A Framework for Generating Aviation Risk Information Mobile User Interfaces from Knowledge Graphs

Junli Liang1, Yalemisew Abgaz2, Rob Brenan2

1School of Computing, Dublin City University, Dublin Ireland
junli.liang4@mail.dcu.ie
2ADAPT Centre, School of Computing, Dublin City University, Dublin, Ireland
yalemisew.abgaz@adapatcentre.ie, rob.brennan@dcu.ie

Abstract. In this paper, we introduce a new framework called Flight Planner for generating Aviation Risk Information mobile-compatible user interfaces from Knowledge Graphs. The framework is divided into three components, a templating engine, a Linked Data Knowledge Graph and a mobile app. Traditional Linked Data frameworks provide poor support for user interface generation and common web frameworks do not natively support Linked Data-based Knowledge Graphs. Moreover, Aviation Risk Information systems is specialised area mainly studied by organisational psychologists or experts in human factors and it has received little attention by computer scientists. Thus, a flexible, efficient, and usable framework for Aviation Risk Information is needed. A comparative evaluation was done with Linked Data Reactor framework as a benchmark. It is observed that Flight Planner produces equivalently flexible and efficient, but less usable results despite being an early prototype.

Keywords: Linked Data. Templating engine. Mobile apps. SPARQL. UI generation. Aviation Safety Management Systems.

1 Introduction

Aviation Risk Information provides pilots and aviation operation staff with warnings of potential hazards or risks on their schedule. Aviation Risk Information is generated every day in huge amounts by many different data publishers [4]. Due to the heterogeneity of Aviation Risk Information, it is challenging to fuse many different sources. This leads to the issue that Aviation Risk Information is difficult to be generated in a human-readable form. Such information can be created in different standards or forms. Aviation Risk Information is also dynamic. Such information in the industry can be represented as a report e.g. flight safety report. Different types of reports are dynamically linked together by some specific relationships. Generating such types of reports are critical. Knowledge Graphs can be used for data fusion in this situation. However once the information is in the Knowledge Graph, role or task-specific sections needs to be presented to users in a way that is easy to interpret. Mobile devices are the most flexible way to display this information to staff who are not often at a traditional computing terminal.

Templating engines provide an approach to flexibly combine data for users. From the state of the art, researches are ongoing into templating engines, Linked Data and mobile apps. However, Aviation Risk Information is a specialised field with specific established ontologies and this could be exploited by a domain-specific templating engine. Web-based, flexible and efficient mobile user interface tools for aviation Knowledge Graphs are immature. Therefore, a new framework for Aviation Risk Information needs to be developed.

The research question explored in this paper is "*to what extent a framework can be developed to generate Aviation Risk Information user interfaces from Knowledge Graphs in terms of flexibility, efficiency and usability*". The developed framework is called Flight Planner, which addresses issues: (i) dynamic and heterogenous Aviation Risk Information, (ii) generation of Aviation Risk Information and (iii) immature UI tools for Knowledge Graphs. Three general technical approaches are used: (i) the NASA Air Traffic Management ontologies and graph-based database, (ii) a flexible templating engine and (iii) mobile-compatible user interfaces powered by web technologies. The paper offers a couple of contributions listed below:

- An efficient and flexible templating engine for flexibly generating Aviation Risk Information user interfaces from Knowledge Graphs
- A usable and professional aviation-specific user interface element for an Aviation Risk Information circulation system.
- A usable mobile-compatible user interface for browser-based applications.

The remains of this paper is divided into five sections. Firstly, the related work section defines the gap addressed in current research about Linked Data, templating engines, mobile apps and Aviation Risk Information system. Secondly the use cases and requirements section indicates the intended user group for the system and the requirements of the framework needs to be met in terms of flexibility, efficiency and usability. Thirdly the system design section describes how the framework is implemented and constructed. Fourthly, the evaluation section describes a set of methods used to evaluate Flight Planner by comparing with the Linked Data Reactor[1] framework developed by Khalili, Ali, et al [4]. A conclusion and future work are provided at the end of paper.

2 Related Work

This section describes related work for four different areas: Linked Data, templating engines, mobile apps and Aviation Risk Information circulation systems. The missing gap among them is defined at the end of the section.

2.1 Linked Data

Linked Data (also known as RDF data) is a technique for publishing structured data on the web. It enables data linked with others in the form of subject -> predicate -> object called a triple described by Bizer, Christian, et al [2] where predicate is a relationship that the subject and object are linked by. An amount of triples forms a graph known as a Knowledge Graph and SPARQL is querying language used to manipulate data over

[1] Linked Data Reactor: http://ld-r.org/

Knowledge Graphs. One of the implementation of Knowledge Graphs is Apache Jena[2], which is a graph-based database integrated with SPARQL.

The challenges described by Keller, et al [4] for Aviation Risk Information could be seen such as different data encoding format, different data field naming and data semantics (different meanings for two identical labels of aviation risk information), which results the heterogeneity to Aviation Risk Information. These issues can be addressed by mapping Aviation Risk Information in Linked Data by the NASA Air Traffic Management ontologies. Keller, et al have defined a set of ontology namespaces[3] including, *atm*, *nas*, etc.

For representing Linked data on a UI, one of the ways is to use RDForms.js[4], which is an open source project designed for representing Linked Data in RDF/JSON[5] on a form without a huge of amount of semantic web knowledges.

2.2 Templating Engine and UI Generation

A templating engine by definition is a tool to support automatic data generation and population into a template (e.g. a form or survey). A templating engine researched by Tatsubori, Michiaki, et al [8] is an early developed templating engine used in the web development. Tatsubori, Michiaki, et al state that a templating engine is mostly used in the web development or browser-based application. An approach of a templating engine for generating questions and answers in Linked Data was introduced by Unger, Christina, et al. [9]. It is an approach to generate questions and matching answers based on parameterised triples in SPARQL queries.

UI generations are related to templating engine. However, templating engines mostly are for developers, which is an obvious difficulty for non-developers to use. Palmieri, Manuel, et al [7] introduced a semi-automatic framework for UI generation for Linked Data. It is a framework producing UI components by selecting required Linked Data over the Web. There is a templating engine handling Linked Data population and UI generation. Linked Data Reactor (LD Reactor) developed by Khalili, Ali, et al [5] is also a framework designed for managing and generating UIs for Linked Data. It provides a templating engine that allows UI designers to generate UIs by modifying configuration files. The Linked Data Reactor has already been used by a couple of companies (RISIS[6] and Open PHACTS[7]).

2.3 Mobile Apps

Mobile developments have been seen a huge increase since Android and iOS were released. According to M.S Ferreira, Cristiane, et al [6], there are three types of mobile applications: native mobile apps, cross-platform apps and mobile web apps. A mobile web app can be referred as the browser-based app. One of the browser-based app in the Linked Data area is DBpedia mobile developed by Becker, Christian, et al [1], which

[2] Apache Jena: https://jena.apache.org/
[3] The NASA ATM: https://data.nasa.gov/ontologies/atmonto/
[4] RDForms.js: https://rdforms.org/#!index.md
[5] RDF/JSON: https://www.w3.org/TR/rdf-json/
[6] RISIS: http://risis.eu/
[7] Open PHACTS: http://openphacts.org/

is a mobile app using semantic and web technologies (i.e. DOM) to provide location-based services on mobile devices.

2.4 Aviation Risk Information System

The Alitalia app [11] is an Aviation Risk Information circulation system case-studied by the airline Alitalia and conducted by the Centre for Innovative Human Systems in the School of Psychology, Trinity College Dublin. It describes a number of professional UI element designs specified by EREA, the association of European Research Establishments in Aeronautics about aviation reporting systems called flight block reports (a form of risk information circulation). Blocker Reports record information (e.g. incidents, safety, action preceded for the report, etc.) for flights. This is an inflexible, siloed and brittle system that cannot easily adapt to changes in the sources of risk information or the user interface changes requested by psychology researchers as they gradually adjust the interface to improve understandability. Hence a much more flexible approach is needed to enable rapid interface change over a flexibly represented knowledge graph that performs a data fusion function. Despite these demands, the performance of the system must not be compromised.

There is a significant gap for quickly building Aviation Risk Information mobile apps. The dynamic and heterogeneity of Aviation Risk Information is suitable to be dealt with by Knowledge Graphs. However, there is a missing framework for dynamically and flexibly generating aviation risk information mobile-user interfaces from these Knowledge Graphs.

3 Use Case and Requirement

In this section, use cases and associated requirements are described. The users for the framework are identified as developers and aviation risk domain experts. Use cases for the target users are listed below. In order to provide use cases below, the framework needs to meet requirements in terms of flexibility, efficiency and usability.

Use Case 1:	A UI for Aviation Risk Information can be configurable.
Use Case 2:	The features of a sample aviation-risk information app (Blocker Reports [11]) developed for safety experts must be supported.
Use Case 3:	A UI produced by the framework is mobile-compatible.

3.1 Flexibility

The flexibility implies whether the framework provides specific requirements defined as follows. A UI is configurable: a template defines UI structure for data. UI generations of Khalili, Ali, et al [5] and Palmieri, Manuel, et al [7] are based on a template. For the use case 1, the framework needs to be flexible to provide configurable UIs, which means that the framework needs to make a template configurable. For example, the framework should allow users to configure a template to be able to configure the UI1. There are two ways to implement the configurable UIs. (i) The first one adapted by LD Reactor Khalili, Ali, et al [5] is to provide a configuration file, which can be treated as a template. (ii) The second way adapted by Palmieri, Manuel, et al [7] is to provide clickable UI components for users to define a template.

A SPARQL query is configurable [9]: a configurable UI generation is based on SPARQL query generation. A template as described above can define the structure of

UIs, while a query template defines what data to query from e.g. a database. The approach by Unger, et al [9] states that a generation of SPARQL queries is based on a SPARQL query template with slots. A slot is defined as a triple of parameters that can be dynamically set. The triple of subject, predicate and object can be treated as parameters in the string format which then this SPARQL query as a template could possibly generate different values.

3.2 Efficiency

The efficiency implies how easy the functionalities are performed by users which define as follows. UI generation is efficient: Bizer et al [2] state that UI generation is one of the challenges for the current state of Linked Data Applications. The framework should provide an efficient and easy way for users to generate data on UIs. The efficiency is the level of resources consumed in performing tasks stated by Brooke, John [10], where resources include the amount of time and iterations. A preferred way adapted by Pazienza, et al [7] is to provide clickable UI buttons.

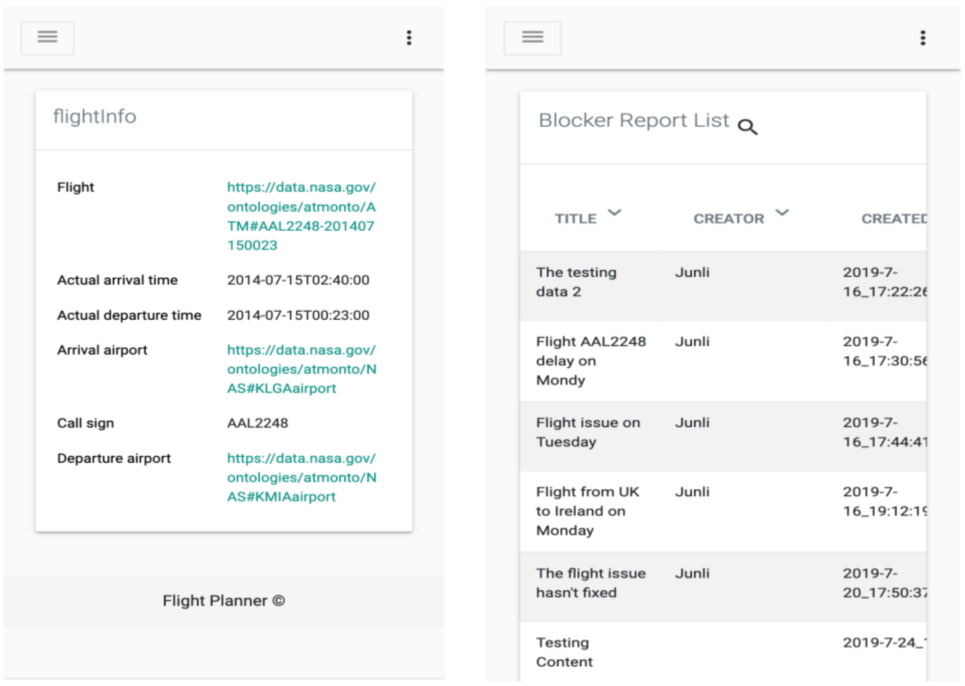
3.3 Usability

According to the Alitalia app [11] described in the section 2.4, an Aviation Risk Information circulation system needs to provides aviation-specific UI elements. Referring the use case 2, the Flight Planner framework is required to implement the Blocker Report functionality adapted from [11] for the aviation risk domain experts.

For the use case 3, the Flight Planner framework is to provide mobile-compatible UIs, which is a factor to affect the usability level. The precise usability level of a system is affected by the intended users, and the measurement of usability can vary widely. But the general usability of a system can be measured by the effectiveness and satisfaction in System Usability Scale (SUS) described by Brooke, John [10]. Effectiveness [10] means *"the ability of a user to finish tasks, and the quality of the outputs generated from tasks"*. Satisfaction [10] means *"a user's subjective reaction to using a system"*. To meet usability requirement, Flight Planner needs to have significant scores in SUS.

4 System Design

This section describes in detail the implementation and technical design of the Flight Planner framework which is shown in the design Fig. 2. Screenshots of two implementation apps, flight reports on the left hand side and Blocker Reports on the right hand side show in Fig. 1. The Flight Planner framework is constructed by a server and client side described in two subsections below respectively.



flightInfo

Flight	https://data.nasa.gov/ontologies/atmonto/ATM#AAL2248-201407150023
Actual arrival time	2014-07-15T02:40:00
Actual departure time	2014-07-15T00:23:00
Arrival airport	https://data.nasa.gov/ontologies/atmonto/NAS#KLGAairport
Call sign	AAL2248
Departure airport	https://data.nasa.gov/ontologies/atmonto/NAS#KMIAairport

Flight Planner ©

Blocker Report List

TITLE	CREATOR	CREATED
The testing data 2	Junli	2019-7-16_17:22:2
Flight AAL2248 delay on Mondy	Junli	2019-7-16_17:30:5
Flight issue on Tuesday	Junli	2019-7-16_17:44:4
Flight from UK to Ireland on Monday	Junli	2019-7-16_19:12:1
The flight issue hasn't fixed	Junli	2019-7-20_17:50:3
Testing Content		2019-7-24_

Flight Report List of Blocker Reports

Fig. 1. Flight Planner Interface

4.1 Server Side

The server side is constructed by the Apache Jena and a server written in Node.js. The Node.js sever shown in the Fig. 2 consists of a SPARQL module, UI generator, and request and response handlers. The general process of UI generation is that when the request handler receives a request to generate a UI, the Template Analyser is told to generate a template (a file having empty entries to be filled with data), and the SPARQL module is told to query requested Aviation Risk Information from Jena via SPARQL. After that, a template and queried data are passed to the UI generator, which is for composing the template and Aviation Risk Information into a UI template. It is a template populated with data. The UI template is defined as a configuration file by RDForms.js. Once the UI template is composed, the client side could use it to render a UI on a mobile device by RDForms.js APIs.

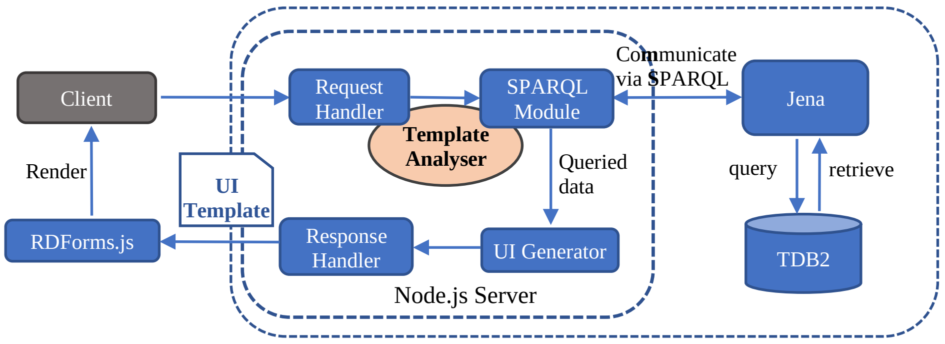


Fig. 2. Templating-based System Design Diagram

To make the UI configurable, the generation of a template by the Template Analyser and queried Aviation Risk Information need to be based on the request, and be matched in order to generate a correct UI template. For example, the Fig. 3 shows the snippet of a UI template/configuration file where *flightInfo* is the type of Aviation Risk Information containing elements *actualArrivalTime* and *arrivalAirport* as data labels. An initial template generated by the Template Analyser contains only data labels (*actualArrivalTime*, etc). The *values* and *types* under data labels are to be queried from Jena. Both data labels and queried data (*types* and *values*) need to be matched. More specifically, the SPARQL module provides a SPARQL query template adapted from Uger, et al [9] that is to be dynamically filled with triples (based on a request) as parameters. The generated UI template can be manually modified by users as to change the UI representation.

```
"flightInfo": {
  "actualDepartureTime": [{
    "value": "2014-07-15T00:37:00",
    "type": "literal"
  }],
  "arrivalAirport": [{
    "value": "https://data.nasa.gov/ontologies/atmonto/NAS#KLGAairport",
    "type": "uri"
  }],
},
```

Fig. 3. A snippet of a UI template/configuration file

4.2 Client Side

For the client side, it is a browser-based app launched on mobile devices' browser tested in the Google Chrome web browser. A UI template generated by the server is compiled into human-readable forms displaying on devices. According to stated requirements, the framework needs to provide a mobile-compatible UI. The Google Material Design for Bootstrap[8] theming framework is used to provide UI components that are styled and shaped into mobile native UI components. Slider and menu components can provide mobile app behaviours. To make the UI more usable in terms of effectiveness and satisfaction, a traditional web UI component is avoided such as selection with large

[8] Bootstrap for Google Material Design: https://fezvrasta.github.io/bootstrap-material-design/

amount of data, datetime picker, breadcrumbs navigation, etc, as they belong to desktop UI components illustrated by the Google Material Design guide.

5 Evaluation

To answer the research question and evaluate the Flight Planner framework, the framework is compared to Linked Data Reactor. This section firstly introduces the dataset and the participants, then followed by hypotheses. Secondly, experimental design and interpretation are described in terms of flexibility, efficiency and usability. Lastly, based on the results, suggested changes are discussed at the end.

5.1 Dataset and Participants

The dataset contains 100 sample flight instance data that have been converted into Linked Data using the NASA ATM ontologies. 10 participants from the IT background and 2 Aviation Risk Information experts from Trinity College Dublin who have more than 20-years' experience in Aviation Risk Information area are invited to evaluate the Flight Planner framework.

5.2 Null Hypotheses

Flexibility:	Flight Planner can provide at least as many flexibilities as LD Reactor provides, including a configurable UI and a configurable query.
Efficiency:	There is no significant difference between two frameworks, in terms of the amount of time spent and trials for participants to perform tasks.
Usability:	Flight Planner can score above 68.2, an average score in SUS.

5.3 Flexibility

Experimental Design

Flight Planner was compared with LD Reactor in terms of the following flexibility requirements:

- A configurable template: two frameworks are tested whether providing a configuration file or clickable UI components by examining the code level.
- A configurable SPARQL query: two frameworks are tested whether they can dynamically generate SPARQL queries based on requests by examining their SPARQL modules and generated SPARQL queries.

Results and Interpretation

Two frameworks are equivalently flexible. By examining two frameworks' code levels, two frameworks can provide configurable UI generation. Flight Planner provides a configuration file (called a UI template described in section 4) and clickable UI buttons to achieve the configurable UI generation, while LD Reactor provides multiple configuration files. Also, both frameworks can provide configurable SPARQL query generation. A query from LD Reactor's SPARQL module is created by several blocks of statements and each block can be dynamically generated. The query template from Flight Planner is created by a number of pre-defined SPARQL sub-queries. Two SPARQL generations are based on a configurable triple similar to Unger, et al [9].

5.4 Efficiency

Experimental Design

10 participants (5 participants to test Flight Planner and another 5 participants to test LD Reactor) were invited to perform two sets of tests listed in Fig. 4. Two sets of tasks are not totally identical due to different use cases of two frameworks, but they are managed to be as related as possible. The equal division of 10 participants can produce records used by a t-test at the last step of procedure. The testing procedure is listed below:

- A single evaluation test of a participant is limited up to 30 minutes, including introduction and presentation, 2-3 minutes getting familiar with and the tasks.
- The "think aloud" methodology [12] was used to let participants speak whatever they are thinking while they are performing the tasks.
- The number of iterations a participant used to get correct answers, the total time spent, and their thoughts and comments were recorded. One iteration is a failed execution including a navigation to an incorrect page, a click on an incorrect functional button, incorrect operation even on a correct functional UI (e.g. typing in an incorrect format for searching on a searching field), etc.
- An independent t-test was conducted on two sets of results to determine whether the null hypothesis can be accepted or not by calculating a p-value.

Task Id	**Common Tasks**	
1	Query a flight with flight number UAL1479 and find the arrival airport for that flight.	
2	Query a flight with flight number WJA1212 and find the airline for that flight.	
3	Query a flight with a flight number DAL1776 and find its arrival airport IATA code.	
	Flight Planner	**LD Reactor**
4	Configure the flight report showing the following information: - The flight and flight callsign - The arrival airport name and airport FAA code - The departure airport name and airport ICAO code - The airline name and the headquarter	Query the CEO (a key people) of an airline called Aer Lingus.
5	Sort the list of Blocker Reports by the created date and find the latest submitted report.	Sort the list of airlines by year founded and find the latest founded one.
6	Search the list of Blocker Reports for a created date 2019-7-20U17:50:37.	Search the list of airlines for a founded year 2002.

Fig. 4. Task table

Results and Interpretation

Two sets of results are recorded shown in Fig. 5 below. The columns named "ldr" and "fp" represent LD Reactor and Flight Planner, respectively. Each cell represents data of time spent or the number of trials for an individual participant.

UID	**ldr**	**fp**
1	11.26	14.58
2	21.39	7.42
3	6.02	6.26
4	10.54	14.26
5	14.17	19.04

a. Time spent

ldr	**fp**
8	9
17	5
13	6
6	5
8	7

b. # Trials for 6 tasks

ldr	**fp**
2	5
10	5
2	3
2	3
3	3

c. # Trials for first 3 tasks

Fig 5. Time spent and trial results

The t-test results for time spent and the number of trials with significant level 0.05 is shown in Fig. 6. According to two sets of tasks stated in the analysis methods, the t-test can be conducted for first 3 identical tasks and for all tasks. Comparing to the significant value 0.05, the p value for time spent is greatly high which there is no significant difference between two frameworks in terms of time spent to perform tasks. The p value for first 3 tasks is relatively high and the one for all tasks shows less significance due to last 3 nonidentical tasks, but it still can deliver high enough significance. Thus, the null hypothesis for efficiency can be accepted.

	ldr mean	**fp mean**	**d.o.f**	**T statistic**	**P value**
Time spent	12.585	12.52	8	0.96633	0.925395
All tasks	10.4	6.4	8	1.860968	0.099782
First 3 tasks	2.6	3.8	8	1.784436	0.813118

Fig. 6. T-test results

5.5 Usability

Experimental Design

For usability, after an efficiency test finishes, participants were given a SUS survey[9] to test Flight Planner's and LD Reactor’s usability. Feedback and suggestions were also be obtained from participants in terms of whether the Flight Planner app is able to provide a mobile-compatible user interface. Two Aviation Risk Information experts were invited to an interview to evaluate whether the Flight Planner framework is able to provide sample aviation-specific app. Introduction slides, app demo and questions were conducted for 30 minutes. Comments and suggestions were recorded during the interview.

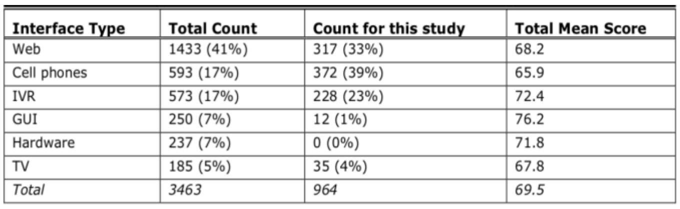
Results and Interpretation

After participants finish SUS survey, two sets of scores can be calculated shown in Fig 7 by the formula provided by Brooke, John [10]. According to the benchmarks conducted by Bangor, et al [18] shown in Fig 8, the total mean of the SUS scores is 69.5 collected from 3463 SUS surveys and the interface type suitable for two frameworks is Web with the mean score 68.2. The SUS means of both frameworks are greater than 68.2, which means that the null hypothesis for usability can be accepted. Additionally, the SUS mean of ldr is greater than Flight Planner’s. And, mobile-compatible UIs are accepted by 5 participants who evaluated Flight Planner framework.

	Sus1	**Sus2**	**Sus3**	**Sus4**	**Sus5**	**Mean**
ldr	72.5	90.0	77.5	75.0	87.5	80.5
fp	75.0	67.5	82.5	87.5	72.5	77.0

Fig. 7. SUS score results

[9] SUS template: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html



Interface Type	Total Count	Count for this study	Total Mean Score
Web	1433 (41%)	317 (33%)	68.2
Cell phones	593 (17%)	372 (39%)	65.9
IVR	573 (17%)	228 (23%)	72.4
GUI	250 (7%)	12 (1%)	76.2
Hardware	237 (7%)	0 (0%)	71.8
TV	185 (5%)	35 (4%)	67.8
Total	*3463*	*964*	*69.5*

Fig. 8. The overall SUS scores table

5.6 Assessment by Aviation Risk Information Circulation System Domain Experts

According to the comments from two experts with over 40 years combined experience, the implementation of Blocker Reports is acceptable for safety event reporting and mobile-compatible UIs are also accepted by two experts. However, there are two missing functionalities in terms of aviation-specific elements. The first one is an integration with predictive data analytic engine. The second one is a second-layer interface for Blocker Reports, which is a receiver of Blocker Reports provides action recommendation by (1) tracking a flight safety issue, (2) reviewing the issue, (3) analysing the issue and (4) suggesting a solution. This functionality requires different types of reports to handle phases (1) – (4) and they would be then linked together. Safety reports' status are monitored in phase (1) and (2). The analytics engine would be used in phase (3) and (4). Thereby, the UI generation needs to circulate the process from reporting a Blocker to suggesting a solution, which is missed in Flight Planner considered by experts.

6 Conclusion and Future Work

The Flight Planner is a framework for generating Aviation Risk Information user interfaces from Knowledge Graphs in terms of flexibility, efficiency and usability. It fills in the missing gap among Linked Data, templating engines and mobile apps. With a comparative evaluation to the LD Reactor, the Flight Planner is able to be equivalently flexible and efficient but less usable. Also, UIs produced by the framework are mobile-compatible accepted by invited participants.

However, the framework has not fully meet the circulation described in the section 5.6, which is significant for an Aviation Risk Information circulation system. The first layer has been implemented, which is reporting safety information. Thus, our plan for the future work is to implement the second layer, an action recommendation consisted of phases (1) – (4) described in the section 5.6. Besides, we plan to make two layers working in a circulation way, which a suggested solution in phase (4) may have impact for a safety issue in the safety reporting layer of other circulation processes. More importantly, the whole process needs to be integrated into the mobile-compatible UI and Knowledge Graphs.

Acknowledgement

This research has received funding from the Enterprise Ireland Commercialisation Fund (CF2018-2012) and co-funded by the European Regional Development Fund under Ireland's European Structural and Investment Funds Programme 2014-2020.

References

1. Becker, Christian, and Christian Bizer. ‘Exploring the Geospatial Semantic Web with DBpedia Mobile’. *Journal of Web Semantics*, vol. 7, no. 4, Dec. 2009, pp. 278–86. doi:10.1016/j.websem.2009.09.004.
2. Bizer, Christian, et al. ‘Linked Data - The Story So Far’: *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, July 2009, pp. 1–22. doi:10.4018/jswis.2009081901.
3. Hartig, Olaf, et al. ‘Executing SPARQL Queries over the Web of Linked Data’. *The Semantic Web - ISWC 2009*, edited by Abraham Bernstein et al., vol. 5823, Springer Berlin Heidelberg, 2009, pp. 293–309. doi:10.1007/978-3-642-04930-9_19.
4. Keller, Richard M. ‘Ontologies for Aviation Data Management’. *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–9. doi:10.1109/DASC.2016.7777971.
5. Khalili, Ali, et al. ‘Adaptive Linked Data-Driven Web Components: Building Flexible and Reusable Semantic Web Interfaces’. *The Semantic Web. Latest Advances and New Domains*, edited by Harald Sack et al., vol. 9678, Springer International Publishing, 2016, pp. 677–92. doi:10.1007/978-3-319-34129-3_41.
6. M. S. Ferreira, Cristiane, et al. ‘An Evaluation of Cross-Platform Frameworks for Multimedia Mobile Applications Development’. *IEEE Latin America Transactions*, vol. 16, no. 4, Apr. 2018, pp. 1206–12. doi:10.1109/TLA.2018.8362158.
7. Pazienza, Maria Teresa, et al. ‘Semi-Automatic Generation of GUIs for RDF Browsing’. *2010 14th International Conference Information Visualisation*, IEEE, 2010, pp. 267–72. doi:10.1109/IV.2010.47.
8. Tatsubori, Michiaki, and Toyotaro Suzumura. ‘HTML Templates That Fly: A Template Engine Approach to Automated Offloading from Server to Client’. *Proceedings of the 18th International Conference on World Wide Web - WWW ’09*, ACM Press, 2009, p. 951. doi:10.1145/1526709.1526837.
9. Unger, Christina, et al. ‘Template-Based Question Answering over RDF Data’. *Proceedings of the 21st International Conference on World Wide Web - WWW ’12*, ACM Press, 2012, p. 639. doi:10.1145/2187836.2187923.
10. Brooke, John. *“SUS-A quick and dirty usability scale.” Usability evaluation in industry*. CRC Press, 1996. Available: https://www.crcpress.com/product/isbn/9780748404605.
11. *A FAA Exploitation Plan For Case Study In ALITALIA*, a document describing an Aviation Risk Information circulation application.
12. Krahmer and N. Ummelen, "Thinking about thinking aloud: a comparison of two verbal protocols for usability testing," in *IEEE Transactions on Professional Communication*, vol. 47, no. 2, pp. 105-117, June 2004. doi:10.1109/TPC.2004.828205