Towards a Common Air Traffic Control Simulation Scenario Development

Shafagh Jafer, Jessica Updegrove, and Bharvi Chhaya[1]

[1] Software Modeling and Simulation Lab, Embry-Riddle Aeronautical University
Daytona Beach, FL, USA
jafers@erau.edu, chhayab@my.erau.edu, updegroj@my.erau.edu

Abstract. Although the importance of scenarios in Air Traffic Control (ATC) training and simulation has long been well known, there still exists a lack of common understanding and standardized practices in scenario development in this domain. It is an extensive process beginning with the stakeholders' descriptions of the scenario and finishing with the generation of the corresponding executable specifications. The purpose of developing an ATC Scenario Language is to implement a standard scenario specification that will lead to a common mechanism for specifying, verifying and executing ATC scenarios, effective sharing of scenarios among various simulation and training environments, improve the consistency among different training modules, and enable the reuse of scenario specifications. The proposed ATC Scenario Language is built as an extension to the newly published Aviation Scenario Definition Language (ASDL), complementing ASDL to support ATC scenario specification.

1 Introduction

Despite the promotion of reuse in the modeling and simulation (M&S) community, the aviation simulation community has struggled to create a standard for defining aviation simulation scenarios that can be shared amongst simulators. The aviation simulation community is plagued by complex, black-box simulation software development which requires years of expertise to create and perfect, creating simulation scenarios that can only be utilized by one target simulator. As a result, hundreds of unique implementations for the same aviation concepts have been developed with no means of reusing or sharing these ideas. The aviation simulation community needs to promote reuse and standardization of common aviation simulation scenario definition to reduce complexity, time, and costs associated with developing aviation simulation environments. Almost all aviation simulators utilize some form of aviation scenario, whether it be an aircraft landing simulation in a part-task trainer or a complex enroute scenario in a high-fidelity ATC simulator. This work presents an extension to the recently proposed Aviation Scenario Definition Language (ASDL) [13] to provide a standard for formally specifying Air Traffic Control (ATC) training scenarios. With the ASDL as a scenario specification standard, aviation scenarios can be shared amongst all types of aviation simulators, providing much needed reuse possibilities

and cost savings. ASDL is on its way to apply to become a Simulation Interoperability Standards Organization (SISO) [16] standard and become the first standard for the aviation simulation community in the area of scenario specification.

ATC training programs are highly reliant on the use of simulation and scenario-based training, providing trainees a close-to-reality environment to prepare for the job at the ATC facilities. One of the challenges of ATC training domain is the cost and complexity of training simulators. Despite the fact that such simulators provide high-fidelity environments where instructors can create and deploy practice scenarios, it is rather complicated and time consuming to generate a rich pool of practice scenarios. This is mainly due to the deep gap between domain experts (ATC instructors) and simulation technologies, requiring a delicate specification of scenario details in a fashion that can be translated into executable scripts. To address this challenge, we have extended the recently proposed ASDL effort to include ATC scenario specification capabilities. By utilizing a domain-specific language, a common standard is created that allows domain experts to easily provide scenario details using a simple graphical user interface, while hiding all the language and simulation details. This approach, not only provides a standard scenario specification platform, but also enforces scenario consistency and completeness checking, ensuring that all necessary scenario elements are captured and ready to be used by simulation experts to produce executable scenario scripts. Concepts such as Model-Driven Engineering, Scenario-based Development, and Ontology-Driven Modeling are the core methodologies utilized by ASDL. This chapter will introduce ASDL, provide background details on such concepts, and discuss details of extending ASDL to support ATC scenario modeling. A sample ATC scenario will be presented to illustrate the application of ASDL and automated script generation will be utilized to create executable ATC scenario script for a target simulator.

2 Background and ASDL Overview

A Domain-Specific Language (DSL) [7] is a custom tailored computer language for a particular application domain. DSL is created to specifically target problems in a specific domain, and stresses upon the main ideas, features, constraints, and characteristics of that domain. DSL enables developers to construct models that are specific to their application. These models are mainly composed of elements and relationships that are verified to be valid for that application. One of the greatest benefits of DSL is allowing non-developers and people who are not experts in the domain to understand the overall design. This is normally supported by allowing graphical modeling, usually in the form of a drag and drop capability to construct models. DSL augmented with model-to-text transformation capabilities directly allows for automatic generation of source code from model.

An ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as a computerized specification of the meaning of terms used in the vocabulary [18]. One of the benefits of using ontologies is their capacity to be easily extended using new knowledge generated by experts so all existing ontologies can be used as a starting point for further de-

velopment [8]. Among the existing ontology specification frameworks, the Web Ontology Language format (OWL) is most commonly used by the DSL community. OWL enables describing a domain in terms of classes, properties and individuals and may include rich descriptions of the characteristics of those objects [2].

Ontology-Driven Software Development (ODSD) has emerged as a significant mechanism in creating domain-specific languages [4], allowing for expressing domain concepts effectively. Ontology provides a quick and simplified description of a DSL, abstracting language's technically details, while highlighting key terminology and specifics. Once an ontology is built, it is a simple process to generate the language's metamodel and establish relationships among related concepts. An automated process that takes in DSL's ontology and generates its corresponding metamodel sounds highly efficient.

2.1 Aviation Scenario Definition Language (ASDL)

Introduced in 2016, the Aviation Scenario Definition Language (ASDL) [13] adopts Scenario-Driven Engineering to develop a DSL to express flight scenarios. ASDL was proposed to standardize aviation scenario development to allow the global aviation Modeling and Simulation (M&S) community, from academia and industry to government, benefit from a common scenario definition platform, enabling model transformation, reusability, and interoperability across various simulation environments. A scenario undergoes three development stages: operational, conceptual, and executable. Following the principles of model-driven engineering, scenario development takes place as the transformation of operational scenarios (defined in a natural language) to conceptual scenarios (conforming to ASDL formal metamodel) then to executable scenarios (specified using ASDL scenario definition).

ASDL takes scenario-based development as the core activity in constructing a formal scenario definition language. Concepts such as ontology-based and model-based development have been highly utilized to incorporate automated transformations and executable simulation script generation [15]. At the core of every DSL exists an ontology that captures all the domain's key terminology and relationships. The elements of model-driven methodology are modelling languages, metamodels and transformations [3]. Modelling languages enable the definition of a concrete representation for a model, while metamodels are used to define modelling languages. Transformations are described as the mappings between models, which are specified at metamodel level.

Compared to general-purpose languages, DSLs allow users to write complete application programs for the given domain more quickly and effectively by mostly relying on master reference sheets and published procedures about that domain. Hence, DSL ends up capturing precisely the semantics of the application domain [9]. As such, serving as a DSL language, ASDL allows non-developers and those who are not experts in the domain to communicate effectively and document a complete scenario understandable by both sides. This is supported by ASDL's graphical modeling capability in the form of drop-down menu utilities to construct scenario models [12].

The current version of ASDL fully supports four flight operations: departure, reroute, enroute, and landing. ASDL ontology [1] captures all scenario details through

its extensive OWL-based definition [14]. Following the well-known DSL evolution property, ASDL is deemed to undergo additional expansions to include new concepts such as those mandated by the Federal Aviation Administration (FAA) NextGen program [6]. The key challenge is continuous conformance of already constructed ASDL scenarios to the language metamodel (syntax) while it evolves. Figure 1 illustrates ASDL approach in turning domain ontology into DSL syntax, then allowing scenario model development graphically, and finally automated generation of simulation script for target simulators.

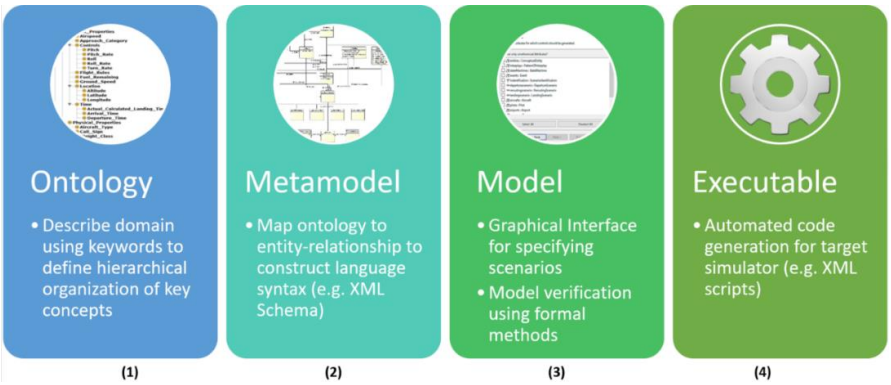

Figure 1. Scenario Development with ASDL: from Ontology to Executable.

Expansion of ASDL happens in two phases. First, ASDL ontology is augmented with new terminology to allow for expressing new types of scenarios or new elements for existing scenarios. Second, ASDL metamodel is updated to utilize new ontology entities and establish relationship with the existing language syntax. This process can be conducted iteratively, addressing new extensions and language evolution systematically while ensuring existing scenarios are not violated against the updated metamodel. Figure 2 illustrates this process.

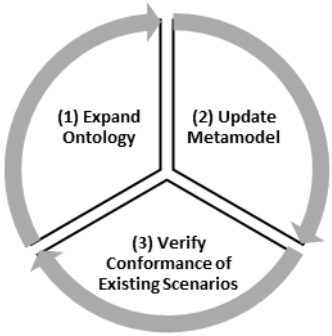

Figure 2. ASDL Continuous Evolution.

To better illustrate scenario exploration capability of ASDL, let's consider a typical flight operation scenario. A flight scenario is first provided by a domain expert, such as a pilot or controller, in natural language. It is basically a non-technical de-

scription of series of events that express an operation such as landing or departure. Such scenario is non-formal, hence, lacking technical aspects and detailed description of every event's precondition, postcondition, and actions. It is the duty of a simulation expert to take the scenario description from this level and generate an executable for a target simulator. This requires series of elaboration and validations by the domain expert to ensure accuracy and common understanding of the scenario elements. ASDL significantly resolves these hurdles by providing domain experts an easy-to-use graphical environment where all scenario properties can be entered (or selected from a set of options). Such framework not only reduces human errors in specifying scenario details, but also, complements operational scenarios by reminding the domain expert about certain missing field that must be fulfilled in order to capture the scenario fully. As such, scenario verification and validation can be applied by ASDL automatically, simplifying the simulation expert's task and utilizing various automation techniques such as field checking, syntax conformance to scenario metamodel, and most rewarding, generating executable simulation scripts.

As covered in details in [13] and [12], ASDL is developed in Eclipse Modeling Framework (EMF) [17] and utilizes EMF Forms to provide a graphical user interface for specifying scenario elements. Sample GUI screenshots from specifying a landing scenario are illustrated in Figure 3.

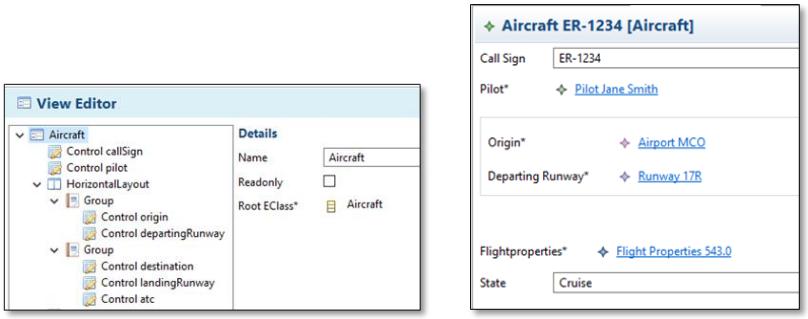

Figure 3. ASDL Scenario Specification GUI Environment.

3 ATC Integration in ASDL

There are three major flight control divisions in ATC and ATC training: Tower, Terminal Radar Approach Control (TRACON), and Enroute. Each division has a unique training track, with each track specifically designed to teach the required materials for only one division; however, all three tracks follow the aforementioned training phases. The Tower track focuses on air traffic management activities within a radius of a few miles of the airport and utilizes simulators to replicate real ATC towers and airport views. The TRACON track is comprised of the job jeopardy Terminal Basic Radar Training Course (RTF) for developmentals proceeding to a standalone radar facility. This course incorporates classroom and simulation training focused on managing traffic outside the radius managed by Towers, generally extending only a 40-mile radius from the primary airport.

To include ATC communication in ASDL, first ATC ontology needs to be defined and added to ASDL. The ATC terms added to the ontology were divided into three categories: Actions, Entities and Events. Actions are responses to the Events, and are performed by or through the use of one or more entities. Entities are objects or beings present in the physical world that interact with each other and are modeled within the scenario. Events are triggered proceedings that occur either with time – for instance, departures or arrivals, or when favorable conditions exist – for instance, yellow alert when separation between any two aircraft is between five to twelve miles.

The ATC entities, events and actions described can be seen in the ASDL Ontology Figure 4, written in Web Ontology Language (OWL). Entities are on the left and include the Air Route Traffic Control Center (ARTCC), Aircraft, Airport, Airspace, Controller and Weather. Some of these are broken down further, and an explanation of each term can be found in Table 1. Only those terms that relate directly to enroute control are presented here, as entities such as an aircraft and airport have already been defined in other works on ASDL [13] and [10]. Events can be seen in the middle of the figure and include arrivals, departures, holding, (Inappropriate Altitude for Direction of Flight) IAFDOF and the various alerts. A description of each of these terms is also available in the table. Actions are shown on the right side of the figure and are performed by the controller. Some examples of actions are: making point outs, handoff, route changes and responding to requests. A description of these can also be found in Table 1.

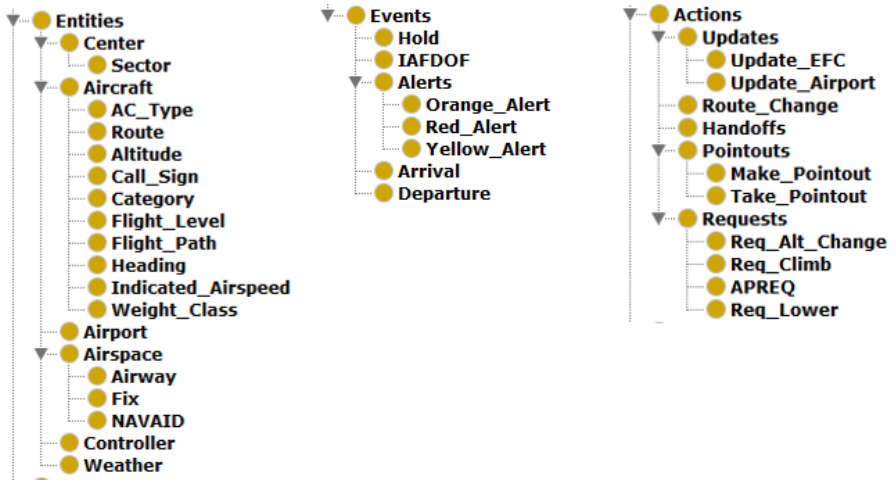

Figure 4. ATC Ontology Capturing Enroute Concepts.

Table 1. Definition of elements added to the Enroute Ontology.

Type	Element	Definition
Entities	Center	Air Route Traffic Control Center, or ARTCC
	Sector	A sector within the ARTCC
	Airspace	The collection of all space available for aircraft to fly within.
	Airway	A Class E airspace area established in the form of a corridor, the centerline of which is defined by radio navigational aids.

	Fix	The geographical position determined by visual reference to the surface, by reference to one or more radio NAVAIDs, by celestial plotting, or by another navigational device.
	NAVAID	Any visual or electronic device airborne or on the surface which provides point-to-point guidance information or position data to aircraft in flight.
	Controller	A person authorized to provide air traffic control services.
Events	Hold	A predetermined maneuver which keeps aircraft within a specified airspace while awaiting further clearance from air traffic control.
	IAFDOF	Inappropriate Altitude for Direction of Flight.
	Alerts	A notification that the controller must pay attention to an event, usually relating to separation of aircraft.
	Orange_Alert	Aircraft to airspace conflict.
	Red_Alert	Aircraft to aircraft conflict. Occurs when separation is less than standard separation (5 miles/1000 feet).
	Yellow_Alert	Aircraft to aircraft conflict. Occurs when separation is greater than standard separation (5 miles), but within detection range (5-12 miles).
	Arrival	The act of an aircraft touching down.
	Departure	The act of an aircraft becoming airborne.
Actions	Updates	Any changes made by the controller to an aircraft's flight characteristics.
	Update_EFC	An update to a prior "Expect Further Clearance" message.
	Update_Airport	An update to an airport (usually arrival) due to unfavorable circumstances.
	Route_Change	A change to the route, either by request or prompted by the controller due to weather, separation or other circumstances.
	Handoffs	The act of transferring radar and radio control of an aircraft to another sector.
	Pointouts	An action taken by a controller to transfer the radar identification of an aircraft to another controller if the aircraft will or may enter the airspace or of another controller and radio communications will not be transferred.
	Make_Pointout	The act of making a point out to another controller.
	Take_Pointout	The act of receiving a pointout from another controller.
	Requests	Any request made by a pilot or a controller which requires the approval of a Controller.
	Req_Alt_Change	A request for an altitude change that needs to be approved or rejected based on traffic conditions.
	Req_Climb	A request for a change to higher altitude that needs to be approved or rejected based on traffic conditions.
	APREQ	An approval request, usually between two controllers, that asks for something unusual or unnatural to be allowed, such as an aircraft travelling in an IAFDOF.
	Req_Lower	A request for a change to lower altitude that needs to be approved or rejected based on traffic conditions.

The ATC enroute extension requires the inclusion of airspace and how aircraft move through different parts of an airspace. This required a variety of changes to the original ASDL ontology, which previously did not extend beyond a tower controller and airport airspace. As a result, a large amount of new terms were needed, especially in the ATC entity. In addition, a new Airspace entity was added in order to capture the movement of the aircraft through different parts of an airspace during the enroute phase.

Figure 5 shows the enroute ontology terms added to the Air Traffic Control entity of the ASDL ontology. Table 2 defines each enroute term seen in Figure 5. These terms are used by air traffic control when communicating with an aircraft in the en-route phase and provide additional control options. For example, the Traffic Alert term is not required for every aircraft in the enroute phase but can be used by ATC if another aircraft may break the separation minima required between aircraft.

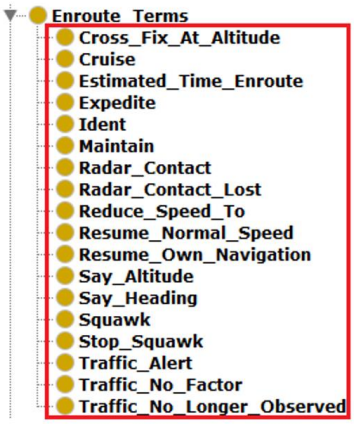

Figure 5. ATC Additional Enroute Terminology.

Table 2. ATC Additional Enroute Terminoloy Definition.

Term	Definition
CROSS FIX AT ALTITUDE	Used by ATC when a specific altitude restriction at a specified fix is required.
CRUISE	Used in an ATC clearance to authorize a pilot to conduct flight at any altitude from the minimum IFR altitude up to and including the altitude specified in the clearance.
ESTIMATED TIME ENROUTE	The estimated flying time from departure point to destination (lift-off to touchdown).
EXPEDITE	Used by ATC when prompt compliance is required to avoid the development of an imminent situation.
IDENT	A request for a pilot to activate the aircraft transponder identification feature.
MAINTAIN	Concerning altitude/flight level, the term means to remain at the altitude/flight level specified by ATC.
RADAR	Used by ATC to inform an aircraft that it is identified on the radar dis-

CONTACT	play and radar flight following will be provided until radar identification is terminated.
RADAR CONTACT LOST	Used by ATC to inform a pilot that radar data used to determine the aircraft's position is no longer being received, or is no longer reliable and radar service is no longer being provided.
REDUCE SPEED TO	An ATC procedure used to request pilots to adjust aircraft speed to a specific value for the purpose of providing desired spacing.
RESUME NORMAL SPEED	Used by ATC to advise a pilot to resume an aircraft's normal operating speed.
RESUME OWN NAVIGATION	Used by ATC to advise a pilot to resume his/her own navigational responsibility.
SAY ALTITUDE	Used by ATC to ascertain an aircraft's specific altitude/flight level.
SAY HEADING	Used by ATC to ascertain an aircraft's specific heading.
SQUAWK	Used by ATC to request a pilot activate specific modes/codes/functions on the aircraft transponder.
STOP SQUAWK	Used by ATC to tell the pilot to turn specified functions of the aircraft transponder off.
TRAFFIC ALERT	A safety alert issued by ATC to aircraft under their control if ATC is aware the aircraft is at an altitude which, in the controller's judgment, places the aircraft in unsafe proximity to other aircraft.
TRAFFIC NO FACTOR	Indicates that the traffic described in a previously issued traffic advisory is no factor.
TRAFFIC NO LONGER OBSERVED	Indicates that the traffic described in a previously issued traffic advisory is no longer depicted on radar, but may still be a factor.

Figure 6 shows the Aircraft class updated with the enroute terms. These terms define aircraft-specific components during the enroute phase of flight, such as Miles-in-Trail, which is used by ATC for enforcing separation minima between two cruising aircraft. The new terms included through the enroute extension are defined in Table 3; all other terms were already present in the original ASDL implementation.

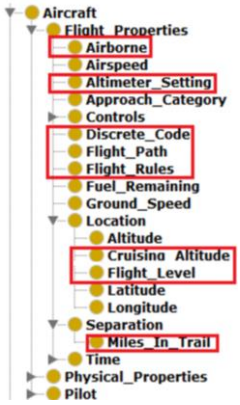

Figure 6. Aircraft Enroute Terminology.

Table 3. Aircraft Enroute Ontology Definitions.

Term	Definition
AIRBORNE	An aircraft is considered airborne when all parts of the aircraft are off the ground.
ALTIMETER SETTING	The barometric pressure reading used to adjust a pressure altimeter for variations in existing atmospheric pressure or to the standard altimeter setting (29.92).
CRUISING ALTITUDE	An altitude or flight level maintained during enroute level flight. This is a constant altitude and should not be confused with a cruise clearance.
DISCRETE CODE	As used in the Air Traffic Control Radar Beacon System (ATCRBS), any one of the 4096 selectable Mode 3/A aircraft transponder codes except those ending in zero zero.
FLIGHT LEVEL	A level of constant atmospheric pressure related to a reference datum of 29.92 inches of mercury. Each is stated in three digits that represent hundreds of feet. For example, flight level (FL) 250 represents a barometric altimeter indication of 25,000 feet; FL 255, an indication of 25,500 feet.
FLIGHT PATH	A line, course, or track along which an aircraft is flying or intended to be flown.
FLIGHT RULES	The set of rules that govern the flight: VFR or IFR. VFR stands for Visual Flight Rules and IFR means Instrument Flight Rules.
MILES IN TRAIL	A specified distance between aircraft, normally, in the same stratum associated with the same destination or route of flight.

Figure 7 shows the new Airspace class. This class supports the movement of an aircraft through an associated airspace by defining the airspace components, such as routes and fixes. Table 4 defines the terms shows in Figure 7. Note that Route is not included as a term and instead acts only as a means of categorizing all subclassifications of routes and the terms related to routes.

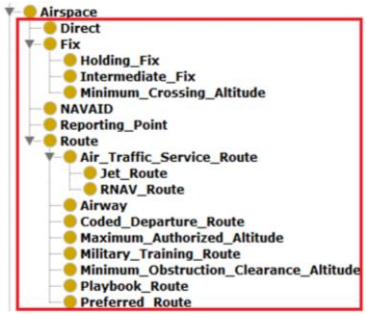

Figure 7. Airspace Ontology Terminology.

Table 4. Airspace Ontology Definitions

Term	Definition
AIR TRAFFIC SERVICE ROUTE	A generic term that serves only as an overall title when listing the types of routes that compromise the United States route structure.
AIRWAY	A Class E airspace area established in the form of a corridor, the centerline of which is defined by radio navigational aids.

CODED DEPARTURE ROUTE	A combination of coded air traffic routings and refined coordination procedures, designed to reduce the amount of information that needs to be exchanged between ATC and flight crews.
DIRECT	Straight line flight between two navigational aids, fixes, points, or any combination thereof.
FIX	The geographical position determined by visual reference to the surface, by reference to one or more radio NAVAIDs, by celestial plotting, or by another navigational device.
HOLDING FIX	A specified fix identifiable to a pilot by NAVAIDs or visual reference to the ground used as a reference point in establishing and maintaining the position of an aircraft while holding.
INTERMEDIATE FIX	The fix that identifies the beginning of the intermediate approach segment of an instrument approach procedure.
JET ROUTE	A route designed to serve aircraft operations from 18,000 feet MSL up to and including flight level 450.
MAXIMUM AUTHORIZED ALTITUDE	A published altitude representing the maximum usable altitude or flight level for an airspace structure or route segment.
MILITARY TRAINING ROUTE	Airspace of defined vertical and lateral dimensions established for the conduct of military flight training at airspeeds in excess of 250 knots IAS.
MINIMUM CROSSING ALTITUDE	The lowest altitude at which the aircraft must cross a fix when proceeding in the direction of a higher minimum enroute IFR altitude.
MINIMUM OBSTRUCTION CLEARANCE ALTITUDE	The lowest published altitude in effect between radio fixes on VOR airways, off-airway routes, or route segments that meets obstacle clearance requirements for the entire route segment.
NAVAID	Any visual or electronic device airborne or on the surface which provides point-to-point guidance information or position data to aircraft in flight.
PLAYBOOK ROUTE	A standard route that ATC can utilize to fit a particular set of circumstances, when the preferred routes are not available.
PREFERRED ROUTE	Preferred routes that are not automatically applied by Host.
REPORTING POINT	A geographical location in relation to which the position of an aircraft is reported.
RNAV APPROACH	An instrument approach procedure which relies on aircraft area navigation equipment for navigational guidance.

Once these additional elements were added to the ASDL Ontology, ASDL Metamodel goes through an evolution phase where new entities and relationships are added to the core language structure. As a result of such extension, ATC training scenarios could then be described using graphical interface of ASDL suite. This process has been described for the sample scenario in Section 6.

4 ATC Integration with ASDL

ASDL syntax and entities relationships are described using a metamodel. Since its original version, ASDL metamodel has gone through series of expansion to include new scenario types such as ATC enroute. After the addition of ATC ontology to ASDL, its metamodel was extended to include ATC-specific elements. A snippet of ASDL updated metamodel is illustrated in Figure 8 and Figure 9 where new entities are demonstrated.

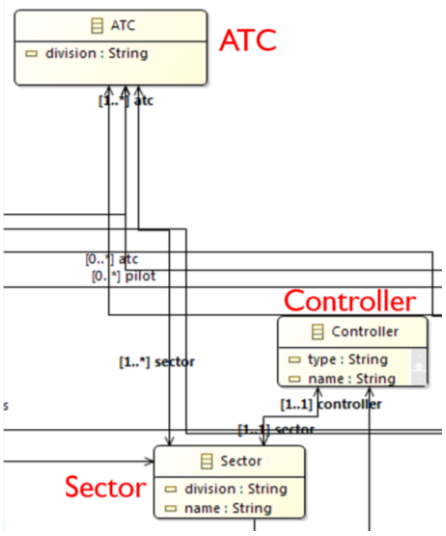

Figure 8. Relationships between ATC, Controller, and Sector Entities in ASDL Metamodel.

The enroute extension begins with the aircraft in the departure climb. Upon reaching the cleared altitude, the aircraft begins to cruise towards its final destination. At any point the aircraft can turn, climb, or descend as specified by ATC. Upon arriving near the destination, the aircraft requests to land, ending the enroute phase. To support the changes needed for capturing the enroute phase of flight, some entities in the metamodel needed to be modified. For example, the original ASDL implementation of the ATC entity did not account for the different controller designations, such as ground controller, tower controller, and radar controller. This was mainly due to the limited scope of the original ASDL implementation, which accounted only for a tower controller. The enroute extension created two new entities: Controller and Sector. As shown in Figure 8, the ATC class now represents a division of ATC, such as Tower, enroute, or Terminal Radar Approach Control (TRACON). Each division of ATC contains any number of sectors that that particular ATC division controls. For each sector, there is one controller working traffic at a given time and a sector division, such as tower in a Tower ATC division. The controller type designates different positions in an ATC division, such as a ground controller versus a tower controller in the Tower ATC division.

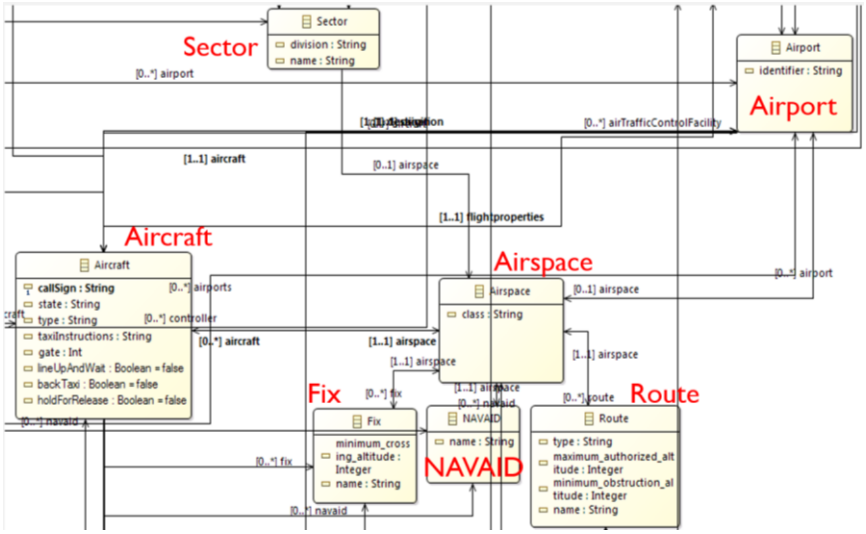

Figure 9. Relationships between Aircraft, Sector, Airport, Airspace, Fix, NAVAID, and Route Entities after Enroute Updates.

In addition to the more detailed breakdown of the ATC structure, airspace was also added to the metamodel in the form of an Airspace class, which is further divided into Fix, NAVAID, and Route classes, as shown in Figure 9. Airspace is defined by class, such as class A airspace or class E airspace. An airspace contains any number of fixes, NAVAIDs, routes, and airports. A sector represents part of an airspace, which allows the controller to see the representation of the airspace they are controlling. Finally, an aircraft flies in an airspace and an airspace can have any number of aircraft in it.

6 ATC Enroute Scenario Logic Verification in ASDL

ATC enroute extension to ASDL was verified for accuracy and completion through the creation of scenarios using EMF's built-in runtime model. Any errors in the model, such as incorrect entity associations, would become apparent when creating a runtime scenario. First, each scenario was written in natural language. Next, a state diagram was created which describes the aircraft's state throughout the written scenario. Finally, the state diagram was translated into the EMF runtime scenario. This reversed engineering process was used to validate that a variety of different simulation scenarios could be captured by the enroute statemachine for automatic verification by ASDL suite. Formal scenario validation and verification in ASDL has been discussed in details in [5]. Moreover, refer to [12] and [11] for more details regarding the EMF runtime scenario implementations.

The enroute scenario describes the aircraft's progression through airspace from the departure climb to the arrival descent. An example enroute scenario is defined as follows:

Aircraft ER-1357 has departed from the Daytona Beach International Airport (KDAB) via runway 7L. ER-1357 will be landing on runway 29 at the Gainesville airport (KGNV). Stable weather conditions nearing Daytona Beach are reported as calm winds, dew point: 23, sky condition: scattered clouds at 5500 feet, temperature: 15, visibility: 10. ER-1357 begins climbing and heads direct to the Ormond VOR (OMN). Daytona departure ATC grants ER-1357 to climb to 8000 feet and tells ER-1357 to proceed along the route as filed. ER-1357 begins to climb to 8000 feet. After passing OMN, ER-1357 joins airway T207-208. At CARRA, Daytona departure hands off ER-1357 to Jacksonville approach. ER-1357 turns west to join T208, heading towards KGNV. Once within 30 miles of KGNV, Jacksonville approach control contacts ER-1357 and the landing phase begins.

The state diagram for the enroute extension is shown in Figure 10. During the enroute phase of flight, the aircraft spends the majority of time in the Cruise state, where the aircraft's location on the route is constantly updating. At any time during the Cruise state, an aircraft may turn, climb, or descend as necessary to avoid traffic, follow the designated route, or as commanded by ATC. After arriving within approach range of the destination, the aircraft transitions into the landing phase, thereby ending the enroute phase.

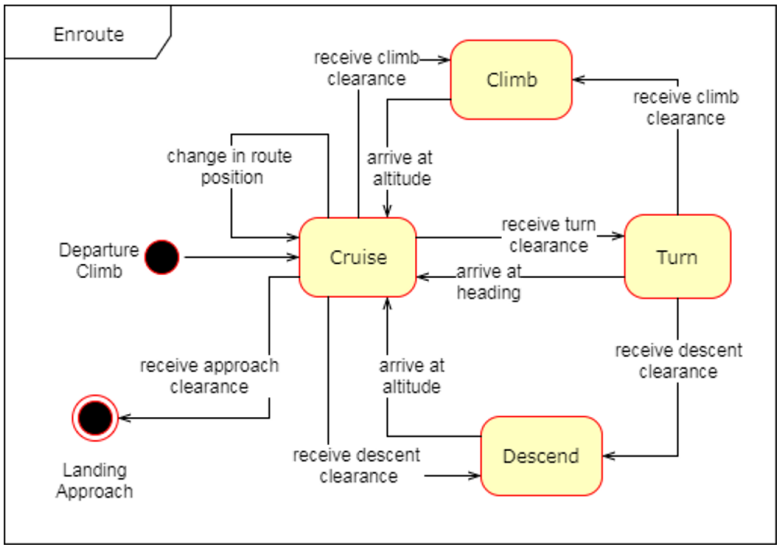

Figure 10. Enroute Scenario State Diagram.

Figure 11 shows the EMF runtime implementation for the enroute example scenario. Figure 12 expands the Pattern of Interplay to show how the scenario follows the events detailed in the written scenario.

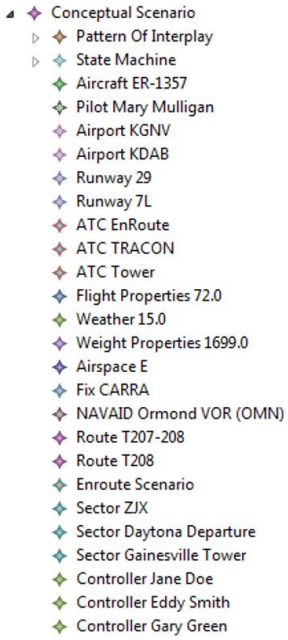

Figure 11. Sample Enroute Scenario Entities Specified in ASDL.

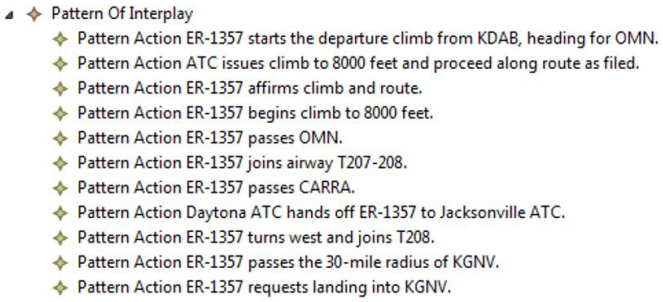

Figure 12. Sample Enroute Scenario Actions Captured in ASDL.

Figure 13 expands the State Machine to show the aircraft's movement through each state. Note how this attribute closely follows the states and transitions seen in the state diagram in Figure 10.

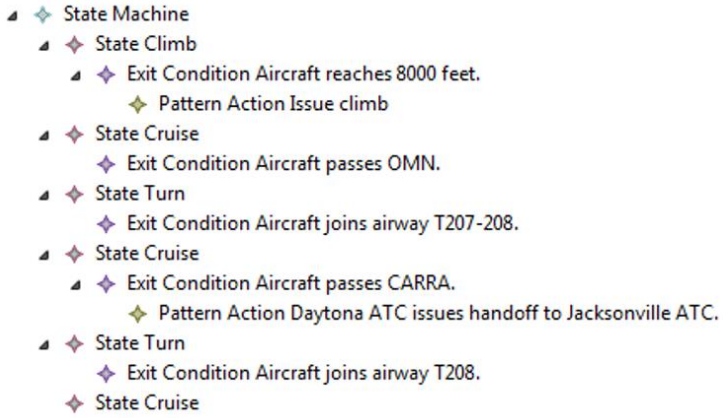

Figure 13. States Specification of the Sample Enroute Scenario.

Figure 14 depicts an excerpt from the enroute scenario XML file. This portion of the scenario shows the definition of all states in the State Machine. The XML script is automatically generated in the ASDL Suite according to ASDL schema that is constructed in EMF as dictated by the ASDL metamodel.

```
<stateMachines>
  <states Name="Climb">
    <guard Name="Aircraft reaches 8000 feet.">
      <exitAction Name="Issue climb" Sequence="1"/>
    </guard>
  </states>
  <states Name="Cruise">
    <guard Name="Aircraft passes OMN."/>
  </states>
  <states Name="Turn">
    <guard Name="Aircraft joins airway T207-208."/>
  </states>
  <states Name="Cruise">
    <guard Name="Aircraft passes CARRA.">
      <exitAction Name="Daytona ATC issues handoff to Jacksonville ATC." Sequence="1"/>
    </guard>
  </states>
  <states Name="Turn">
    <guard Name="Aircraft joins airway T208."/>
  </states>
  <states Name="Cruise"/>
</stateMachines>
```

Figure 14. Automatically Generated Scenario XML Script in ASDL Suite.

7 ATC Enroute Scenario Schema in ASDL

The XML schema (XSD file) for the ATC addition can be generated directly from EMF. Figure 15 shows a snippet of this schema.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
ecore:nsPrefix="ScenarioSchema" ecore:package="ScenarioSchema">
  <xsd:element name="scenario" type="scenario_._type"/>

  <xsd:complexType ecore:name="FlightType" name="flight_._type">
    <xsd:sequence>
      <xsd:element name="acid" type="xsd:string"/>
      <xsd:element minOccurs="0" name="rules" type="xsd:string"/>
      <xsd:element minOccurs="0" name="beacon" type="xsd:int"/>
      <xsd:element name="type" type="xsd:string"/>
      <xsd:element ecore:name="start_time" minOccurs="0" name="start_time" type="xsd:string"/>
      <xsd:element ecore:name="start_point" minOccurs="0" name="start_point" type="xsd:string"/>
      <xsd:element ecore:name="is_departure" minOccurs="0" name="is_departure" type="xsd:boolean"/>
      <xsd:element ecore:name="start_alt" minOccurs="0" name="start_alt" type="xsd:int"/>
      <xsd:element ecore:name="is_interim" minOccurs="0" name="is_interim" type="xsd:boolean"/>
      <xsd:element ecore:name="assigned_speed" minOccurs="0" name="assigned_speed" type="xsd:string"/>
      <xsd:element ecore:name="filed_speed" minOccurs="0" name="filed_speed" type="xsd:string"/>
      <xsd:element minOccurs="0" name="departure" type="xsd:string"/>
      <xsd:element ecore:name="dep_runway" minOccurs="0" name="dep_runway" type="xsd:string"/>
      <xsd:element minOccurs="0" name="destination" type="xsd:string"/>
      <xsd:element ecore:name="dest_runway" minOccurs="0" name="dest_runway" type="xsd:string"/>
      <xsd:element minOccurs="0" name="routed" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="route" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType ecore:name="ScenarioType" name="scenario_._type">
    <xsd:sequence>
      <xsd:element ecore:name="syntax_version" minOccurs="0" name="syntax_version" type="xsd:int"/>
      <xsd:element ecore:name="isim_version" minOccurs="0" name="isim_version" type="xsd:string"/>
      <xsd:element minOccurs="0" name="name" type="xsd:string"/>
      <xsd:element minOccurs="0" name="description" type="xsd:string"/>
      <xsd:element ecore:name="start_time" minOccurs="0" name="start_time" type="xsd:string"/>
      <xsd:element ecore:name="stop_time" minOccurs="0" name="stop_time" type="xsd:string"/>
      <xsd:element ecore:name="external_airspace" minOccurs="0" name="external_airspace" type="xsd:string"/>
      <xsd:element minOccurs="0" name="wind" type="wind_._type"/>
```

Figure 15. Snippet of ASDL XSD Schema.

A sample scenario was defined in the Eclipse Model Editor using the ASDL meta-model. The scenario was created using the information available in plain-text description, which was a collection of weather data, flights and prompts for an en route scenario. Some of the properties included in this scenario can be seen in Figure 16.

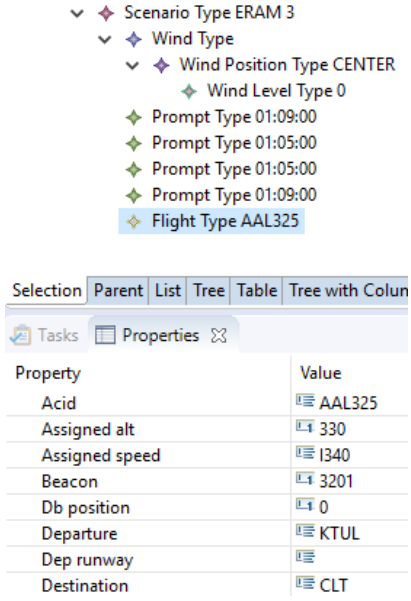

Figure 16. ATC Scenario Definition in Eclipse Model Editor.

Next, an XML script was generated for this scenario, which adheres to the schema defined earlier. The script is obtained using the scenario definition in the Eclipse

Model Editor. Figure 17 shows a snippet of this XML file with one prompt and one flight.

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario>
  <syntax_version>5</syntax_version>
  <isim_version>2.2.0.2316</isim_version>
  <name>ERAM 3</name>
  <description>1 AIRCRAFT, CBM COLD, R931 COLD, R357 COLD, VFR</description>
  <start_time>01:00:00</start_time>
  <stop_time>02:30:00</stop_time>
  <external_airspace>MasterZAE.xm</external_airspace>
  <wind>
    <wind_position>
      <id>CENTER</id>
      <position>400000N/0850000W</position>
      <wind_level>
        <from>360</from>
      </wind_level>
    </wind_position>
  </wind>
  <prompt>
    <time>01:09:00</time>
    <acid>N5334D</acid>
    <active>true</active>
    <action>General Text</action>
    <text>REQUEST VISUAL APPROACH GWO</text>
  </prompt>
  <flight>
    <acid>AAL325</acid>
    <beacon>3201</beacon>
    <type>MD88/A</type>
    <start_time>01:05:00</start_time>
    <start_point>332448N/0912312W</start_point>
    <start_alt>330</start_alt>
    <assigned_speed>I340</assigned_speed>
    <filed_speed>I340</filed_speed>
    <departure>KTUL</departure>
    <destination>CLT</destination>
    <remarks></remarks>
    <routed>true</routed>
    <route>.MEI.</route>
  </flight>
</scenario>
```

Figure 17. Snippet of XML File for ATC Scenario.

For the first step, an XML schema (in XSD format) was generated for the metamodel using Eclipse. The schema defines each entity and lists its attributes along with their data types. Attributes which are of a base type, such as integer (int), string, floating point number (float) or double-precision floating point number (double) are considered ‘simple’ attributes, and those that are objects of other classes with their own set of attributes are called ‘complex’ attributes. In this case, the simple attributes are assigned to each of the entities directly, and the complex ones are referenced to their original definition. The generated schema implements the constraints for formatting and rules for each entity present within the metamodel. It includes restrictions on the quantity and types of valid inputs for each element within the scenario. The schema was created by loading the generic XML package in Eclipse’s XSD Editor, making it Eclipse-independent. The schema was checked using XML tools and was validated as well-formed and error-free.

Next, an XML script was created for this scenario. The Eclipse runtime instance allows properties to be defined for each entity in the metamodel. This is performed using the runtime editor shown in Figure 16. A runtime model can be edited using the GUI but is directly as an XML file, which can be passed directly to a simulator.

8 Conclusion

Air Traffic Control (ATC) training scenarios are generated manually from a subject-matter-expert's (i.e. controllers) oral or written briefing. The effort in extracting and verifying operational scenarios and translating them into a machine-understandable language is rather cumbersome and currently conducted completely manual. To address these challenges, here we extended the recent effort in standardizing aviation scenario specification (Aviation Scenario Definition Language - ASDL) to provide a common mechanism for specifying simulation scenarios in the ATC domain. With the support of graphical user interface, an ATC instructor or controller can easily provide details on various ATC scenarios without the need to understand the simulation technology behind it. ASDL suite utilizes automation to generate scenario XML scripts that can be deployed on target simulators. This chapter presented an overview of ASDL and its newly added features to capture ATC scenarios. The additional terminology, known as ATC ontology, is represented, mainly focusing on enroute ATC. ASDL metamodel was also extended to include ATC addition. The scenarios logic was captured and verified with a statemachine and a sample enroute scenario was presented. Finally, XML schema of the updated ASDL was briefly discussed and the automatic XML generation capability of ASDL suite was presented. By presenting a formal approach in specifying ATC scenarios, we aim at providing a standard mechanism for specification and development of scenarios, supporting reusability and shareability across the aviation simulation community.

References

[1] ASDL, 2017, URL: https://github.com/ASDL-prj/Ontology. [retrieved 1 January 2018].

[2] Bechhofer, Sean. 2009. "OWL: Web Ontology Language." In Encyclopedia of Database Systems, by Ling Liu and M. Tamer Özsu, 2008-2009. Boston: Springer US.

[3] Brambilla, M., Cabot, J., and Wimmer, M., "Model-Driven Software Engineering in Practice", Morgan & Claypool Publishers, 2012.

[4] Čeh, Ines, Matej Črepinšek, Tomaž Kosar, and Marjan Mernik. 2011. "Ontology driven development of domain-specific languages." Computer Science and Information Systems 2 317-342.

[5] Chhaya, B., Jafer, S., and Durak, U. "Formal Verification of Simulation Scenarios in Aviation Scenario Definition Language (ASDL)". Aerospace Journal, *In press*. 2018.

[6] FAA, M., "NextGen Implementation Plan," Washington, DC, 2016, URL: https://www.faa.gov/nextgen/media/NextGen_Implementation_Plan-2016.pdf [retrieved 2 December 2017].

[7] Fowler, M., Domain-Specific Languages, Pearson Education, 2010.

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

[9] Hudak, P., “Domain-specific languages,” Handbook of Programming Languages,; Vol. 3, No. 39-60, 1997, p. 21.
[10] Jafer, S., Chhaya, B., and Durak, U. "OWL Ontology to ECORE Metamodel Transformation for Designing a Domain Specific Language to Develop Aviation Scenarios," in SpringSim-Mod4Sim, Virginia Beach, VA, 2017.
[11] Jafer, S., Chhaya, B., and Durak, U. “Automatic Generation of Flight Simulation Scenarios with Aviation Scenario Definition Language”. Journal of Aerospace Information Systems (AIAA – JAIS). *In press*. 2018.
[12] Jafer, S., Chhaya, B., and Durak, U., “Graphical Specification of Flight Scenarios with Aviation Scenario Defintion Language (ASDL),” AIAA Modeling and Simulation Technologies Conference, 2017, pp. 1311.
[13] Jafer, S., Chhaya, B., Durak, U., and Gerlach, T., “Formal Scenario Definition Language for Aviation: Aircraft Landing Case Study,” AIAA Modeling and Simulation Technologies Conference, 2016. pp. 3521.
[14] OWL. “OWL,” Web Ontology Language Reference, edited by M. Dean, and G. Schreiber, W3C Recommendation, 2004, URL: http://www.w3.org/TR/owlref/ [retrieved 7 February 2017].
[15] Saeki, M., and Kaiya, H., “On Relationships among Models, Metamodels and Ontologies,” Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM 2006), 2006.
[16] Simulation Interoperability Standards Organization - SISO, "Overview of SISO: Who We Are and What We Do," 2017. [Online]. Available: https://www.sisostds.org/AboutSISO/Overview.aspx. [Accessed 28 November 2017].
[17] Steinberg, D., Budinsky, F., Merks, E., and Paternostro, M., EMF: Eclipse Modeling Framework, Pearson Education, 2008.
[18] Yao, Zhong, and Quan Zhang. 2009. "Protégé-Based Ontology Knowledge Representation for MIS Courses” In Web Information Systems and Mining, 2009. WISM 2009. International Conference on, pp. 787-791. IEEE, 2009.