 2 shows the grammar of the logical expression. *Exp* at Line 26 is exactly equivalent to the *Exp* at Line 13 in Listing 1, which we will refrain from representing again in this section.

```
grammar org.xtext.example.mydsl.LogicalExp with
org.eclipse.xtext.common.Terminals

generate logicalExp
"http://www.xtext.org/example/mydsl/LogicalExp"

LogicalExp:
 exps+=NamedExp+;

NamedExp:
 exp=OrExpression;

OrExpression returns Expression:
 AndExpression ({OrExpression.left=current} "or"
 right=AndExpression)*;

AndExpression returns Expression:
 ComparisonExpression ({AndExpression.left=current}
 "and" right=ComparisonExpression)*;

ComparisonExpression returns Expression:
 Exp ({ComparisonExpression.left=current}
 operator=("<" | "<=" | "==" | "!=" | ">=" | ">")
 right=Exp)*;

Exp returns Expression:
```

Listing 2: A snippet of Logical expressions grammar

Samaneh Hoseindoost holds B.Sc., M.Sc., and Ph.D. degrees in Computer Software Engineering from the University of Isfahan, Isfahan, Iran (2014, 2017, and 2024, respectively). During her studies, from 2015 to 2024, she was actively involved with the Model-Driven Software Engineering Research Group (MDSERG) at the University of Isfahan. Her research focuses on Model-Driven Software Engineering, Multi-Agent Systems, and Crisis Management Systems.

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, respectively, all in Computer Engineering. Currently, she is an associate professor in the Faculty of Computer Engineering at the University of Isfahan, and a member of the UI Big Data research group. Her main research interests lie in the fields of Complex Networks, Question Answering and Dialogue Systems, and applications of LLMs.

Bahman Zamani holds a Ph.D. in Computer Science from Concordia University, Montreal, QC, Canada, for his work on Pattern Language Verification. He is an Associate Professor Emeritus in the Department of Software Engineering, University of Isfahan, Isfahan, Iran. His main research interest is Model-Driven Software Engineering (MDSE). He was the founder and is now a member of the MDSE Research Group at the University of Isfahan.