Domain Specific Language for Collaborative Determination of Separation Minima between Aircrafts

Sakon Sinlapakun and Yachai Limpiyakorn

Department of Computer Engineering, Chulalongkorn University,
Bangkok 10330, Thailand
Sakon.Si@student.chula.ac.th, Yachai.L@chula.ac.th

Abstract

Phenomenal growth in air traffic across the world is leading industries to maximize the use of airspace. One of the promising approaches would be to optimize the utilization of information derived from existing data. In early period, all air traffic rules can be programmed and integrated to the system using a general-purpose language like C#, C++, or Java. To increase sustainability of the system, general-purpose languages are too complicated for the users to understand and maintain. In addition, general purpose languages have to be recompiled every time the code has been changed. This research has developed a domain specific language called Aeronautical Rules Script Language (ARSL). The language is particularly designed for collaborative environment aiming at determining separation minima required between aircrafts at planning phase. ARSL can be used as a formal language for configuring air traffic rules and information sharing. As a domain specific language, ARSL is easy to understand and maintain for aeronautical domain experts. The language has been implemented and integrated into the Collaborative Decision Making (CDM) project of Aeronautical Radio of Thailand to help integrate the major elements essential for defining safety longitudinal aircraft separation. The evaluation results show that the implemented ARSL and its services function accurately, providing the same answers as obtained from the specialist.

Keywords: *Domain Specific Language, Aeronautical, Collaborative Decision Support*

1. Introduction

Air traffic congestion is massively increasing, and it may raise the chance of aviation incidents. For the reasons of safety, pollution reduction, and optimization of air space usage in Thailand, Aeronautical Radio of Thailand (AEROTHAI) has commenced the Collaborative Decision Making (CDM) project. One of the project's goals is "Able to determine separation minima required between aircrafts". Since the performance of each aircraft and its equipment are varied among aircrafts, the air traffic controller must know the characteristics of controlled aircrafts in order to determine the separation required for each aircraft. The fact that lots of data need be fed to the flight data processor for determining aircraft separation minima, several aeronautical data models are developed, namely Aeronautical Information Exchange Model (AIXM) [1], and Flight Information Exchange Model (FIXM) [2]. AIXM is particularly designed for configuring the management and distribution of Aeronautical Information Service (AIS) data in digital format. The current structure level of AIXM is 5.1. FIXM is particularly designed as a data model for flight data exchange. The structures of both AIXM and FIXM are defined by FAA (Federal Aviation Administration) and

EUROCONTROL. Currently, AIXM and FIXM do not support the determination of complex separation rules used in BANGKOK FIR (Flight Information Region).

A domain specific language, DSL, is a specification language dedicated to a particular problem domain. DSLs are designed to allow domain specialists to easily understand, reuse, maintain, and write the code [3]. These advantages make DSLs very popular. The language also contributes to software safety and reliability [4]. Therefore, this research has developed a domain specific language called Aeronautical Rules Script Language (ARSL) to integrate the major elements, such as aircraft information and waypoint (coordinates that define a point in airspace) information, essential for defining required separation between aircrafts.

The remainder of this paper is organized as follows: Section 2 introduces the readers with some useful background knowledge of longitudinal aircraft separation and domain specific language that are related to this research work. Section 3 describes the details of design and implementation of ARSL. The evaluation of ARSL is described in Section 4. Finally, Section 5 summarizes the work presented in this paper and future direction of this research.

2. Background

2.1. Longitudinal aircraft separation

The longitudinal aircraft separation is horizontal spacing between aircrafts which can be calculated by estimating the position of each aircraft. For safety, the longitudinal separation must never be less than longitudinal separation minima. Several techniques are used by air traffic controllers to maintain separation of aircrafts during tactical phase (flight already departed), *e.g.*, Mach number technique, Distance measuring technique, etc. All techniques can be categorized into 2 types: 1) time based, or 2) distance based. Figure 1 illustrates the time based technique to maintain safety longitudinal aircraft separation.

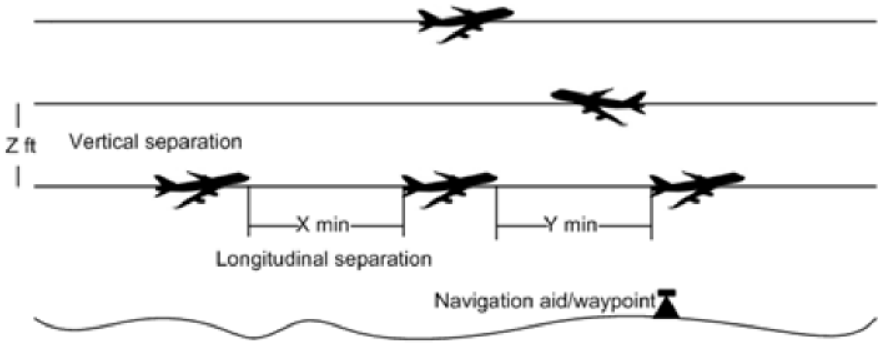

Figure 1. Longitudinal separation minima based on time

Air traffic controllers must ensure that the separation between adjacent aircrafts must not violate the terms of longitudinal separation minima. When the controlled aircraft's separation reaches the minimum, air traffic controllers must maintain the aircraft's separation by limiting its speed not to violate the terms. Unlike a vertical separation, a longitudinal separation can be affected by many factors, namely aircraft performance, destination and navigation aid. To optimize the use of airspace, pragmatically the aircraft separation can be controlled in the planning phase by delaying the take off time of one of the conflicted aircraft. Therefore, it is required to have the information of the actual overfly time over specific waypoints and the required separation prior to flight departures.

Aeronautical Information Package (AIP) [8] declares the longitudinal separation minima between aircrafts. The document contains the aeronautical rules that domain experts can read and understand properly, for example, "10 mins longitudinal separation between RNAV-equipped aircraft apply Mach Number Technique; 15 mins longitudinal separation between other aircrafts". At some waypoints, there are also some special rules contained in the AIP supplement, such as A16/11 stating "2.4 flight intending to land within YGN FIR, CPDLC will not require to achieve RNP 10 navigation requirement".

2.2. Domain Specific Language (DSL)

A domain specific language is a programming language tailored to a specific problem, rather than being of a general purpose programming language. It contains the syntax and semantics that model concepts at the same level of abstraction provided by the problem domain. In general, DSLs are used to help communication among developers and others who are concerned in a specific environment. The benefit of DSLs in software engineering could be "Time and time again, the key bottleneck of software development is communication between developers and those for whom they are developing system" [5].

DSLs are typically developed by experts in a particular domain and focuses only on must-have ability. What makes the DSL very popular may be its easy-to-understand and easy-to-maintain. Moreover, domain experts can develop and maintain the DSL scripts with a little programming skill. Compared to general-purpose languages like C#, C++, or Java, the advantages of domain specific languages can be summarized as follows [5, 6, 7]:

1) Improve development productivity and reduce design time.
2) Improve software quality, productivity, reliability, maintainability, portability and reusability.
3) Communication with domain experts to express solutions in the level of abstraction of the problem domain, so that domain experts can understand, design and validate software by themselves.
4) Change can be made easily during execution context, because some parts of software can be programmed by domain experts.

Domain specific languages can be divided into three main categories [5]:

1) An external DSL is a language separate from the main language and has its custom syntax. Examples of external DSLs are regular expression, SQL and XML configuration.

2) An internal DSL is an extended of a general-purpose language as a subset of that language features. Examples of internal DSLs are Ruby and Lisp.

3) A Language workbench is an IDE for defining and building DSLs.

It is observed that every application has business rules that need to be represented in a readable and declarative form. The applications implemented with DSL scripts enable domain experts to easily understand the domain models and business rules. Developers are merely responsible for implementing the DSLs under the working environments. DSL is generally designed by collecting common vocabularies between problem and solution domains. Developers have to define the DSL elements from the collected vocabulary and generate the semantic model along with the DSL syntax. The correctness of DSL also need be verified. Ideally, the domain experts themselves have to write the DSL scripts to test the correctness of the designed DSL.

3. Aeronautical Rules Script Language (ARSL)

3.1. Requirements

For the current version of ARSL, the purpose of ARSL is to provide sufficient information elements for determining the required longitudinal aircraft separation at the planning phase before airplanes take off. ARSL and its implemented services are developed based on the following requirements:

1) ARSL is developed as the external DSL. As the result, the users are not expected to have any programming skills in order to learn and write ARSL scripts.
2) ARSL parser can detect syntax errors.
3) The Integrated Development Environment (IDE) shall be developed to support editing and maintaining ARSL.
4) ARSL shall be capable for determining the longitudinal separation minima at waypoints (coordinates that define a point in airspace) only, and
5) shall be able to calculate Estimated Time Over (ETO) across waypoints, and
6) shall be capable of accessing the data (routes, departure time, arrival time, and estimated time to cross boundary) from the flight plan in order to calculate ETO, and
7) shall be able to create a flight trajectory (A path that the aircraft will follow) by matching extracted waypoints from the route and their calculated ETO, and
8) shall be able to provide the required separation for each aircraft after flight trajectories have been created, and
9) shall be capable of accessing the required separation contained in Aeronautical Information Package (AIP) [8] of target FIR (Flight Information Region) and its supplement, and
10) shall be able to notify and alert the authority for early problem resolution prior to flight departures.

3.2. ARSL design

ARSL is created for describing aeronautical rules to serve collaborative purpose among stakeholders and to determine the required separation between aircrafts on specific waypoints. The design of ARSL starts by collecting rules and problem vocabulary from International Civil Aviation Organization doc. 4444 and flight rules within Bangkok Flight Information Region. The collected elements are then reviewed by the domain specialists to identify missing data/ information as well as to remove the unused data/ information. Next, the Parser is generated using the validated problem vocabulary and solution vocabulary.

ARSL scheme (Table 1) can be divided into 2 parts: 1) rule declaration, and 2) rule adoption. There are fourteen major elements consisting of four elements required to declare and apply rules, and the rest of ten optional elements used only when needed.

The structure of rule declaration part can be modeled with the class diagram as shown in Figure 2. Each aeronautical rule contains ten major flight information elements:

- HasEquipment− the list of airplane's equipment as one of the elements used for matching the rule and returning the required separation

- NoEquipment− the list of equipment that the target aircraft does not have
- Destination− the waypoint at the end of route
- Departure− the waypoint at which the aircraft takes off
- NAV (Navigation Equipment), COM (Communications Capabilities), SUR (Surveillance Capabilities)− the list of special equipment contained in information section (item 18) of flight plan.
- FlightLevel− the range of flight levels applied to the written rules.
- PBN (Required Navigation Performance)− another special equipment
- Separation− resulting separation required when flight information from flight plan (contained in the previously described nine elements) is matched with a rule. The comparison of the flight information for the matching rules will be performed in sequential. The default rule will be defined as the last rule for matching in case the flight information does not match any rule. Examples of rule declaration of ARSL are illustrated in Figure 3.

Table 1. ARSL Scheme

Elements	***Type***	***Example***
<Rules>	Required	*Rules : rulename;*
<HasEquipment>	Optional	*HasEquipemnt: S, R;*
<NoEquipemnt>	Optional	*NoEquipment: V;*
<Destination>	Optional	*Destination: VTBS;*
<Departure>	Optional	*Departure:VTBS,VTBD;*
<NAV>	Optional	*NAV:RNP10;*
<COM>	Optional	*COM:CPDLC*
<FlightLevel>	Optional	*FlightLevel:>270*
<SUR>	Optional	*SUR:ADSC*
<PBN>	Optional	*PBN:A1;*
<Separation>	Required	*Separation : 10 mins, 80 nm;*
<TargetFix>	Required	*targetFix :TAMOS;*
<TargetTime>	Optional	*targetTime : between 1200 and 1245;*
<TargetRules>	Required	*TargetRules : Basic15Minute;*

The second part of ARSL is rule adoption. The TargetRules contains a list of rules created in rule declaration that will be applied for specific waypoints at certain duration of time. The structure of rule adoption can be visualized using the class diagram as shown in Figure 4.

The part of rule adoption has two major properties: 1) TargetTime, and 2) TargetFixes. TargetTime contains a range of applied start time and applied end time, for example, the time period that the adopted rule has been declared. TargetFixes is the list of waypoints for which the adopted rules are applied.

Examples of rules adoption of ARSL are shown in Figure 5. The former example can be interpreted as “if a flight flies over PASAT, ARSL will adopt the rule of Basic10Minutes which will return 10 minutes time-based separation and 80 nautical miles distance-based separation”. The latter example can be translated to “if a flight flies over BETNO, LIMLA or TANEK, ARSL will check with TESTRULE and TESTRULE1, respectively”.

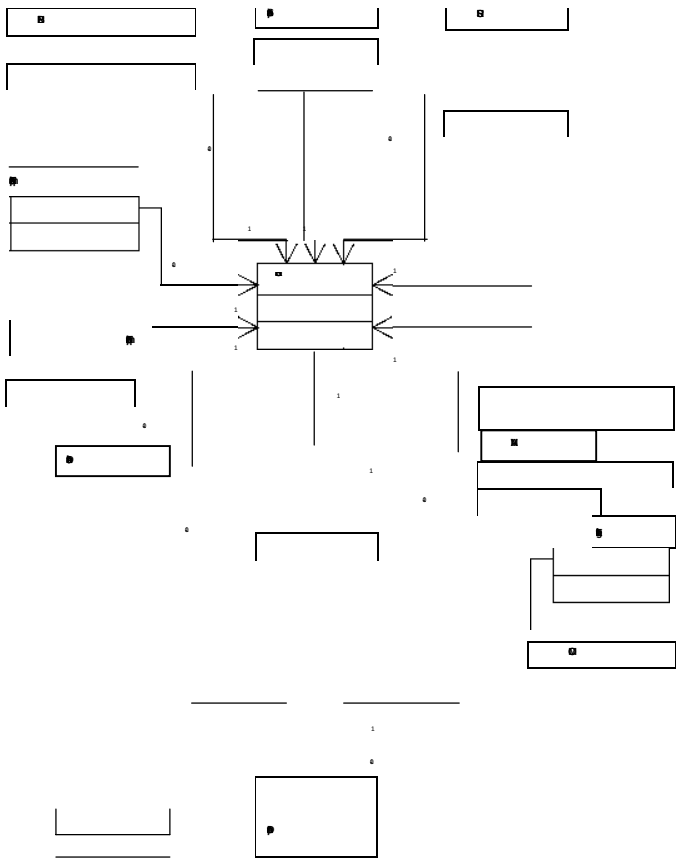

Figure 2. Class diagram of rule declaration of ARSL

```
Rules : TESTRULE;
HasEquipment : J;
NoEquipment : Z;
Departure : VTBS;
Destination : LFPG;
PBN:A1;
NAV:RNAV10;
FlightLevel: >270, <290;
Separation : 9 mins, 60 nm;

Rules : TESTRULE2;
HasEquipment : S, R, J1, J2, J3, J4, J5, J6, J7;
PBN:A1;
FlightLevel: >270;
Separation : 7 mins, 50 nm;
```

Figure 3. Examples of rule declaration of ARSL

Figure 4. Class diagram of rule adoption of ARSL

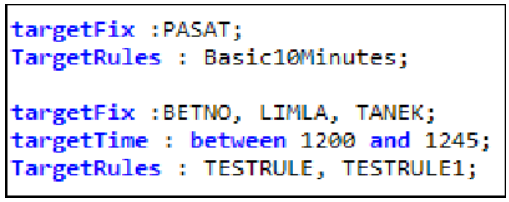

Figure 5. Examples of rule adoption of ARSL

3.3. ARSL parser

Parsing a domain specific language is a strongly hierarchical operation. When parsing text, the chunks are arranged into a tree structure. To build the tree structure, common vocabularies will be collected from problem domain artifacts and solution domain artifacts as shown in Figure 6. The common vocabularies will constitute the lexicons defined in DSL syntax.

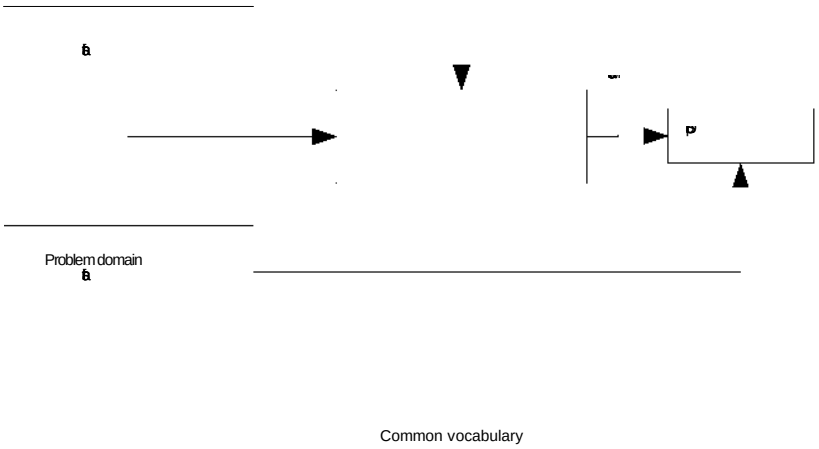

Figure 6. Method of identifying lexicons in ARSL Syntax

Solution domain

In this research, Irony [9] is the development kit used for implementing ARSL on .NET platform. Irony's grammar and parser are coded directly in .net platform C# using operator overloading, rose express grammar constructs in order to generate grammatical rules and data. The ARSL parser reads a script file, builds the parsed tree, and translates it to ARSL parse tree.

3.4. ARSL editor and verification

Editing ARSL scripts has to be performed over the network because only one valid and verified ARSL script should be returned to the requestor. In other words, ARSL scripts will be implanted inside the required separation response service. The ARSL script provider service is implemented for the edit and save interface and it is connected to the IDE developed in this work. The implementation of the ARSL editor is modeled with the class diagram as shown in Figure 7.

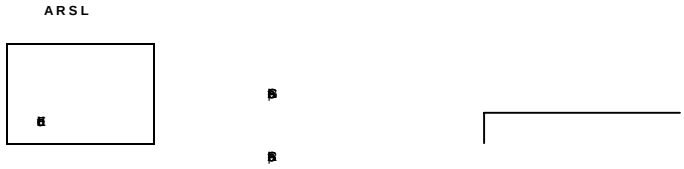

Figure 7. Class diagram of ARSL editor

Both interfaces are created using Windows Communication Foundation (WCF), which supports SOAP protocol to allow other development environments for connecting and accessing ARSL scripts.

In addition, the Grammar checking is implemented within the ARSL Workbench. The

Workbench is responsible for verifying grammar errors, syntax errors, and rule conflicts. It also provides the ability to connect and access ARSL scripts from the ARSL script provider

service, and save the edited scripts back to the ARSL script provider service.

The implementation of ARSL Workbench is shown in Figure 8. The ARSL Workbench can verify rule conflicts including:

1) duplicate applied waypoint, and
2) duplicate rule name, and
3) unassigned rule, and
4) rules not found.

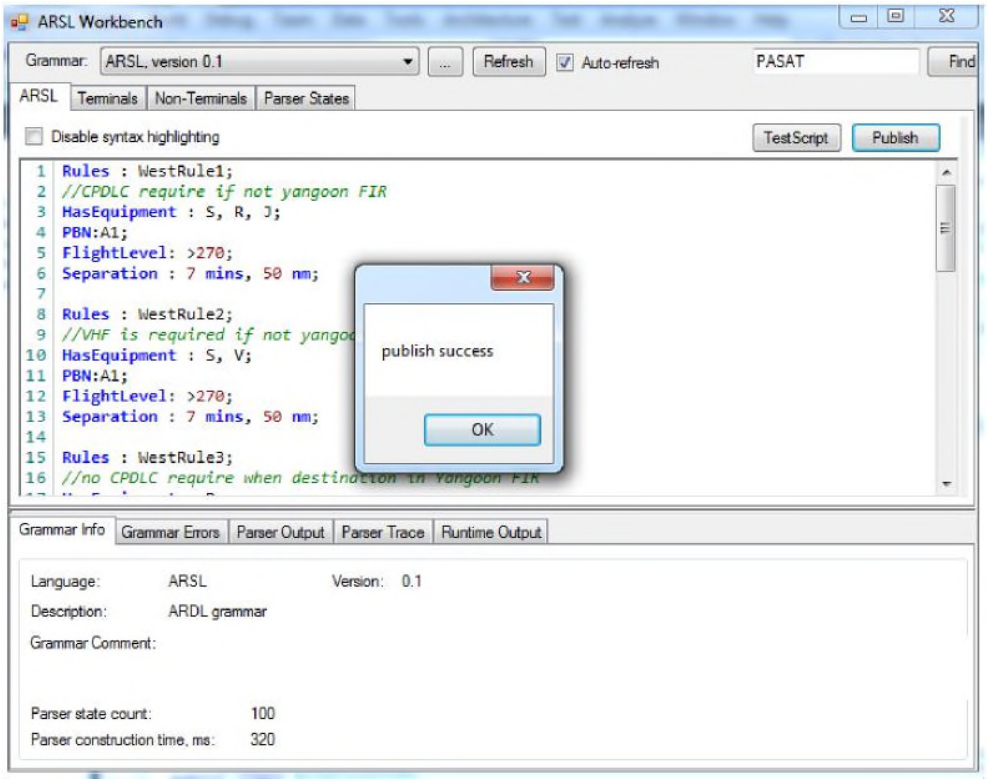

Figure 8. ARSL workbench (ARSL IDE)

3.5. Using ARSL

ARSL is implemented within the Collaborative Decision Making (CDM) project environment. The current scope of CDM is to manage flights in Thailand FIR. In order to manage a flight plan in CDM, we need to know a longitudinal required separation of each aircrafts. Currently, the required separation determined in AIP is simply a constant value, e.g. 10 minutes for RNP10 flights. Unfortunately, in some cases there are special exceptions other than RNP10 flights, for example, flights that fly over LIMLA and BETNO can get 7 minutes required separation without CPDLC equipped.

The purpose of ARSL implementation is to facilitate the air traffic controllers for the aircraft required separation determination. Initially, the ARSL template is established and it contains all of the generic rules listed from Procedure of Air Navigation Services Air Traffic Management [10] that are sufficient to define longitudinal required separations. A domain specialist can start adding or editing rules contained in the

template to produce an ARSL script through the ARSL workbench. Next, he or she has to submit the completed ARSL file to the server by clicking the publish button. If the submitted rules have any errors, the ARSL workbench will not allow publishing to the ARSL script provider service. Once the ARSL script has been validated, rules and data are then generated in order to serve the requesting system that needs the required separation for aircrafts and the rules used to determine the separation. The requesting system can connect to the Rules&Data Generation service via SOAP protocol. The system overview is depicted in Figure 9.

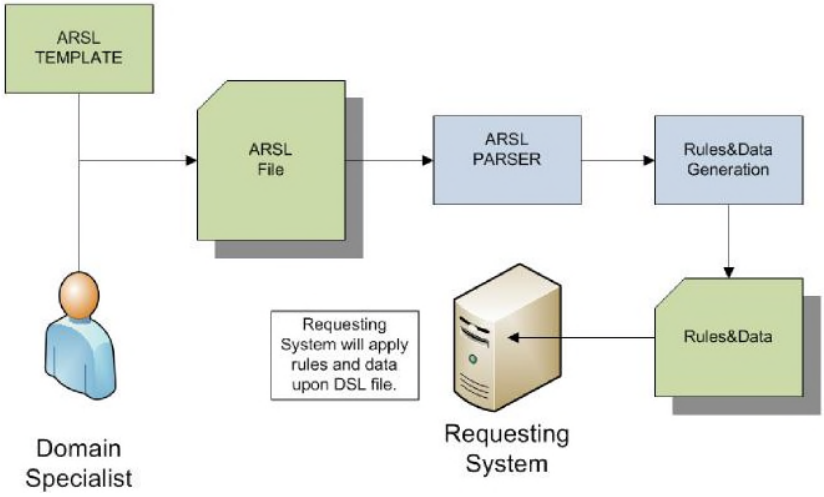

Figure 9. Overview of the implemented system

Request

Name	Value	Type
WaypointName	BETNO	System.String
TimeOverFly	1/28/2013 11:44 AM	System.DateTime
FlightPlan	(FPL-)	System.String
flightLevel	260	System.Int32

Figure 10. ARSL required separation response interface

The requesting system must send some information to the Rules&Data Generation service. The Rules&Data Generation requires four input elements as the property of its web service including waypoint name, time over fly, flight plan, and flight level as shown in Figure 10.

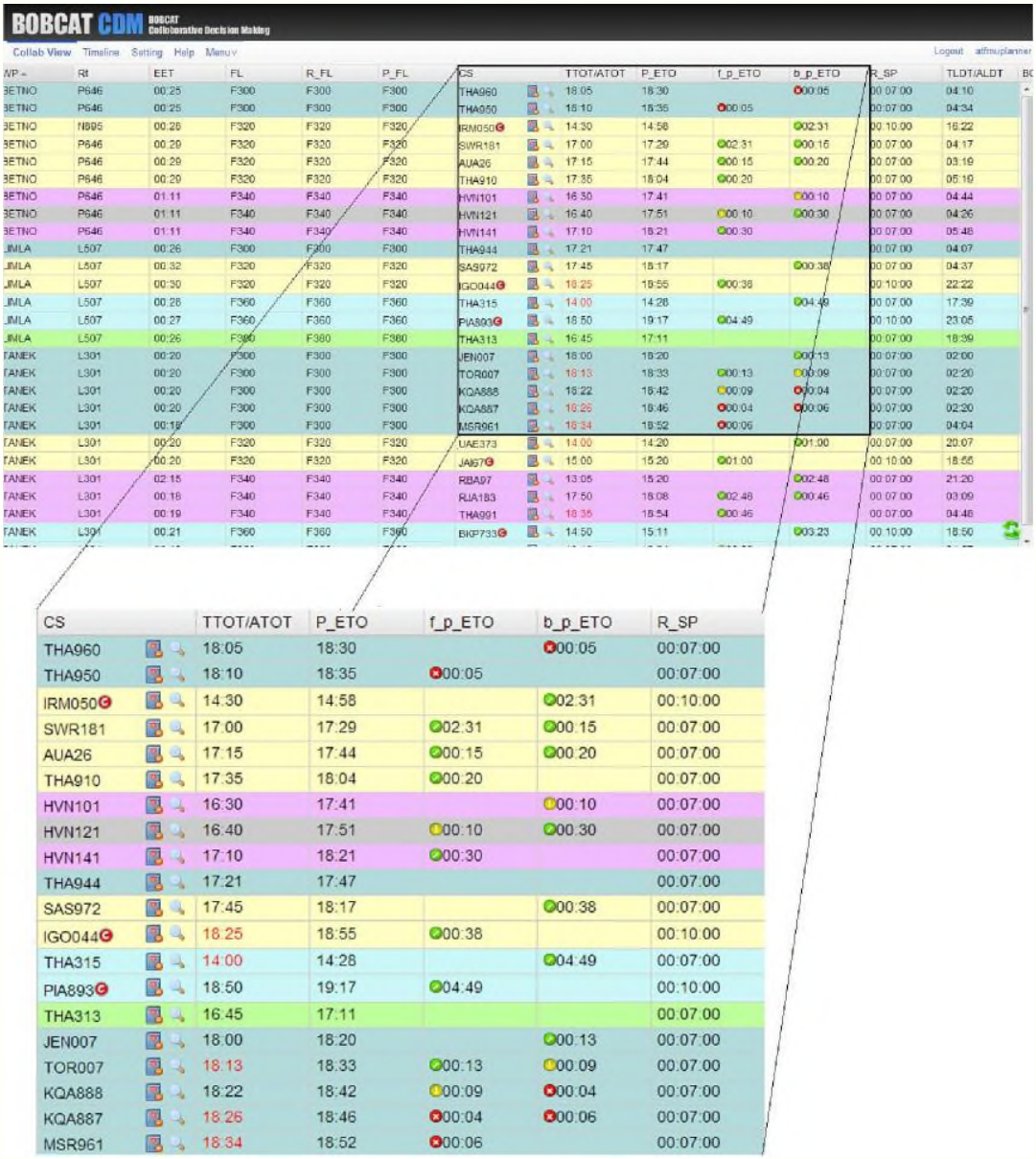

Figure 11. Example output of collaboration view

Figure 11 illustrates the example output, called collaboration view, resulting from the integration of ARSL into CDM. Focusing on Planned Estimated Time Over (P_ETO) of which the value is calculated from route, Estimated Elapsed Time (EET) contained in the flight plan, in addition with Target Take-Off Time (TTOT) of which the value is obtained from the collaboration of the planner and the aerodrome traffic controller. Once the calculation of P_ETO of each flight has been completed, CDM will line up the aircrafts and calculate the separation between the adjacent. However, each flight has individual

longitudinal required separation shown in R_SP, thus CDM has to ask Rules&Data service implemented in this work for the required separation. Three status will be displayed as the result of the comparison between the value of required separation and that of calculated separation. The red mark signals “alert” status, the yellow mark notifies “warning”, and the green mark indicates “normal”. For example, flight THA950, given the required separation is 7 minutes, if the calculated separation is less than 7 minutes, the red mark will then be assigned. In case the calculated separation is between 7 and 7 plus warning buffer time, the warning mark is then assigned; otherwise, the green mark of normal status is displayed. It is mandatory that the planner must adjust the plan, i.e. change TTOT, change flight level or change route, in case of alert status. For the warning status, the planner may opt to ignore the signal.

4. Evaluation

The evaluation of the correctness of CDM integrated with ARSL to return the required separation between aircrafts at the waypoints was carried out compared to the result from a specialist. A set of rules covering selected waypoints, i.e. BETNO, LIMLA, TANEK, and EKAVO, has been declared in the ARSL script. The script has been parsed and sent to the Rules&Data service. The results of required separation at selected waypoints compared to that from the specialist are reported in Figure 12.

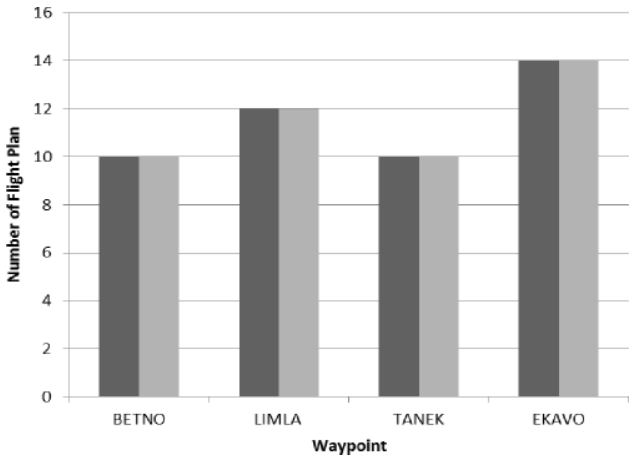

Figure 12. Evaluation result of ARSL correctness compared to a specialist

The number of random flight plans containing a selected waypoint is labeled on the Y axis. For example, the number of 14 random flight plans containing EKAVO waypoint was sent to the ARSL services. An example of random flight plan as a test case is shown in Figure 13.

```
(FPL-TOR007-IS
-B744/H-SDE1E2E3FGHIJ3J5M1RWY/LB1D1
-VTBS1745
-N0507F300 DCT BETNO P646 IBITA/N0503F320 P646 BBN R460 LLK M890
CHG/N0487F340 M890 SAMAR L509 TAPIS M875 AMDAR A466 TMD A66
ODIVA B824 DOKAM/N0475F360 B824 URL G3 FV R11 UK/N0483F380 R11
OPOKA M869 RIA N623 NEKET N616 XILAN DCT
-ESSA1025 EFTU
-PBN/A1B1D1L1O1S2 REG/HSTGL EET/VYYF0025 VECF0117 VIDF0240
OPLR0358 OAKX0436 UTSD0509 UTTR0544 UAII0556 UATA0623 UATT0641
UARR0714 UWWW0734 UWPP0753 UUWV0815 ULOL0910 EVRR0923 ESAA0958
SEL/FJAC CODE/887200 ORGN/VTBSTHAW PER/D RMK/TCAS AWUT1805)
```

Figure 13. Example of random flight plan

For safety reason, the evaluation's goal is all responses returned from ARSL services must be the same as the answers obtained from the specialist referring to the Aeronautical Information Package and its supplement [8]. The four waypoints of BETNO, LIMLA, PASVA, and EKAVO are selected because they are parts of the pilot route tested in the CDM project. The ARSL code used in the evaluation is shown in Figure 14.

Assuming that the answers from the specialist are 100% correct, the results of Figure 12 report that the answers from ARSL services are completely accurate compared to that from the specialist.

Next, the comparison between the size of ARSL script and the excerpts of AIP, i.e. A4/11 and A16/11, was evaluated. According to Figure 15, the dark-shaded bars represent the size of AIP excerpts, while the light-shaded bars represent the size of ARSL scripts. The findings report that the size of source document the authority needs to read is approximately six times larger than the contents contained in the ARSL scripts as the input source for automation processing that return the equivalent answers. Therefore, the ARSL and its services provide significant benefit in speed for data communication and use less storage compared to the manual approach.

```
Rules : WestRule1;
//CPDLC require if not yangoon FIR
HasEquipment : S, R, J;
PBN:A1;
FlightLevel: >270;
Separation : 7 mins, 50 nm;
Rules : WestRule2;
//VHF is required if not yangoon FIR
HasEquipment : S, V;
PBN:A1;
FlightLevel: >270;
Separation : 7 mins, 50 nm;
Rules : WestRule3;
//no CPDLC require when destination in Vangoon FIR
HasEquipment : R;
NoEquipment : N;
//destination in yangoon FIR
Destination : VYMM, VYYY, VYNT, VYBG, VYMD, VYHH, VYLS, VYHL, VYSH, VYTD, VYPN;
PBN:A1;
FlightLevel: >270;
Separation : 7 mins, 50 nm;
Rules : RNAVEquipped10Minute;
HasEquipment : R;
Separation : 10 mins, 80 nm;
Rules : WestRule4;
Separation : 10 mins, 80 nm;
Rules : Basic10Minute;
Separation : 10 mins, 80 nm;
Rules : Basic15Minute;
Separation : 15 mins, 120 nm;
targetFix :PASAT, SELKA, BUTRA, RAMEI, OKENA, BISOR;
TargetRules : Basic10Minutes;
targetFix :TAMOS;
TargetRules : Basic15Minute;
targetFix :PASVA;
TargetRules : RNAVEquipped10Minute;
targetFix :BETNO, LIMLA, TANEK;
targetTime : between 1200 and 1245;
TargetRules : WestRule1, WestRule2, WestRule3, Basic10Minute;
targetFix :AKATO;
TargetRules : WestRule1, WestRule2, Basic10Minute;
```

Figure 14. ARSL script for evaluation

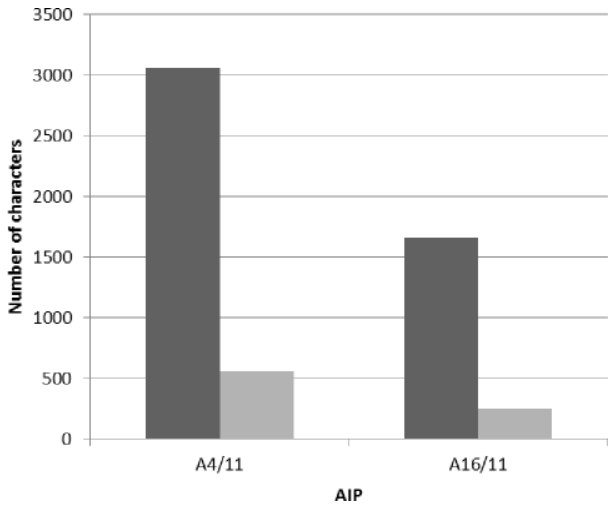

Figure 15. Comparison of size of AIP excerpts and ARSL scripts

5. Conclusion

In this research, Aeronautical Rules Script Language (ARSL), which is a domain specific language for aircraft separation minima determination, is designed to cover all information elements sufficient to determine the longitudinal separation between adjacent aircrafts. The lexicons comprising the language are defined based on the elements collected from International Civil Aviation Organization (ICAO) standard, associated with other extra elements from Bangkok FIR's Aeronautical Information Package (AIP) [8]. Since ARSL is text-based, it is easy to exchange data by merging to other standards such as AIXM. Moreover, ARSL promotes the ease of rule manipulation.

The evaluation results show that the implemented ARSL and its services are trustworthy and function accurately. The required separation returned from ARSL services is totally the same as that obtained from the specialist. In addition, ARSL scripts provide much more compact format compared to the parts of contents that the authority needs to read from AIP. It can be concluded that the ARSL and its services provide significant benefit in speed for data communication and use less storage compared to the manual approach.

The ARSL presented in this paper is capable for determining the longitudinal separation minima at waypoints only. Future research direction could be the enhancement of its capability to support determining the required separation at airports, and to support determining the vertical separation minima.

References

[1] EUROCONTROL:Aeronautical Information Exchange Model (AIXM), http://www.aixm.aero/public/subsite_homepage/homepage.html, (**2013**).
[2] Flight Information Exchange Model (FIXM), http://www.fixm.aero/, (**2013**).
[3] P. Comitz,. "A Domain Specific Approach to Aviation Data", Proceedings of Integrated Communications Navigation and Surveillance Conference, **(2010),** May 10-11; Westin Washington Dulles Herndon, VA, USA.
[4] M. Jiménez, F. Rosique, P. Sánchez, B. Álvarez, and A. Iborra, "A Hibitation: Domain Specific Language for Home Automation", IEEE SOFTWARE, vol. 26 issue 4, **(2009)**, pp. 30-38.

[5] M. Fowler. "A pedagological framework for Domain Specific Languages", IEEE SOFTWARE, vol. 26 issue 4, **(2009)**, pp. 13-14.
[6] P. Arpaia, L. Fiscarelli, G. Commara, and C. Petrone, "A Model-Driven Domain-Specific Scripting Language for Measurement-System Frameworks", IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, vol. 60 no. 12, **(2011)**, pp. 37-56.
[7] M. Fowler, "Domain Specifics Languages", Addison-Wesley Professional. Boston, **(2010).**
[8] AIP Thailand, **(2013)**, http://www.aisthai.go.th/webais/download_aip.php.
[9] Irony -.NET Language Implementation Kit, http://irony.codeplex.com/, (**2013**).
[10] International Civil Aviation Organization: Procedures for Air Navigation Services Air Traffic Management (Doc 4444), **(2007).**

Author

Sakon Sinlapakun

He received his bachelor degree in Electrical Engineering (major Telecommunication Engineering) from King Mongkut's Institute of Technology Ladkrabang in 2006. Currently, he is pursuing the master degree in Software Engineering at Chulalongkorn University, Bangkok, Thailand. He is involved in the Collaborative Decision Making project and Air Traffic Flow Management Information Support System project in Aeronautical Radio of Thailand Ltd.