# An Ontology-based approach for Scenario Generation in Flight Simulation Systems

**Hung Tran, Howard Cheung, Michael Tillett**
**CAE USA Inc.**
**Tampa, FL**
**hung.tran@caemilusa.com; howard.cheung@caemilusa.com; michael.tillett@caemilusa.com**

## ABSTRACT

The core component of a simulation-based training system is creating training scenarios. The scenario creation process is essential for simulation-based training systems since scenarios are designed to provide the context for the training. An effective training scenario should provide opportunities for trainees to practice their skills and receive feedback from the instructors on their performance. A training scenario is usually characterized by three main components: (1) the initialization, (2) the key events that must happen during the training, and (3) the termination conditions. Typically, the creation of validated and effective training scenarios must be carried out by qualified instructors and highly trained subject matter experts. The process is challenging and time-consuming, therefore expensive. Training objectives must be analyzed before developing a training scenario to determine the knowledge and skills required as part of the training to ensure scenario outputs are domain-valid and pedagogically effective. The objective of this paper is to describe an approach that will facilitate the task of generating training scenarios. The proposed approach is based on the ontology of knowledge presentation. The paper presents an ontology developed to capture simulation scenario attributes pertinent to the flight simulation domain and describes the role of this ontological analysis in creating training scenarios. Leveraging the Simulation Interoperability Standards Organization (SISO) interoperability standard, this study expands the C2SIM core ontology by adding a Training Scenario Extension layer. As a result, Training scenarios can be generated in a standardized format that complies with the base and extended C2SIM ontology. Finally, the paper will present a use case of Air Refueling Training as an application of this approach. This work constitutes an important step toward standardizing practices in simulation scenario development for flight simulation applications.

## ABOUT THE AUTHORS

**Hung Tran** is a Technical Authority for Software Engineering at CAE USA Inc. He joined CAE USA more than 20 years ago and has worked on the modeling and simulation of several electronic warfare systems, weapon systems, and virtual environments. Hung holds a Bachelor of Science in Electrical Engineering from Polytechnique Montréal at University de Montréal in Quebec, Canada.

**Howard Cheung** is a Tactics Software Engineer at CAE USA Inc. He joined CAE USA 3 years ago and has worked on the Navy MH-60 R/S and S-70B programs. Howard holds a Bachelor of Science in Computer Science from the University of South Florida.

**Michael Tillett** is an Interoperability Software Engineer at CAE USA Inc. He joined CAE USA 10 years ago and has worked on the C-130J and KC-135 Distributed Mission Operations (DMO) programs. Michael holds a Bachelor of Science in Computer Engineering from the University of South Florida.

# An Ontology-based approach for Scenario Generation in Flight Simulation Systems

**Hung Tran, Howard Cheung, Michael Tillett**
**CAE USA Inc.**
**Tampa, FL**
**hung.tran@caemilusa.com; howard.cheung@caemilusa.com; michael.tillett@caemilusa.com**

## INTRODUCTION

Generating training scenarios for real-time simulators is currently a resource-intensive activity because the design of a flight simulator usually does not focus on generating and manipulating input data but instead on the execution of the simulation. Simulation-based training systems have become more complex in recent years. For example, a flight simulator is nowadays designed to operate in conjunction with external training resources, such as synthetic training environments, computer-generated forces, or networked flight simulators. Flight simulators are often connected with local or external training devices via distributed training networks, such as Distributed Mission Operations (DMO). Because of this added complexity, the effort required to generate and maintain training scenarios has increased. Developing training scenarios that induce a trainee to utilize specific skills is one of the facets of simulation-based training that requires significant effort. Conventionally, instructors have to build the scenarios manually each time, according to the level of a trainee, trainee class, and training objectives. To do so, the instructors spend much time and effort generating suitable scenarios appropriate to a trainee's objectives and competency level. For this reason, the creation of diverse scenarios has largely depended on the knowledge and experience of the instructors. Thus, the task of generating a variety of scenarios to support the required training curriculum occupied the majority of preparation time for instructors. Figure 1 illustrates the high-level scenario development process.

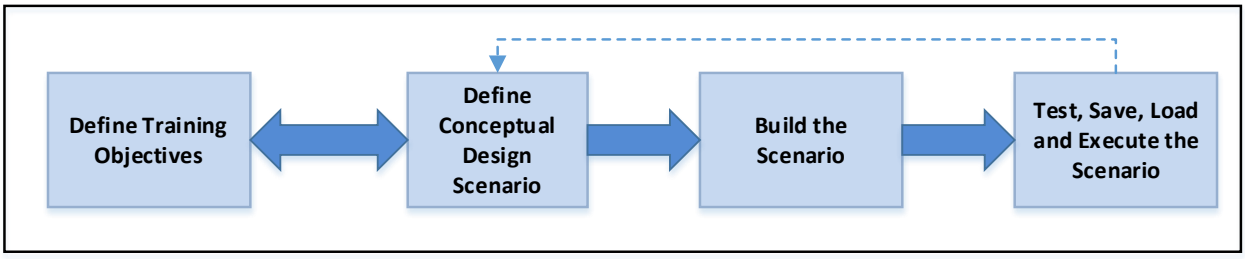


**Figure 1. Scenario Generation Process**

The first step of the process— Define Training Objectives—defines the training curriculum established for the training program. The second step—Define Conceptual Design Scenario—identifies key entities, events, and their interactions. Typically, the following three main components are defined in this step: (1) the initialization, (2) the key events that must happen during the training, and (3) the termination conditions. The next step—Build the Scenario—uses available tools and applications at their disposition. Instructors normally use the flight simulator embedded instructor operational system to create a training scenario for flight simulators. The entities, behaviors, and events that must happen during the training are defined during this step. The last step of the process—Test, Save, Load, and Execute the Scenario—tests the scenario in the simulator to ensure that all events execute as intended. The scenario can then be saved for future utilization.

The above process is challenging and time-consuming if performed manually. Therefore, developing an approach to completely or partially automate the scenario generation process has become required for flight simulator suppliers. Developing an effective automated scenario generation system in simulation requires prior knowledge of the training domain that the simulation environment was designed for, how the simulation components were designed, and how these components are related to each other. The behavior of the participating entities, whether they are virtual or constructive simulation entities, must also be understood by the instructors. This knowledge base should be logically well-predefined and represent the information in a generic, consistent, and unambiguous manner.

Because ontology analysis is an effective approach to constructing a robust knowledge-based system, we will explain the role of this ontological analysis in the process of creating training scenarios, especially for flight simulation.

## BACKGROUND

Although the importance of creating training scenarios in modeling and simulation has long been well known, a lack of common standardized practices in simulation scenario development still exists. Each simulation system developed their own approach to generate the required training scenarios. Additionally, after a scenario is developed, it will not be reusable in any training device other than the one on which it was developed. In other words, the scenario generation methods provided by a specific simulator are tightly coupled to this simulator and cannot be used to generate or manipulate scenarios for other simulators. For this reason, any process to generate training scenarios should be well defined, enabling the reuse of generated scenarios among different simulators. Another benefit of automated scenario generation is discussed in the study of Zook et al. (2012). The author noted that an automated scenario generation "can augment existing human-based scenario development practices to make more training scenarios available, thereby increasing the frequency at which training can occur in virtual environments."

Prior to developing a training scenario, training objectives must be analyzed to determine the training knowledge and skills required to ensure that scenario outputs are domain-valid and pedagogically effective. Automating the process is one way to facilitate the creation of training scenarios. The idea of "automated scenario generation" is not new, but because the automated scenario generation process represents a complex problem, no general solution currently exists. However, many efforts have been made toward developing limited functional systems.

### Scenario and State Abstraction

There are many ways to define a "scenario" and categorize the concepts of a "training scenario" to fit into a particular defined concept of a scenario. Often, these definitions conflict with each other as well as themselves, making it difficult to effectively rely on a true definition. For example, consider a simple "refueling scenario" where the intent is to describe all the aspects of "aerial refueling" to create the scenario as a template with the details filled in as parameters for specific training. An example of that could be the following:

- Refueling tanker aircraft (1x)
    - Type of refueling system (probe-and-drogue or flying boom)
- Refueling receiver aircraft (up to 2x)
- Location
- Flight parameters

This definition is sufficient for many aerial refueling scenarios. However, it is not fully compatible since, for example, certain tanker aircraft can have multiple refueling systems (not used simultaneously), and there are other methods of refueling helicopters (e.g., in-flight helicopter refueling) from a warship in the sea. As such, scenario definitions and their parameters can conflict, such as "flight parameters," even though they describe very similar training. Adding updates to the scenario or modifications to the definition itself can also lead to compatibility conflicts, which would either have to be resolved at a granular level (the implementation) or diverge to allow both to be utilized without much modification. The former resolution is time-consuming and potentially costly since implemented scenarios will have to be retroactively updated (and tested), while the latter does not promote compatibility across training platforms.

Challenges encountered with defining a scenario are mainly twofold: 1. designing the definitions to be abstract in a way that allows it to be dynamic enough in applicability while simultaneously retaining enough specificity to be utilized, and 2. allowing the implementation of the definition to incorporate incoming changes in a manner where, typically, minimal conflict is introduced (if any). As these are also the fundamental challenges in designing an ontology, the approach for our ontology is to abstract the concept of a scenario as well as its composition.

Conceptually, a scenario in the context of a training scenario can be abstracted to "a description of an overall evolution." It serves as a blueprint to contain concepts such as the synthetic environment, input parameters, training objectives, etc., to facilitate training. At its core, however, the composition of a scenario can be abstracted to a scenario state(s), where they themselves describe a subset of the evolution and contain the concepts specific to that state, and can facilitate the entry, progression, and transition of itself.

From the scenario state abstraction, we can expand the concept to be categorized into the following main types.

1. A Single (continuous) state is one where the initial parameters are given. That Single state defines the training scenario itself, the training parameters define objective parameters, and the exit parameters determine the end of the scenario. An example of this is if you are given a time and a location for a unit to attack, the state that would define that training scenario would be "unit attacking XYZ," where the scenario would remain in that state and play out, attempting to reach the objective parameters (i.e., to be successful). Once the final parameters are reached, the unit is either successful or not, the state is complete, and thus, so is the training scenario.
2. A Rigid (finite defined) state machine is one where the scenario states transition in a predefined order, given that the exit parameters for each scenario state are met. An example of this type of training scenario is if you attempt to teach a new student about the basic parts of a hose-and-drogue aerial refueling, the scenario states would be set up to model each part of the refueling mission. A scenario with rigidly defined states and transitions is useful when attempting to teach a particular aspect(s) of a scenario itself, and the transitions need to be predictable or controlled at the simulation level (i.e., to hold the scenario in a state until the instructor completes a lesson).
3. A Dynamic (finite defined) state machine is one where all of the scenario states are defined, but the transition to the next state can be dependent on which exit criteria or transition criteria are met. An example would be a patrol mission, where individual actions or discoveries could decide the next stage of the training scenario. Deciding to split up and patrol more areas can lead to new states, avoiding certain areas can ignore states, and introducing an unexpected action (like being ambushed) can reroute the next states entirely.
4. A Mixed-State machine can contain any combination of the above (i.e., dynamic state machine where decisions decide the next state) but ultimately leads to a continuous state to allow the simulation to react to training accordingly. An example would be having a dynamic state for a unit patrol. Still, most states will lead a template Single state of battle (the area and enemy dependent on which previous state they transitioned from), where the scenario does not progress further, regardless of the result of a battle.

Figure 2 conceptualizes the flow of three of the four types of scenario states.

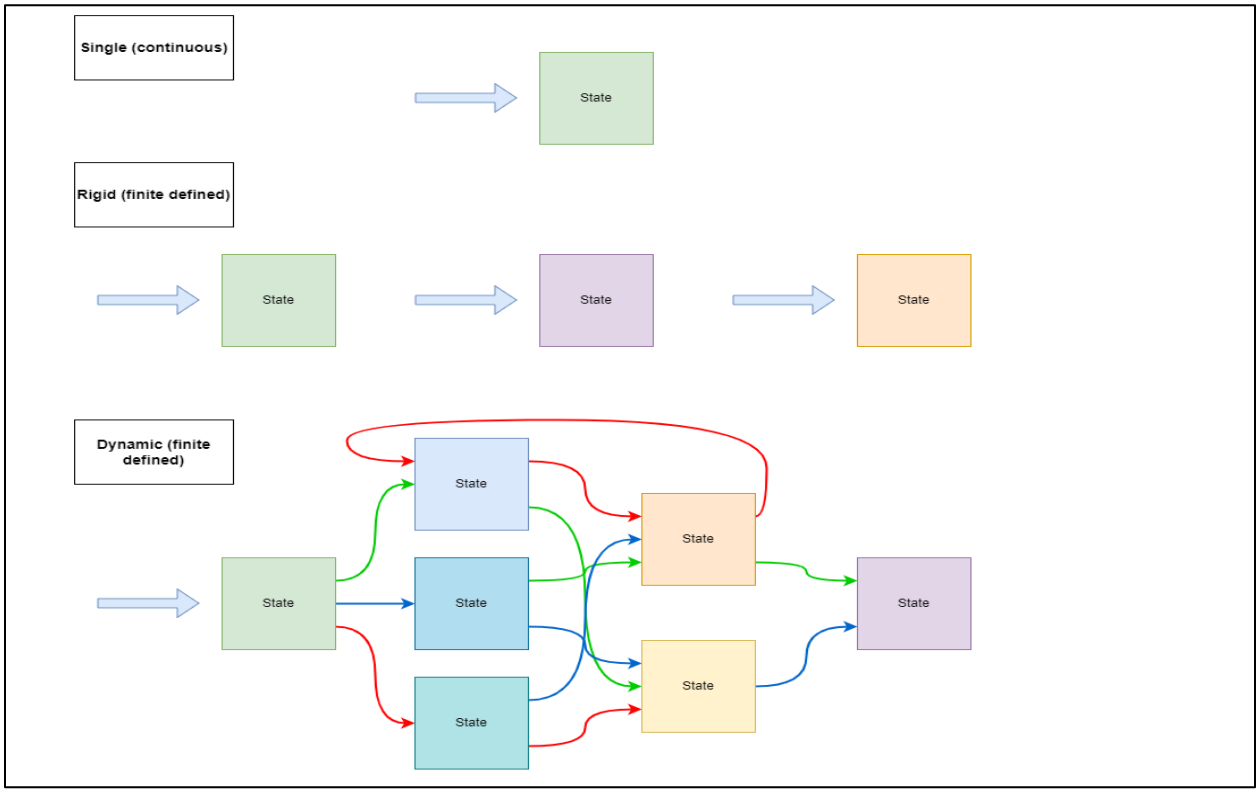


**Figure 2. Three of the Four Types of Scenario States**

**Past Works**

A literature review was conducted on the topics of automated scenario generation. (1) Loftin et al. developed the Training Scenario Generator for a NASA space shuttle flight-controller simulation. The system was designed to address the need to generate unique training scenarios for each trainee. The authors employed a production-rule expert system and an object-oriented database to define the scenario. This work represents a significant first step to automating the scenario generation process. (2) The Federal Aviation Administration (FAA) developed another scenario generation system for pilot qualification testing and training. This system was very specific to its own training objectives as it constructs its scenarios from collections of small, fixed scenarios that were FAA approved. (3) The Interactive Specification Acquisition Tools, or ISAT (Hall, 1998), developed by AT&T, used a heuristic approach to guide the assembly of a complete scenario from the compilation of smaller scenario pieces, each of which achieved sub-training objectives. (4) Oaks et al. (2003) proposed a methodology to generate air traffic scenarios for testing and analysis of Air Traffic Management decision support tools. Their method consisted of extracting the data from various and placing these data into relational database tables. The data was then manipulated to induce a conflict scenario, and finally, the actual scenario was created and generated based on the traffic data retrieved from the modified database tables. The final generated scenario was saved in ASCII format and could be recalled during training. (5) Pfefferman (1993) developed a prototype architecture to support the automatic generation of scenarios for combat simulations. His work focused on developing procedures that took a structured "mission file" as input to the scenario process. Knowledge Based Systems Inc. developed a knowledge-based simulation architecture. Their framework was based on training context and knowledge modeling methods to automate the generation of adaptive training scenarios for the United States Air Force. (6) Zook et al. introduced a "combinatorial optimization" approach to automate the scenario generation process. The approach presented in this study focused mainly on producing diverse training and quality scenarios while tailoring the scenarios to a particular learner's needs and abilities. This study's originality also resides in their proposed set of evaluation metrics to evaluate the overall quality of the scenarios.

In summary, although these previous efforts represent viable approaches toward an automated scenario generation system, there is a lack of standardization in terms of knowledge presentation of the training domain and the format of the generated scenarios. These present a main obstacle to enabling reusability among different simulators.

**Motivation for Using Ontologies in the Training Scenario Generation Process**

Ontology can be defined as a knowledge representation of a domain of expertise (Benjamin et al. 2006). It captures the concepts and structures them in a manner that people and machines can process without any ambiguity. An ontology typically consists of hierarchical arrangements of the knowledge into classes and subclasses pertinent to a specific domain of expertise (Peeters et al., 2014). A well-defined ontology can facilitate their use as models for learning the principles of the domain. Hilera et al. (2010) explained very well in their study the concept of using an ontology to present a domain of knowledge and the necessity of creating ontologies in the software domain to enable modeling, sharing, and reusing knowledge. The benefit of using ontologies for a scenario generation process for flight simulation applications is the ability to reuse the same ontology independently of the training platform. For instance, besides each aircraft's specific avionic equipment, aerial refueling training objectives and simulation elements should not differ much for a C-130J or a C-17 aircraft. Additional benefits of using ontology reside in their capacity to be easily extended to incorporate new knowledge generated by experts so all existing ontologies can be used as a starting point for further development (Hilera and Fernández-Sanz, 2010). An ontology for training scenarios could help improve training efficiency and effectiveness by providing a common framework for creating and sharing scenarios across different domains and organizations. It could also help to improve interoperability and integration between different training systems and tools.

## ONTOLOGY-BASED APPROACH FOR FLIGHT SIMULATION

The scenario generation process that we developed in this study possesses the following characteristics:

1- Based on a common and reusable ontology of knowledge presentation. An ontology for training scenarios would formally represent the knowledge and concepts related to training scenarios. It would help standardize the vocabulary and concepts used to create and share training scenarios across domains and platforms.
2- Scenario output in an open standard format that computers can process without any ambiguity (e.g., reusable).
3- Platform independent (i.e., can be used on any type of flight simulator).

Figure 3 illustrates the process of generating training scenarios for flight simulation using an ontology presentation.

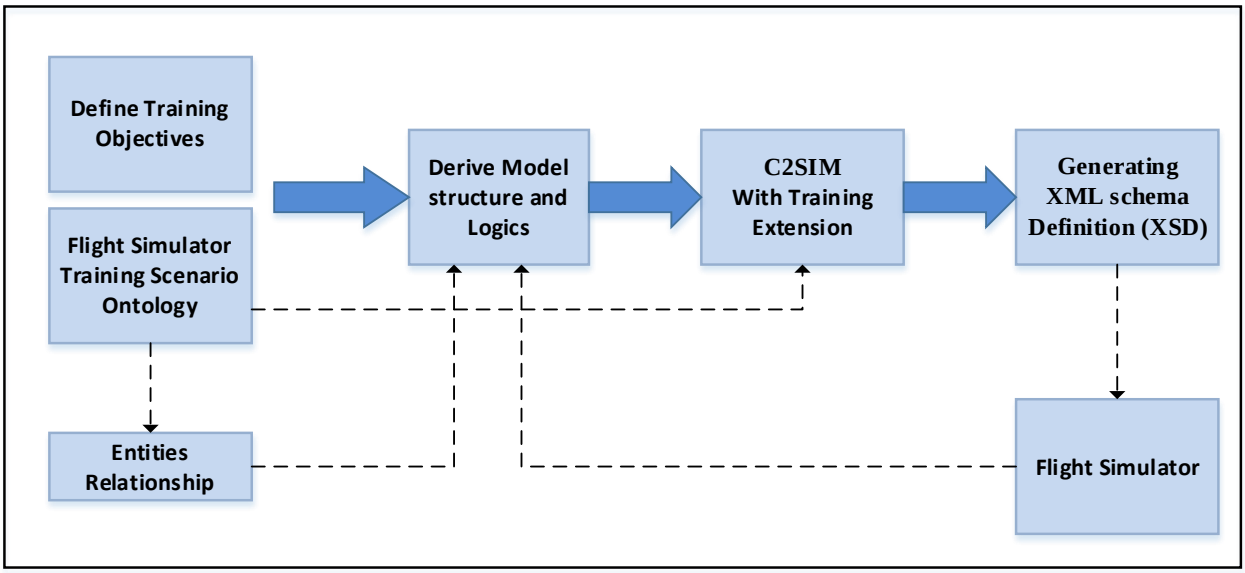


**Figure 3. Flight Simulator Scenarios Generation Process Using an Ontology Presentation**

**Define Training Objectives** – An instructor normally performs this step when using a conventional approach. Training objectives are usually derived from the training curriculum.

**Flight Simulator Training Scenario Ontology –** The ontology is the knowledge presentation of the domain. In this case, it represents a flight simulator's well-defined training capabilities. An ontology consists of classes and subclasses of the domain of knowledge. It was developed as a reusable domain reference model and is the heart of the scenario generation system facilitating knowledge sharing. Additionally, ontology provides the declarative knowledge needed to produce a scenario plan. Defining an ontology for training scenarios helps improve training efficiency and effectiveness by providing a common framework for creating and sharing scenarios across domains and platforms. Figure 4 shows the ontology of the newly defined flight simulator training scenario. The main class of Flight Simulator Training Scenario ontology is the "**TrainingScenarioContent**." At a high level, the Training Scenario ontology is structured as follows:

- **Scenario**: Defines all of the necessary objects and data to go through a training scenario from start to end
- **ScenarioCode**: Defines all of the enumerations that can be used through the "TrainingScenarioContent" similar to how C2SIMContent defined enumerations
- **ScenarioDescriptor**: Contains all of the objects that can be used to describe a Scenario, ScenarioState, or both
- **ScenarioObject**: Contains all of the objects that are required to create a training scenario

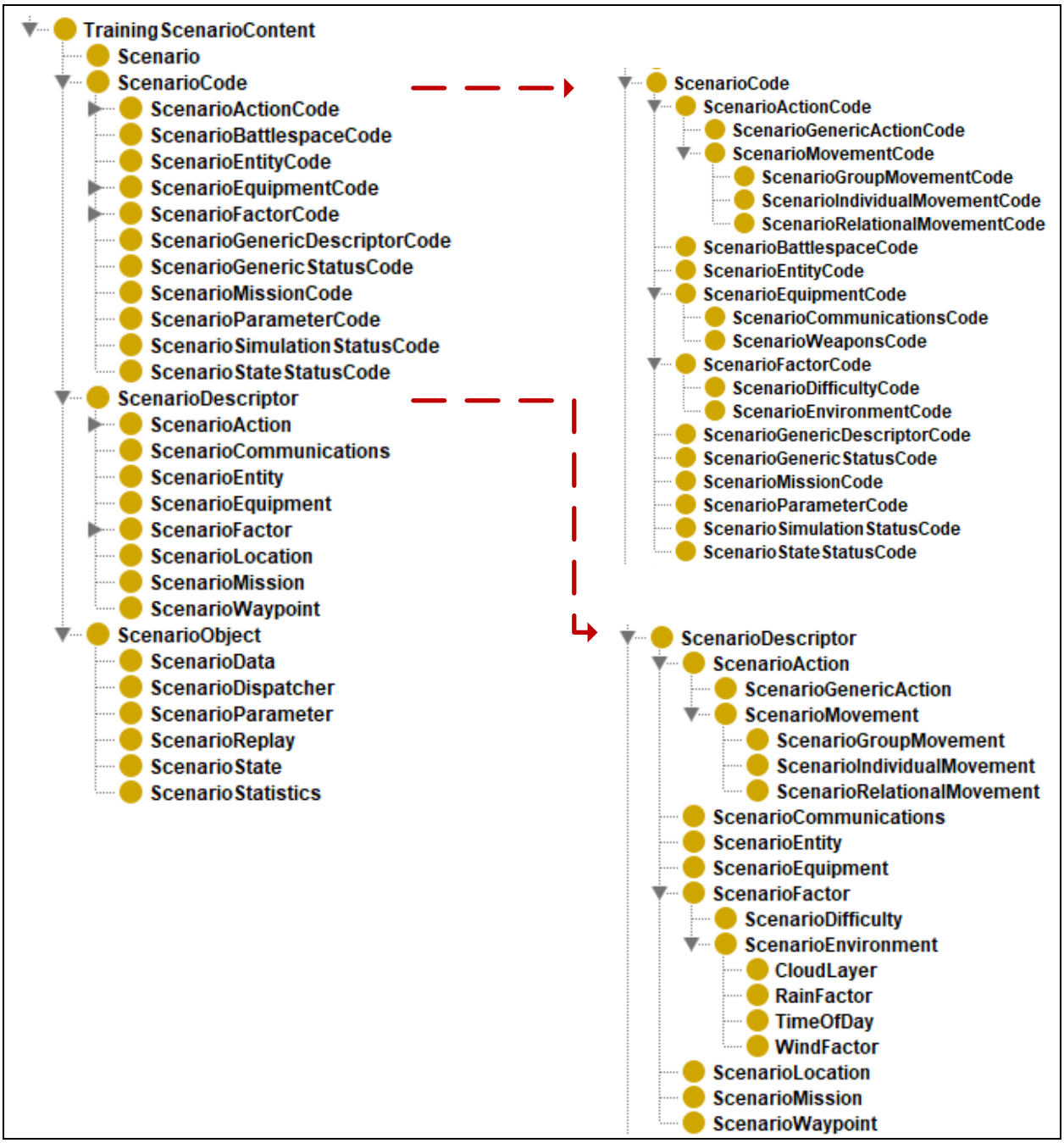


**Figure 4. Flight Simulator Training Scenario Ontology**

**Entities Relationship – As** part of the ontology, it describes the relationship between the elements of a class and its subclasses. Generally, an "entity-relationship diagram" is used to display the objects of the system and the relationships among them.

**Derive Model Structure and Logic –** The model structure and logic provides the relationship and dynamic behavior of the objects when they interact. For the case of a training scenario with a flight simulator, it defines the set of methods used by the objects during the execution and state transitions. An example of such dynamic behavior and state transition for an Air Refueling Training Scenario would be "*Aircraft tanker extends the boom*" when the aircraft receiver is in the connection position (i.e., ready to receive fuel).

**Command and Control Systems – Simulation Systems Interoperation (C2SIM) with Training Extension –** The SISO C2SIM (Command and Control Simulation Interface) is a standard for representing and communicating information related to military command and control. It is designed to facilitate the integration of various simulations and modeling tools used in the planning and execution of military operations. The C2SIM ontology represents an ontology designed to support the C2SIM framework. The ontology is used to represent the concepts and relationships that are important for modeling command and control operations. It includes a range of standards, such as data models, message formats, and protocols, which ensure interoperability between different systems (Singapogu et al., 2016).

Three classes define the high-level organization of the C2SIM ontology:

1. The **C2SIMContent** class is a collection that includes Action, Entity, and PhysicalConcept. Action includes Tasks, which are the key part of orders. Action also includes Event, which is named moments in time that may be referred to by other elements.
2. The **InitializationConcept** defines complex information elements contained in initialization messages. This class has several subclasses that describe the information exchanged in initialization messages.
3. The **MessageConcept** defines the structure of all C2SIM messages and the substructure of messages in the C2 domain (e.g., orders and reports). The key subclass in the MessageConcept collection is the Message class.

Using an ontology to generate flight simulator training scenarios can provide a structured and standardized way to represent the concepts and relationships involved in flight simulation. In this study, we propose adding a Training Scenario Extension, TrainingScenarioContent, defined for flight simulation training scenarios to the C2SIMContent class of the original C2SIM ontology. The first layer is the **Scenario class,** used to define the elements of a given training scenario, and the three classes ScenarioCode, ScenarioDescriptor, and ScenarioObject are used to build the elements of the Scenario class itself (Figure 5).

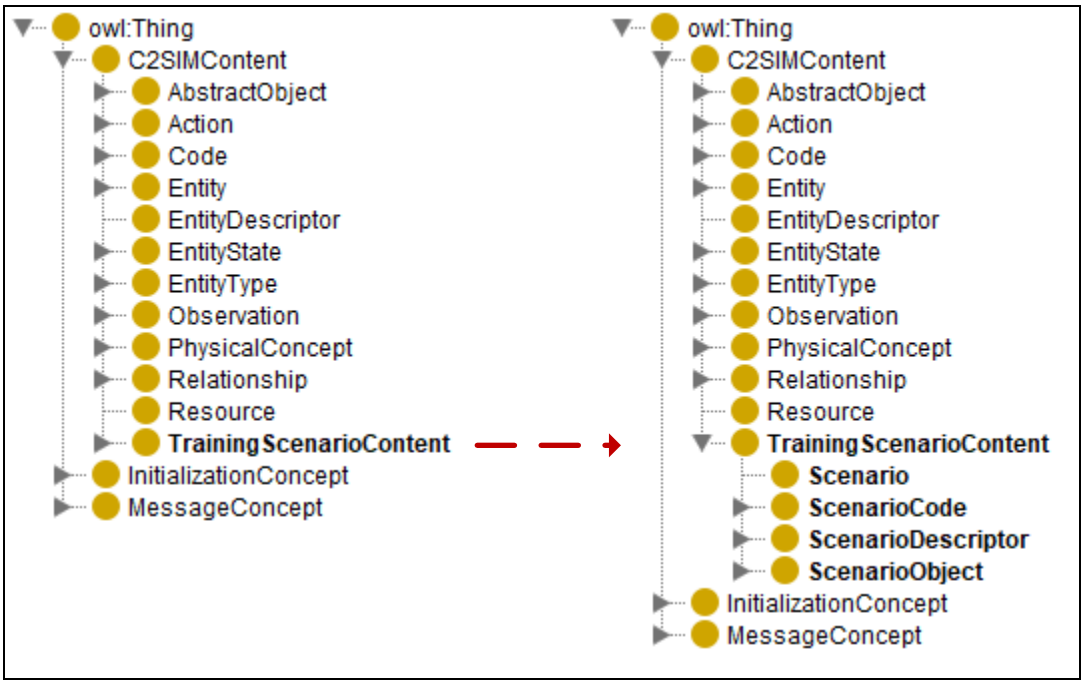


**Figure 5. Training Scenario Extension within C2SIM ontology**

**Generating XML Schema Definition (XSD)**

The C2SIM Standard provides a procedure for generating a physical data model XSD from the ontologies (the Core ontology plus extensions) by using the Extensible Stylesheet Language Transformations (XSLT) Processor. The XSD is a machine-readable file that provides the syntax and defines how elements and attributes can be represented in an XML document. This essentially represents a standard for defining an XML document - in this particular case, a

C2SIM scenario document. Because XSD Schema supports "extension," where one schema can extend from another schema (i.e., C2SIM), this is a great feature as it supports reuse and conformance to a standard.

By generating the XSD file from the C2SIM core ontology and the Training Scenario Extension, the "TrainingScenarioContent" is standardized across simulation platforms (i.e., platform agnostic), allowing flight simulators to access to a common definition and attributes of training scenarios. The process of generating the C2SIM (plus the TrainingScenarioContent) scenario XSD file is illustrated in Figure 6.

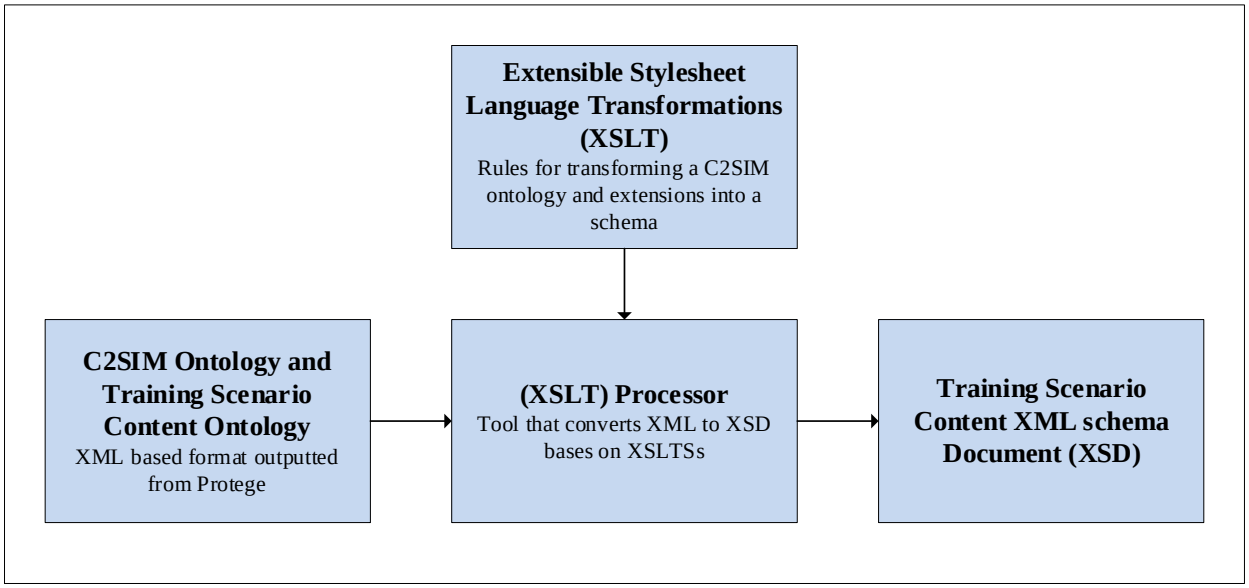


**Figure 6. Process of Generating the C2SIM XSD**

## AERIAL REFUELING CASE STUDY

To illustrate how the Flight Simulator Training Scenario ontology is used to create training scenarios, we present a case study of Air Refueling Training.

Aerial refueling consists of transferring fuel from one aircraft to another during flight. This training capability enabled the United States Air Force to reduce flying hours by shifting aerial refueling training to the simulators (Carretta & Dunlap, 1998). Therefore, the possibility of training aerial refueling using flight simulators represents a significant cost benefit because virtual refueling training saves money and requires far less support than live training.

Independent of the training platform (e.g., flight simulator), a scenario for air-to-air refueling (AAR) training generally requires the definition of the following simulation elements (Thomas et al., 2014):

- Type of aircraft tanker (e.g., KC-135)
- Type of aircraft receiver (e.g., MC-130J)
- Method of fuel transfer (e.g., flying boom or hose-and-drogue)
- Rendezvous approach
- Weather conditions
- Gaming area (i.e., where the training must happen)
- Unexpected events

Depending on the training objectives, the instructor must build appropriate training scenarios to conduct the training. A typical AAR training scenario can be defined as follows:

- The tanker KC-135 and the receiver C-130J aircraft perform an Rendez-Vous Echo AAR procedure.
- When the scenario is initialized, the constructive tanker heading is set at the receiver heading plus 90 degrees with a separation of 80 mi.
- The mission is at an altitude of 25kfts with a cross-wind of 20 knots and is set during the ARR procedure.

- The synthetic tanker is programmed to fly a racetrack pattern and return to the RV datum every 15 minutes.

Figure 7 illustrates graphically the definition of the above-described typical AAR scenario.

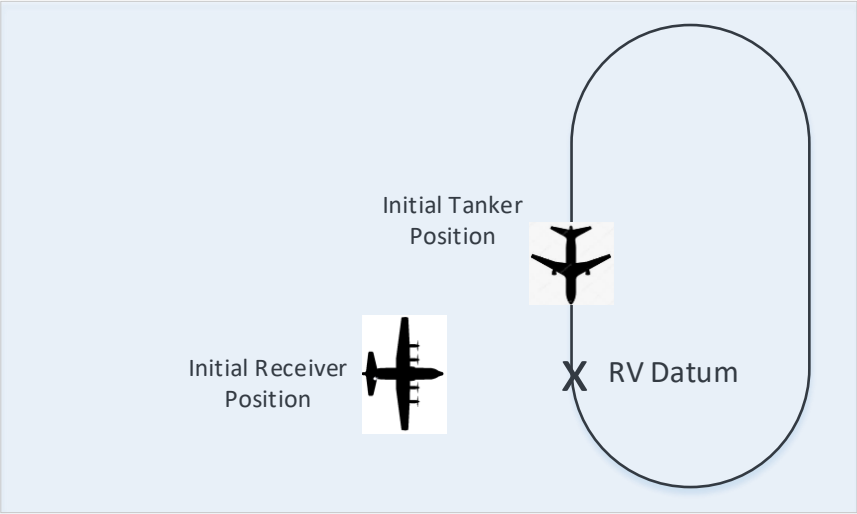


**Figure 7. Typical AAR Training Scenario**

**Define Training Objectives – Air Refueling Training Scenario**
The purpose of using a flight simulator as the receiver aircraft is to train the student pilots to perform the following (Tran and Tillett, 2021):

- Familiarity with AAR procedure to achieve close visual contact between receiver and tanker (i.e., rendezvous [RV] procedure, which could be established with just a single or a formation of tankers)
- Receiver position prior to and during the refueling operation
- Receiver sequence during the refueling operation
- Boom signal and tanker Pilot Director Lights (PDLs)
- Post-refueling and reform

**Training Scenario Ontology – Air Refueling Training Scenario**
For the described Air Refueling Training Scenario, we define the abstraction with the given parameters and target objectives:

- Overall scenario is to conduct refueling training
- The type of refueling is AAR (aerial refueling, more specifically)
- The entities involved are both aircraft consisting of:
  - One tanker to give fuel
  - One receiver to take fuel
- Overall location the scenario takes place is in the air (the battlespace of the scenario)
- Initial scenario conditions for the tanker are: already in the air, flying a continuous racetrack pattern
- Initial scenario conditions for the receiver are: at a military airport on the ground, waiting to take off
- Air refueling procedures:
  - The receiver performs take-off procedures.
  - Once the receiver is in stable flying condition, it flies toward the predetermined rendezvous location.
  - After visually identifying the tanker, it performs the specific rendezvous procedures and maneuvers into an offset position relative to the tanker to prepare for refueling.
  - The receiver performs the correct procedures pertinent to refueling and connects to the fueling system.
  - Once contact is achieved, the tanker transfers fuel to the receiver.
  - After fueling is complete, the receiver disconnects from the fueling system and prepares to depart away from the tanker.
  - The receiver performs the departing procedures.

- Once depart is complete, the receiver maintains its current flight parameters, and the scenario is complete.

We can use the defined abstraction to transform it into the following defined steps for the scenario to progress:

1. The tanker and receiver are in their respective starting locations.
2. The receiver takes off from the airport, flies to the tanker at predefined rendezvous, and performs a race track pattern.
3. The receiver then flies to a relative position, offset from the tanker, and perform AAR refueling procedures.
4. The tanker transfers a predetermined amount of fuel to the receiver through a refueling method.
5. Once the fuel transfer is complete, the receiver disconnects from the refueling system.
6. The receiver then performs departing procedures.

The above steps can now transform into defined scenario states for the training scenario using the Training Scenario ontology:

- **Scenario**: Air Refueling Training Scenario
    - ScenarioState: Initial
    - ScenarioState: FlyRendezvous
    - ScenarioState: Refueling
    - ScenarioState: Depart
    - ScenarioMission: Refueling mission
    - ScenarioSimulationStatusCode (dependent on the current state of simulation)

**Hierarchical Arrangement of Classes and Subclasses – Air Refueling Training Scenario**

With the defined scenario states, scenario parameters, and training objectives defined, the information for the scenario can be channeled into the TrainingScenario ontology to populate the pertinent fields and generate a scenario for an Air Refueling Training Scenario. At the top level for the ***Scenario*** class, we define the following relevant fields:

- ScenarioMission: RefuelingMission (enumeration)
- ScenarioSimulationStatusCode (changes according to the current state of the simulation)
- ScenarioState (contains the states as a collection of objects)

The four individual scenario states can be created with the scenario parameters and training objectives, which are added to the **Scenario** object as a **ScenarioState** object:

- **ScenarioState (Initial)**.
- **ScenarioState (FlyRendezvous).**
- **ScenarioState (Refueling).**
- **ScenarioState (Departing).**

With the Air Refueling Training Scenario generated using the ontology, we can visually show the hierarchical arrangement of the ontology's classes and subclasses in Figure 8.

With the ontological representation of the Air Refueling Training Scenario, the information can be parsed from the file to be understood and resolved as such:

- The **Air Refueling Training Scenario** is a Scenario object that contains the overall mission that it aims to train (Refueling), the overall environment in which the training will take place (air), and four scenario states (initial, flying to, refueling, departing). Each scenario state describes the battlespace (the first two states have two battlespaces because the receiver is on the ground and the tanker is in the air), the current mission of the scenario state (leveraging the ScenarioMission enumeration), and the ScenarioParameter class to elaborate on how each scenario state progresses.

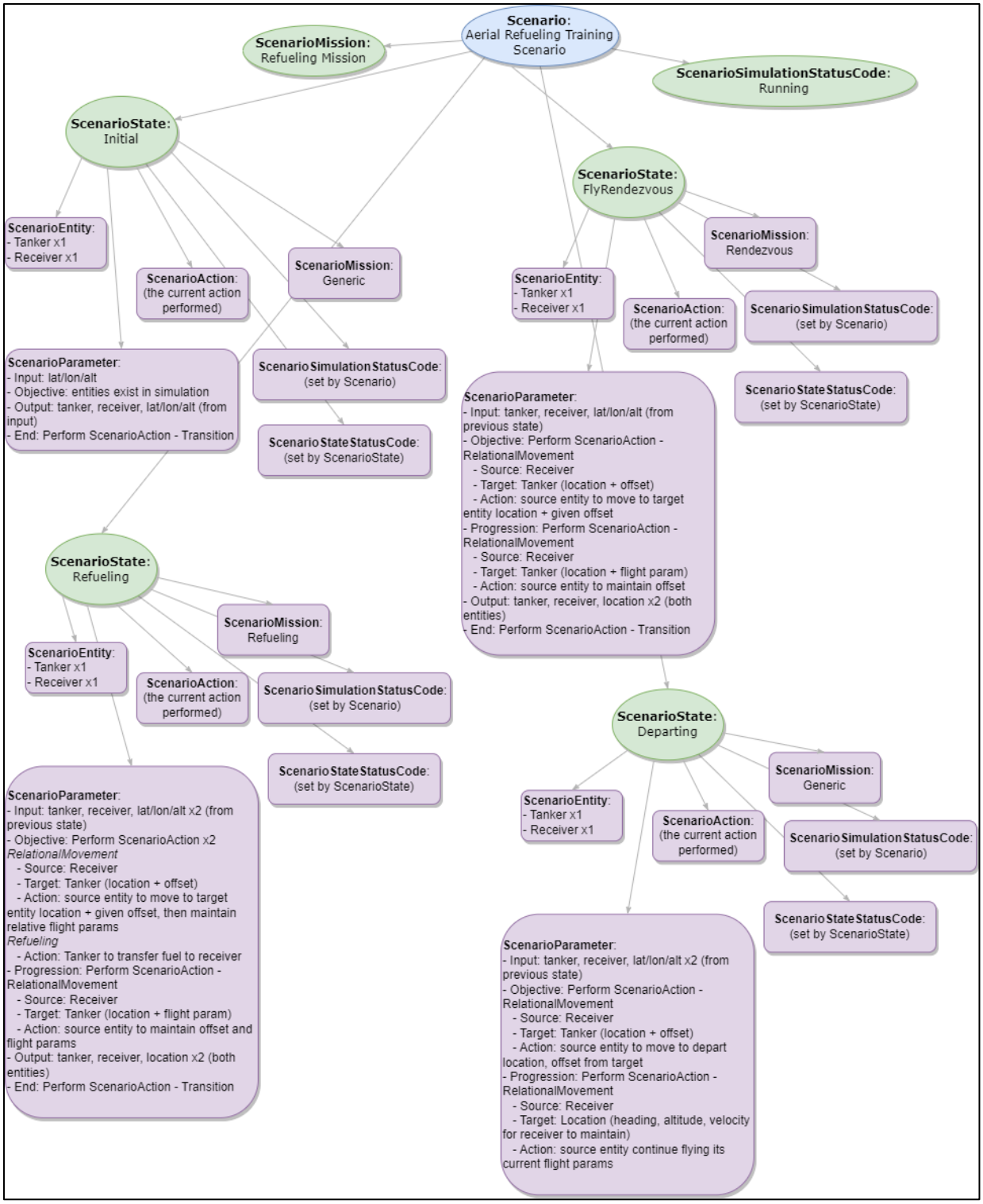


**Figure 8. Hierarchical Arrangement of Ontology Classes and Subclasses for the Air Refueling Case Study**

- The first state (**Initial**) sets up the rest of the scenario states by taking in a location (as lat/lon/alt). The objective parameter of the state is to ensure that both the receiver and tanker are defined (they exist). Once the objective parameter is met, it progresses to the next state (the End ScenarioParameter), and the output parameter outputs the two entities along with the location (in this case, the receiver location because it has to perform take-off).

- The second state (**FlyRendezvous**) refers to the receiver flying to the tanker to rendezvous, so the ScenarioMission changes (Rendezvous). The input parameter for the current state is the output of the previous state. The objective parameter defines the ScenarioMovement that needs to be performed in order for it to be met, which is to fly to the tanker aircraft location plus an additional offset from it. Once the objective parameter is met, the progression parameter defines the movement of the receiver aircraft (source) to maintain an offset position and speed relative to the tanker aircraft (target). Since the end parameter defines a state transition to the refueling state, the order that the parameters occur is as follows: objective parameter (continuous, until met) → objective parameter (continuous, until met) → progression parameter (set once) → end parameter.

- The third state (**Refueling**) refers to the refueling portion of the training scenario, so the ScenarioMission changes (Refueling). The input parameter is the same as the output parameter from the previous state, and the receiver aircraft starts by maintaining the positional offset and speed relative to the tanker. There are two objective parameters for this state (to be performed in series): the first is to have the receiver move to the fueling position, offset to the tanker (defined in AAR procedures), and the second is to transfer a certain amount of fuel. Once the objective parameters are met, the progression parameter sets the receiver aircraft to move to the departure offset location and maintain relative speed.

- The final state (**Departing**) has the receiver aircraft departing, and the ScenarioMission changes (Generic). The input parameter is the same as the output parameter from the previous state. The objective parameter directs the receiver aircraft (source) to move to a certain lat/lon/alt (target) away from the tanker aircraft. Once the objective parameter is met, the progression parameter directs the receiver aircraft (source) to maintain a certain heading, altitude, and velocity. The end parameter here is not defined for this particular scenario state because no further action is required, so this scenario state will have the same behavior as a Single (continuous) state and will flow in the following manner: objective parameter (continuous, until met) → progression parameter (continuous).

**Generate Training Scenario from XSD File**

Via the Instructor Operator Station (IOS), flight simulator instructors can load the XSD file generated previously to create a training scenario. The instructor decides what type of scenario to create (Single, Rigid, or Dynamic) and the scenario's training goals. After the scenario is created, it is saved as an XML file to be either updated later for future improvements or loaded later for training. The XML file is loaded via C2SIM to initialize the different players (virtual or constructive). Initializing could include position data, types of entities, and initial state. After all participants are initialized, the instructor can start the scenario for training. The process of using the XSD file to create and conduct a training scenario is illustrated in Figure 9.

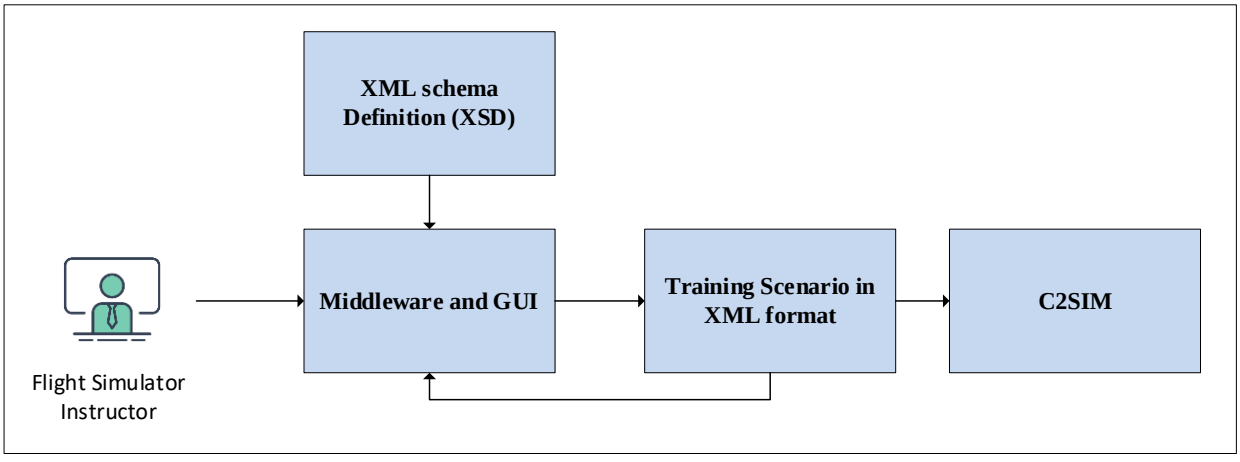


**Figure 9. Process of Generation of Training Scenario from the C2SIM XSD File**

At the run time (i.e., execute scenario), the C2SIM Controller (i.e., the Master) sends a message to all C2SIM applications letting them know a scenario will be starting. The flight simulator notifies the C2SIM marshaller declaring

the objects it supports (e.g., Synthetic tanker and/or Synthetic receiver, refueling system). The C2SIM Controller (through the instructor) sends the XML file to the marshaller, which then sends the initialization information to the flight simulator (e.g., location, type of entities). When the flight simulator is ready to execute the training scenario, the C2SIM Controller sends a start message, at which point the scenario is running. This process is illustrated in Figure 10. Depending on the type of scenario that was developed, the instructor would be able to monitor the state of the scenario and, using C2SIM, can command the scenario into progression. An example would be a synthetic tanker flying a racetrack pattern while waiting for the simulator to approach for refueling. The instructor would be able to progress the refueling scenario when the student approaches astern, including deploying the boom.

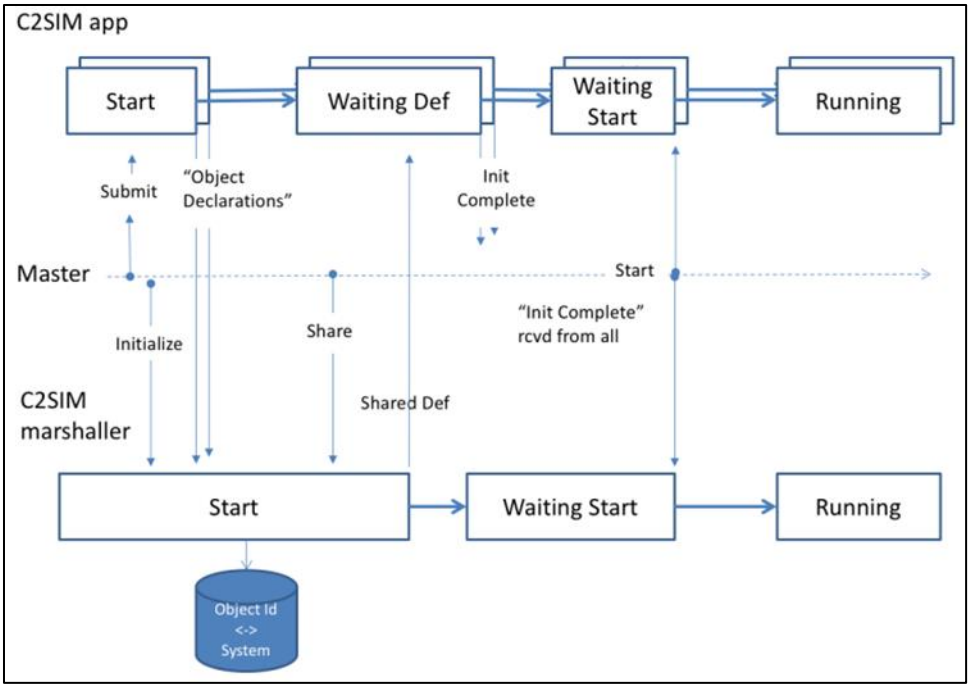


**Figure 10. Process of Initiating a Scenario Execution (from SISO-STD-019-2020)**

## CONCLUSION

The current state of practice in flight simulation suggests a need for a generation process that is reusable, consistent, and platform agnostic. In this paper, we presented an approach toward developing an automated scenario generation system. To do so, we defined an ontology pertinent to flight simulation training as an extension to the C2SIM ontology. Using this defined ontology, the XSD file was generated, and along with an appropriate middleware, an XML training scenario was generated. The output (i.e., training scenario) was formatted with the C2SIM Standard (e.g., XML), representing a common interchange format to produce initialization data for a platform-independent flight simulator. In this study, we demonstrated the utilization of the extension TrainingScenarioContent class for an Aerial Refueling training scenario. This same approach can be used for other type of training scenarios such as Search-and-Rescue (SAR), Formation Flying, etc.

Of the three high-level classes provided in the original C2SIM ontology, this study explores extending only the C2SIMContent class for the training scenario. The main reason that we did not introduce any extension to the InitializationConcept and the MessageConcept class is the majority of existing flight simulators already possess an Initial Condition (IC) system. The flight simulator IC system refers to the starting state of the training session. It encompasses various initial parameters that define the position, attitude, velocity, and other characteristics of the aircraft, as well as the initial state of the simulated systems. (e.g. Avionic Systems). Because the specific implementation of the initial condition system may greatly vary depending on the flight simulator software or hardware being used, we will need to identify and gather additional data pertinent to these IC systems. Therefore, we decided to defer the work of extending the InitializationConcept and the MessageConcept class of the C2SIM to a future study once we completed gathering additional data pertinent to flight simulators's IC systems.

Finally, the work presented in this paper constitutes an important step toward standardizing practices in training scenario development for flight simulation applications. Future development includes expanding this framework to additional simulation environments other than flight simulation.

**REFERENCES**

Benjamin, P., Patki, M. & Mayer, R. (2006). Using Ontologies for Simulation Modeling. Proceedings of the 2006 Winter Simulation Conference. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto. pp. 1151-1159.

Bowers, C., Jentsch, F., Baker, D., Prince, C., & Salas, E. (1997). Rapidly reconfigurable event-set based line operational evaluation scenarios. Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting, 4, 912–915

Erraguntla, M., Benjamin, P. C., Mayer, R. J. (1994). An architecture of a knowledge-based simulation engine. In J. D. Tew et al. (Eds.), Proceedings of the 1994 Winter Simulation Conference (673−680). San Diego, CA: Society for Computer Simulation International.

Hall, R. J. (1998, October). Explanation-based Scenario Generation for Reactive System Models. Paper presented at the 13th Conference on Automated Software Engineering, Honolulu, HI

Hilera, José, and Luis Fernández-Sanz. 2010. Developing Domain-ontologies to Improve Software Engineering Knowledge. International Conference on Software Engineering Advances.

Loftin, R. B., Wang, L. & Baffes, P. (1989, October). Intelligent Scenario Generation for Simulation-based Training. Paper presented at the 7th AIAA Computers in Aerospace Conference, Monterey, CA.

Loftin, R. B., Wang, L., Baffles, P., & Rua, M. (1987). An intelligent training system for payload-assist module deploys. In W. C. Chiou (Ed.), SPIE Proceedings, Vol. 851 (53−59). Bellingham, WA: SPIE.

Loftin, R., Wang, L., Baffles, P., & Hua, G. (1988). An intelligent training system for space shuttle flight controllers. Telematics and Informatics, 5(3): 151−161.

Peeters, M.M., Neerincx, K., Bosch, K. & Mayey, J. (2014). An ontology for scenario-based training. International Journal of Technology Enhanced Learning. Vol 6, No 3, pp 195-211.

Simulation Interoperability Standards Organization (2008). Military Scenario Definition Language Product Development Group, "Standard for: military scenario definition language," SISO, Orlando, FL, Rep. SISO-STD-007-2008, 2008.

Simulation Interoperability Standards Organization (2020). Command and Control Systems – Simulation Systems Interoperation Product Development Group "Standard for Command and Control Systems - Simulation Systems Interoperation," SISO, Orlando, FL, Rep. SISO-STD-019-2020, 2020.

Singapogu. S. S., Gupton. K., Schade. U. (2016). The role of Ontology in C2SIM. International Command and Control Research and Technology Symposium (ICCRTS).

Thomas, P. R., Bhandari, U., Bullock, S., Richardson, T.S, du Bois, J.L. (2014). Advance in air to air Refueling. Progress in Aerospace Sciences. 71, 14-35.

Tran, H and Tillett, M (2021). An Emulation of a Flying Boom Operator Using a Rule-Based Expert System. Interservice/Industry Training, Simulation, and Education Conference Proceeding (I/ITSEC).

Zook, A., Lee-Urban, S., Riedl, M. O., Holden, H. K., Sottilare, R. A., & Brawner, K. W. (2012, May). Automated scenario generation: toward tailored and optimized military training in virtual environments. In Proceedings of the international conference on the foundations of digital games (pp. 164-171).