Formal Scenario Definition Language for Aviation: Aircraft Landing Case Study

Shafagh Jafer[1] and Bharvi Chhaya[2]
Embry-Riddle Aeronautical University, Daytona Beach, FL, 32114-3900

Umut Durak[3] and Torsten Gerlach.[4]
German Aerospace Center (DLR), Institute of Flight Systems, Braunschweig, 38108, Germany

Although the importance of scenarios in modeling and simulation has long been well known, there still exists a lack of common understanding and standardized practices in simulation scenario development. This paper proposes a Domain-Specific Language (DLS) to provide a standard scenario specification that will lead to a common mechanism for verifying and executing aviation scenarios, effective sharing of scenarios among various simulation environments, improve the consistency among different simulators and simulations, and even enable the reuse of scenario specifications. Following DSL design practices, the proposed Aviation Scenario Definition Language (ASDL) will provide a well-structured definition language to formally specify complete aircraft landing scenarios. In order to capture the necessary constructs for a simulation scenario, Simulation Interoperability Standards Organization (SISO) Base Object Model (BOM) is adopted as the baseline metamodel. This baseline is extended using the fundamentals of aircraft landing that cover all the domain-related concepts and terminology as constructs. By taking a formal approach in defining aviation scenarios, ASDL aims at providing consistency and completeness checking, and model-to-text transformations capabilities for various targets in the aviation scenario definition domain. The results of this work will be used to develop a graphical modeling environment and automatic means to transform scenario models into executable scenario scripts. The work presented here is the first stepping stone in formal scenario definition in aviation domain.

I. Introduction

MODELING and simulation is a swift, accurate and cost-effective method to investigate potential solutions to problems. In the case of aviation applications, flight simulation refers to the process of simulating an aircraft and its dynamics through the use of computer systems and advanced software. Modern flight simulators are used for initial and advanced training and for proficiency maintenance and evaluation. They have become integral aspects of the research, development, test, and evaluation cycle.[1] The execution of any simulation requires a clearly-defined scenario which is being investigated. This simulation scenario can be defined as the specification of initial and terminal conditions, significant events and the environment as well as the major entities, their capabilities, behavior and interactions over time.[2]

Although the importance of scenarios in modeling and simulation has long been well known, there still exists a lack of common understanding and standardized practices in simulation scenario development. It is an extensive

[1] Assistant Professor, Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, 600 S Clyde Morris Blvd, Daytona Beach FL 32114-3900, jafers@erau.edu .
[2] Master of Software Engineering Graduate, Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, 600 S Clyde Morris Blvd, Daytona Beach FL 32114-3900, chhayab@my.erau.edu .
[3] Research Scientist, Flight Dynamics and Simulation, Lilienthalplatz 7, 38108 Braunschweig, Germany, umut.durak@dlr.de , AIAA Member.
[4] Team Leader, Flight Dynamics and Simulation, Lilienthalplatz 7, 38108 Braunschweig, Germany, torsten.gerlach@dlr.de .

process beginning with the stakeholders' descriptions of the scenario and finishing with the generation of the corresponding executable specification.[3]

This work targets aviation scenario generation by proposing a Domain-Specific Language (DSL) for the aviation domain, called the Aviation Scenario Definition Language (ASDL). ASDL aims at providing a standard scenario specification that will lead to a common mechanism for verifying and executing aviation scenarios, effective sharing of scenarios among various simulation environments, improve the consistency among different simulators and simulations and enable the reuse of scenario specifications. Based on Domain-Specific Language (DSL) design methodologies, ASDL will provide a well-structured definition language to define departure, enroute, re-route, and landing scenarios. This work focuses on aircraft landing scenarios as a case study to demonstrate the feasibility of creation and significance of such a language. For the purposes of this study, a landing scenario refers to the last part of a flight, where the aircraft returns to the ground.

By taking a formal approach in defining aviation scenarios, the aim of this effort is to provide consistency and completeness checking, and model-to-text transformation capabilities for various targets of an aviation scenarios.

II. Background

This section describes the main concepts that have been used in the creation of ASDL which include Domain-Specific Languages (DSL), ontologies as well as the current scenario generation practices.

A. Domain-Specific Languages

A DSL is a custom tailored computer language limited expresiveness for a particular application domain.[4] DSL is created to specifically target problems in a specific domain, highlighting the main features, requirements, constraints, and characteristics of that domain. With DSL, developers construct models that are specific to their application. Such models are composed of elements and relationships that are verified to be valid for that application. One of its greatest benefits is allowing non-developers and people who do not know the domain to understand the overall design. This is normally supported by allowing graphical modeling, where DSL provides a drag and drop capability to construct models. DSL augmented with model-to-text transformation capabilities would then allow for automatic generation of source code from model. A simple example of DSL is the Modelica for dynamics systems modeling and simulation.[5]

B. Ontology

In philosophy, the term ontology has the meaning of a systematic explanation of existence. Gruber defined it for computer science as: "Ontology is an explicit specification of a shared conceptualization".[6]

The merits of ontologies are: a) common vocabulary, b) explication of what has often been left implicit, c) systematization of knowledge, d) standardization and e) meta-model functionality.[7] The applications of ontologies are classified in four main categories: Neutral authoring, ontology as specification, common access to information and ontology-based search.[8] The Web Ontology Language[9] is a language for defining ontologies on the web. An OWL ontology describes a domain in terms of classes, properties and individuals and may include rich descriptions of the characteristics of those objects.

C. Scenario Generation

Generating scenarios for real-time simulators is currently a resource intensive activity. Typically, the simulator development does not concentrate on generation and manipulation of input data, but instead is focused on the execution of the simulation. In the typical process, after a scenario is developed, it is not generally portable to any other simulator other than the one on which it was developed. Where scenario generation capabilities are provided currently, these scenario generation methods are tightly coupled to a specific simulator and cannot be used to generate or manipulate scenarios for other simulators.[10] Researchers tend to use various simulators and associated tools, and create customized data processing programs in order to modify an existing data set into the desired scenario. This leads to the drawback that each scenario generator is currently specific to the application and simulator it is being used for. Also, it appears that each of these efforts utilize their own method for this purpose which they develop from beginning to end. This indicates the necessity for a well-defined language for aviation which would enable the reuse of scenario generation methods among different simulators.

III. Related Work

A basic survey of the literature was conducted on the topics of domain-specific languages, ontologies and scenario generation in the aviation domain. Some of the applicable resources that were discovered are summarized and discussed here.

Sinlapakun and Yachai[11] described the business rules that define aircraft separation and used them as a base to create the Aeronautical Rules Script Language (ASRL). The purpose of ASRL is to provide sufficient elements for determining the required longitudinal aircraft separation at the planning phase, i.e., before the airplanes take-off. A short description of the ASRL design has been given in the paper, along with some details of the scheme used. However, ASRL is only designed for a particular collaborative environment aiming at determining the separation minima required between aircrafts at the planning phase. This makes it a useful resource for understanding the concepts of DSL as they apply to this effort, but it suggests a need for a broader aviation-specific domain language which can be used for purposes beyond this single task.

Gauci and Zammit-Mangion[12] have described the need for a rapid scenario prototyping tool to support the evaluation of a runway conflict alerting system, called the Runway Collision Avoidance Function (RCAF). The tool developed for this purpose has three main components which are a traffic motion generator to generate the flight paths, a scenario definition and control unit to define the runway conflict scenarios using the trajectories created earlier, and a dynamic traffic control unit to control the speed of the target to maintain the intended leeway in the scenario conflict. This tool allows the user to define scenarios related to runway maneuvers. The paper concluded that this tool was shown to provide flexibility and repeatability in scenario execution. The performance of the tool was proved to be appropriate for rapid generation of scenarios of traffic trajectories and conflicts. This paper presents the importance of being able to generate scenarios and how widely these are used in the aviation field.

The work presented by Durak et al.[3] laid down the foundation for the research presented in this paper. It proposed a scenario development process adopting a Model-Driven Engineering (MDE) perspective. The approach can be summarized as to start with a metamodel to enable building a conceptual model, and then to provide transformation of the model from the source. In order to support this idea, the paper described a sample implementation of this method using the Base Object Model (BOM) metamodel. This metamodel introduced the sequence of events and interplay between various simulation elements. The paper recommended that the next step would be to set up these entities and integrate them into the model in order to define flight scenarios. It outlined a proposed development process, which was also explained using a case study. The ASDL aims to follow the process laid out by this paper and use the base metamodel and extend it to constitute a set of entities that are specific to the aviation domain. This metamodel can then be used to accurately model conceptual flight scenarios which can later be transformed into executable scenarios and deployed in simulators.

DSL seems to be a well-defined way of providing a central source of information which can be reused by specific applications for that domain. This recognizes the current need for an aviation-specific domain language. Fablo et al.[13] explained the concept and importance of ontologies in software creation. This led to the construction of a basic aviation ontology being the first step required in the creation of ASDL.

Currently, there are several researchers who recognize the need for an automated scenario generation process and who have been working on it independently. A lot of resources appear to be used for similar tasks which could be saved if there was a process to reuse some aspects of other efforts. This leads to the conclusion that if there was a common standardized language for defining aviation scenarios, a lot of duplicate work and efforts could be avoided. Each related effort (for example: air traffic scenarios, runway collision avoidance scenarios and full-flight simulation scenarios) could use relevant information (airport, runway, flight plan objects) from this language in order to build their specific scenarios which could then be ported to and read by any simulator they choose. This is the main purpose of creating the Aviation Scenario Definition Language which could benefit all aspects of the aviation community for modeling and simulating the various features of flight.

IV. Methodology

This section describes the methodology followed to create the model of the aircraft landing scenario. The ontology creation is described first, followed by an overview of the tool selected to create the model and then a discussion of the actual design and implementation of the model is presented.

A. Ontology

The ontology created for ASDL consists of two parts: keywords that describe the physical model and operation of flights, and words that describe key communication between the control tower and pilots. Once over 100 keywords were identified, the primarily used terms were added to a basic ontology created using Protégé[14], which saves them in OWL format.

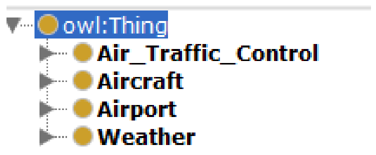


Figure 1. High-level view of ASDL ontology.

The ASDL ontology has four base classes: Air_Traffic_Control, Aircraft, Airport and Weather. This can be seen in Figure 1. The main part of this ontology involves the aircraft and its properties. Figure 2 shows the subclasses of the Aircraft class. The Flight_Properties subclass describes the rules (IFR or VFR) that govern the flight, the speed of the aircraft, the fuel remaining and has three other subclasses: controls (pitch, roll and turn rates), location (altitude, latitude, longitude) and time (arrival time, departure time and ACLT). The physical properties subclass contains the call sign, type of aircraft and its weight class.

B. Eclipse Modeling Framework

EMF is described as a modeling framework and code generation facility for building tools and other applications based on a structured data model[15]. Once a model specification, metamodel, has been defined, EMF provides tools to produce a set of Java classes for the model, along with a set of adapter classes which enable viewing and editing of the model.

The first step was metamodeling, which can be introduced as defining the modeling language itself.[16] The

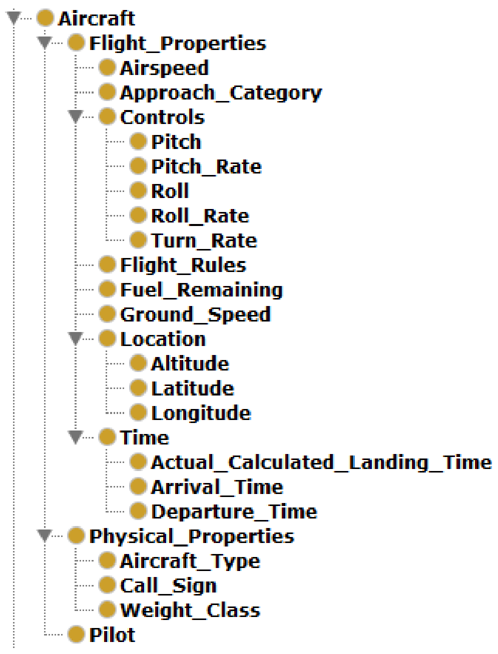


Figure 2. Entities present in Aircraft class in ASDL ontology.

metamodel includes all data items and the relationships between them. This was then defined in EMF in the Ecore, which is the metamodel, that describes the structure of the model. A metamodel is then further utilized to construct a model, which is a concrete instance of this structured data.

The Ecore provides users with the following elements for metamodeling:

- EClass: a class with zero or more attributes and references
- EAttribute: an attribute of the class which has a name and a type
- EReference: an association between two classes
- EDataType: the data type of an attribute

V. Modeling

A. ASDL Metamodel

The overall diagram for the aviation-specific entities of the model can be seen in Figure 3. The Aircraft class is connected to several other classes and contains various attributes that define its properties. An aircraft has one attribute – its call sign, and has various associations with other classes. It is connected to two airports – its port of origin and destination. The runway that the aircraft will be landing on is also specified in the model. At any point, the properties of the flight and weather are also associated with the aircraft class. All aircrafts must have a pilot, and are associated with the ATC tower which is following them at the given time. In addition to these properties, an aircraft also has a state at all times. This has been defined in detail in the next section.

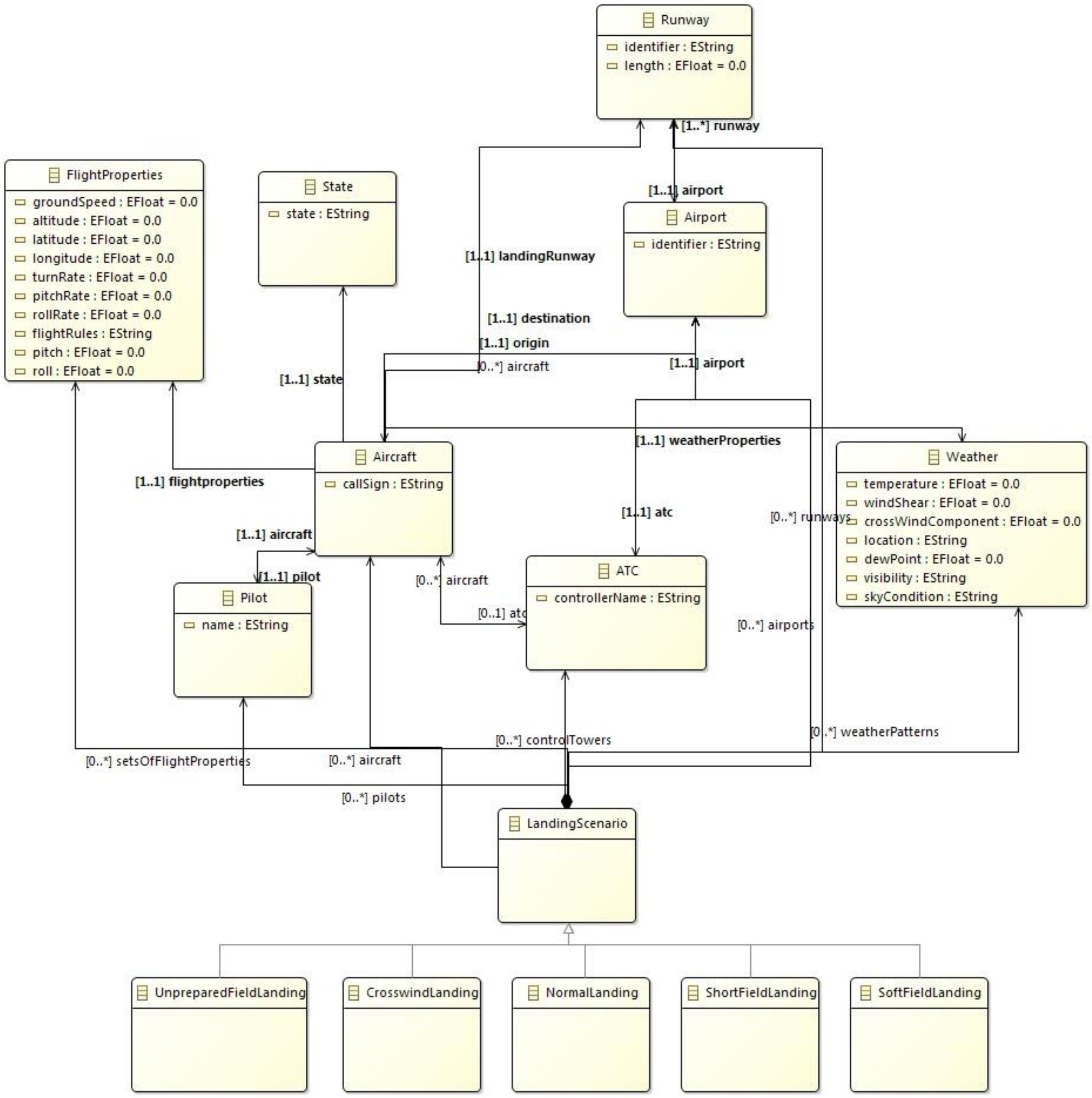


Figure 3. Overall model of aviation entities in ASDL

The Airport class contains one attribute - a string identifier for the airport code. It is also connected to various (zero or more) aircrafts that are departing from and arriving at it. Each airport has an air traffic control tower and one or more runways. A runway has three properties – an identifier that includes its heading, the name of the airport that operates it, and the length of the runway.

The ATC class includes the controller who is acting on behalf of the tower, the airport that it is attached to and the zero to many aircrafts that it is currently monitoring and instructing. The Pilot class consists of the name of the pilot and the aircraft they are associated with. This class would invoke the method to make changes to the state of the aircraft in response to the instructions from ATC. The FlightProperties class shows the instantaneous properties of the flight. It includes the location as well as the motion and speed characteristics. The Weather class shows the instantaneous weather conditions around the aircraft.

B. Landing Model

LandingScenario is the superclass which contains five subclasses: NormalLanding, CrosswindLanding, ShortFieldLanding, SoftFieldLanding and UnpreparedFieldLanding. These are the types of landing scenarios that can be handled by the simulation environment. A type is assigned based on the scenario being executed.

The LandingScenario class also contains all others so that an instance created of this class can define the properties of all other classes and their attributes. It is a collection of pilots, airports, runways, control towers, flight properties, weather patterns and aircrafts which collectively define all the required scenarios.

C. Integration with BOM Metamodel

These classes were integrated with the BOM framework developed by Durak et al.[3] The details of some of the important classes in this model have been explained here.

The ConceptualScenario class is the base class that contains all the other classes in the model. An instance created of this class can define the properties of all other classes and their attributes. In the integrated model that includes all BOM and Aviation Domain classes, LandingScenario is also contained within the ConceptualScenario class. This also includes all other properties, such as other conceptual entities, patterns of interplay among various entities, a state machine that defines the states of an entity, events which cause state changes as well as a class listing all the basic attributes of the scenario (ScenarioIdentification class).

The ScenarioIdentification class lists a number of attributes that explain the scenario in terms of its name, version, description, purpose along with a point of contact and their details. This class is meant to be for informational purposes.

The ConceptualEntity class has a name and characteristics of the entity. It defines all the domain entities that interact with each other in order to perform actions within the scenario. All the aviation scenario classes defined in sub-section A are conceptual entities.

The PatternOfInterplay class provides a mechanism for identifying sequences of pattern actions (including variations and exceptions) which are necessary for fulfilling a pattern of interplay, which is being represented by the BOM. A PatternAction is a single step in a pattern of interplay that may result in a state change of a conceptual entity. It has a name, a position in the sequence of the pattern of interplay, a conceptual entity as a sender and another as a receiver, an event that represents the pattern action as well as related variations and exceptions. This class identifies a pattern action to be carried out in order to successfully accomplish the named pattern of interplay that it is a part of.

The StateMachine class provides a mechanism for identifying the different behavior states that are expected to be exhibited by the conceptual entities. It identifies the state machines including the conceptual entity in question and the states required for representing the aspects of the conceptual model.

VI. Behavior Model

The State class in the ASDL metamodel has only a single attribute and is connected to the aircraft. The purpose of this class is to describe the behavior of the aircraft at any given time. A state diagram of the aircraft during the landing process has been shown in Figure 6. This figure shows the pattern actions by ATC and the pilot, and the states exhibited by the aircraft in response to these actions from cruise until completion of the landing process. In this case, state change occurs when ATC issues or denies the respective clearance requested by the pilot. The clearance status is changed by ATC and in response to this clearance, the state of the aircraft is changed by the pilot to reflect the next action required. The specific state changes for each type of landing have been shown in the figures in Section VII.

VII. Sample Landing Scenarios

Two sample landing scenarios have been presented in this paper in order to highlight the use of ASDL in defining these scenarios. These are a normal landing scenario with a standard pattern of interplay, and a crosswind landing scenario.

A. Normal Landing Scenario

This section describes the process of building a normal landing scenario. In this case, before the process and pattern of interplay of a normal landing could be modeled, the main entities needed to be defined. These were done as shown in Figure 4. The Pattern of Interplay of events for a normal landing was then added to the scenario. The behavior of the aircraft can be modeled using the diagram shown earlier (Figure 6). The exit conditions for each state are listed in terms of the Pattern Actions and the state following it. This can be seen in Figure 5. In this way, the normal landing scenario can be conceptualized using ASDL and the BOM framework to define the related attributes. Scenarios can also be created for other types of landing as explained in the following sub-section.

B. Crosswind Landing Scenario

A crosswind is any wind that has a perpendicular component to the line or direction of travel. A crosswind landing has to be performed when a substantial component of the wind is perpendicular to the runway center line. Wind gusts, downdrafts, and wind shear are often part of a crosswind landing. These factors require a pilot to adjust his approach path, speed, configuration, and technique. For gusty conditions or wind shear, the pilot needs to increase the approach

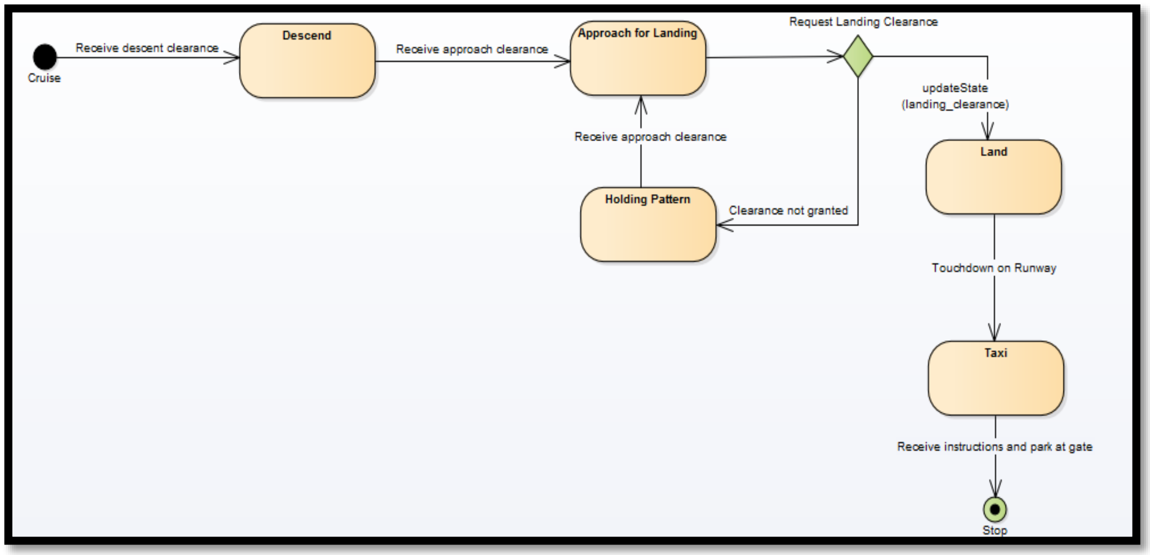
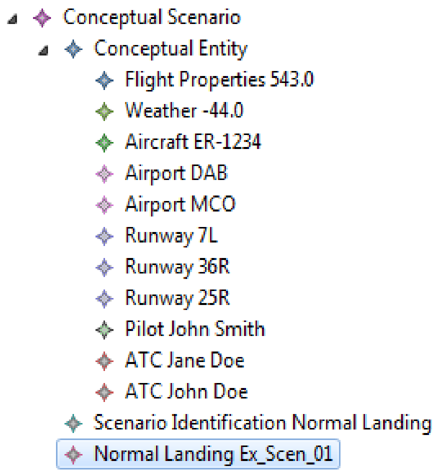
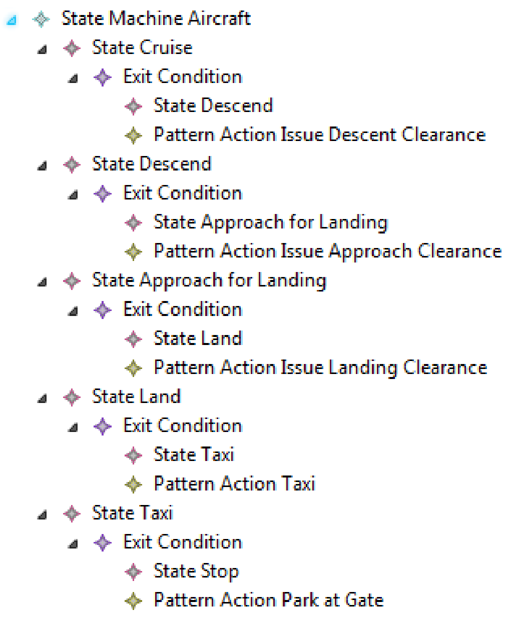

Cruise
Receive descent clearance
Descend
Receive approach clearance
Approach for Landing
Request Landing Clearance
updateState
(landing_clearance)
Receive approach clearance
Holding Pattern
Clearance not granted
Land
Touchdown on Runway
Taxi
Receive instructions and park at gate
Stop

Figure 6. State diagram for aircraft landing model.



- Conceptual Scenario
  - Conceptual Entity
    - Flight Properties 543.0
    - Weather -44.0
    - Aircraft ER-1234
    - Airport DAB
    - Airport MCO
    - Runway 7L
    - Runway 36R
    - Runway 25R
    - Pilot John Smith
    - ATC Jane Doe
    - ATC John Doe
  - Scenario Identification Normal Landing
  - Normal Landing Ex_Scen_01

Figure 4. Conceptual scenario attributes for normal landing.



- State Machine Aircraft
  - State Cruise
    - Exit Condition
      - State Descend
      - Pattern Action Issue Descent Clearance
  - State Descend
    - Exit Condition
      - State Approach for Landing
      - Pattern Action Issue Approach Clearance
  - State Approach for Landing
    - Exit Condition
      - State Land
      - Pattern Action Issue Landing Clearance
  - State Land
    - Exit Condition
      - State Taxi
      - Pattern Action Taxi
  - State Taxi
    - Exit Condition
      - State Stop
      - Pattern Action Park at Gate

Figure 5. State of the aircraft for normal landing.

speed by one half the gust factor, or one half the reported airspeed loss due to wind shear. In this case, the only differences in the scenario generation from the one for normal landing are the weather properties and pattern of interplay for adjusting to the wind conditions.

VIII. Conclusion and Future Work

Current scenario development practices in flight simulators suggest a need for a common scenario specification language and process to improve consistency and enable reuse of scenarios. This paper documents the development of such an Aviation Scenario Definition Language using the BOM framework to create a scenario generation metamodel for aviation applications. This research focused on landing as a case study to demonstrate feasibility and use. As the first step, Protégé was used to create an Ontology of aviation scenario vocabulary. This list of keywords was then used to create a metamodel of a flight's landing scenario in EMF. In order to capture the necessary constructs for a simulation scenario, SISO BOM was adopted as the baseline metamodel. The metamodel for ASDL generated using the BOM framework and aviation keywords was then executed and was able to model five different types of landing scenarios in this case study. The metamodel developed is thus considered to be effective at the initial stage and can be further expanded to model other aspects of flight. This can in turn be used to generate executable scenarios using model transformations.

The initial framework presented in this paper is highly extensible. Some improvements of the metamodel could be proposed as the addition of more atmospheric conditions and other elements encountered by aircrafts. The immediate next step would be to extend the metamodel to account for takeoff, taxi and cruise states so that the duration of an entire flight can be modeled. This would expand the number of scenarios that can be generated using the metamodel. The results of this work can be used to develop a graphical modeling environment and to automatically transform scenario models into executable scenario scripts. The goal in this case would be the recognition of the ASDL as a standardized language through SISO, and its ultimate deployment in simulators. The work presented here is the first stepping stone in formal scenario definition in aviation domain.

References

[1]Moroney, W. F., and Lilienthal, M. G., "Human Factors in Simulation and Training: An Overview," *Human Factors in Simulation and Training,* edited by D. A. Vincenzi, J. A. Wise, M. Mouloua, and P. A. Hancock, CRC Press, Boca Raton, 2008, pp. 3-38.

[2]US Department of Defense, *High Level Architecture Glossary,* US DoD, Washington, DC, 1996.

[3]Durak, U., Topcu, O., Siegfried, R., and Oguztuzun, H., "Scenario Development: A Model-Driven Engineering Perspective," *Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH), 2014 International Conference on*, 2014.

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

[5]Fritzson, P., and Engelson, V. "Modelica – A Unified Object-Oriented Language for System Modeling and Simulation," *ECOOP'98—Object-Oriented Programming*, Springer Berlin Heidelberg, 1998, pp. 67-90.

[6]Gruber, T., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Int. Journal of Human-Computer Studies*, Vol. 43, 1995, pp. 907-928.

[7]Mizoguchi, R., "Ontological Engineering: Foundations of the Next Generation Knowledge Processing," *Web Intelligence 2001*, Maebashi City, Japan, 2001.

[8]Falbo, R. A., Guizzardi, G., and Duarte, K. C., "An Ontological Approach to Domain Engineering," *International Conference on Software Engineering and Knowledge Engineering*, Ischia, Italy, 2002.

[9]Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., and Stein, L.A, OWL Web Ontology Language Reference. Edited by Dean, M., Schreiber, G. W3C Recommendation, 2004. http://www.w3.org/TR/owlref/. As of 24 February 2016.

[10]Signor, D., Davis, P., Lozito, S., Andre, A., Sweet, D., and Wallace, E., "Efficient Air Traffic Scenario Generation," *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, Chicago, 2004.

[11]Sinlapakun, S., and Limpiyakorn, Y., "Domain Specific Language for Collaborative Determination of Separation Minima between Aircrafts," *International Journal of Software Engineering and its Applications,*2013, pp. 399-414.

[12]Gauci, J., and Zammit-Mangion, D., "A Rapid Scenario Generation Tool for Repeatable Simulated Traffic Conflicts in Flight Simulation," *The 26th International Congress of the Aeronautical Sciences*, 2008.

[13]Fablo, R., Guizzardi, G., and Duarte, K., "An Ontological Approach to Domain Engineering," *Proceedings of the 14th International Conference on Software Engineering*, ACM, Ischia, 2002.

[14]Knublauch, H., Fergerson R. W., Noy, N. F., and Musen, M. A., "The Protégé OWL plugin: An Open Development Environment for Semantic Web Applications," *The Semantic Web–ISWC 2004*, Springer Berlin Heidelberg, 2004, pp. 229-243.

[15] Gronback, R. C., *Eclipse Modeling Project: A Domain-Specific Language Toolkit,* Addison-Wesley, Upper Saddle River, NJ, 2009.

[16]Brambilla, M., Cabot, J., and Wimmer, M., *Model-driven software engineering in practice,* Morgen &Claypool Publishers, 2012.