iStarJSON: A Lightweight Data-Format for i* Models

Oscar Franco-Bedoya[1,2,3], David Ameller[1], Dolors Costal[1], and Lidia López[1]

[1] Group of Software and Service Engineering (GESSI)
Universitat Politècnica de Catalunya, Barcelona, Spain
[2] Universidad de Caldas, Manizales, Colombia
[3] Universidad Nacional de Colombia, Manizales
{ohernan,dameller,dolors,llopez}@essi.upc.edu
http://www.essi.upc.edu/~gessi/

Abstract. JSON is one of the most widely used data-interchange format. There is a large number of tools open for modelling with i*. However, none of them provides supporting for JSON. In this paper we propose iStarJSON language, a JSON-based proposal for interchanging i* models. We also, present an open source software that transforms XML-based format models to JSON models that expose a set of web services for mining iStarJSON models.

Keywords: iStarML, JSON, iStar, i-star, i* modelling, REST

1 Introduction

i* is a well-known framework for goal and agent-oriented modelling and reasoning [1]. Throughout the years, different research groups have proposed i* language variations and several tools for the i* framework [2]. iStarML is an XML-based format for enabling interoperability among i* tools [3]. In order to keep i* up-to-date with new trends in software development (e.g., apps, JavaScript, REST, NoSQL), we proposed the basic structure of iStarJSON, a JSON-based model interchange language for i* models. In the last years, JSON has emerged as the new standard format for exchanging data in both academy and industy. Additionally, there is a large numbers of tools openly available in the web that provide transformations between different data representation formats and JSON. However, most of these approaches only provide a single mapping in the data structure through individual schemas. In addition, we have developed an open source software set of RESTful services that exposes the functionality needed to transform iStarML, a XML-based format, models to JSON models. Furthermore, this tool not only allows mapping from XML to JSON but also change the hierarchy structure of iStarML schema for a lightweight graph structure. This transformation does not affect the i* syntax and semantics. However, it allows to use a variety of tools and algorithms for graph manipulation (e.g., graph layouts, node centrality, shortest path). As a particular use case we have developed a web service for i* model visualization using different layouts. Using JSON to represent i* models as a graph enables the i* researchers and practitioners communities to develop new tools based on graph theory and the simplicity of the JSON-based programing. This tool has been carried out in the context of a general framework for supporting the open source software ecosystems (OSSECOs) modelling process.

This paper presents JSON-based iStarJSON model interchange proposal is presented. The objectives of this proposal are: (1) to keep i* up-to-date with new trends in software development and (2) to present an easy programming, flexible and lightweight data format for i* models.

2 Background on iStarML and JSON

iStarML is a XML-based interchange format for i* diagrams. It includes six basic categories to represent the i* abstract concepts: actor, intentional element, dependency, boundary, intentional element link and actor association [3], [4]. iStarML is also a powerful vehicle that enables interoperability inside the i* community [5]. The iStarML proposal has been widely discussed in several papers: [2], [6], [5] and [3].

JavaScript Object Notation (JSON) is a lightweight data-interchange format that is less verbose and more readable than XML [7]. It has become predominant in the mobile and browser domain in the last few years. JSON and XML are conceptually similar [8], however XML leads to problems with large models in terms of scaling. On the other hand, JSON data format supports high scalability [9].

3 iStarJSON Schema

A schema that serves as a means to understand the structure of a document is as relevant for JSON as it is for XML. But there is little enthusiasm in the JSON community for schema languages, especially complex schema languages [10] . However, in the last three years a set of methodological proposals to promote the definition of a standard JSON schema have emerged[4].

A JSON schema is a JSON-based grammar declaration used to describe the structure and content of a JSON document. this schema provides a contract specifying which JSON data is required for a given application and how to interact with it. A JSON schema is intended to define validation, documentation, hyperlink navigation, and interaction control of JSON data.

In iStarJSON we represent a i* model as a directed graph $G = (V, E)$ where nodes V correspond to i* intentional elements and actors:

$$V = \{actor, agent, role, position\} \cup \{goal, task, resource, softgoal, belief\}$$

The set of edges $E \subseteq E \times E$ represents relationships connecting nodes. The type of relationships that can exist between nodes is determined by the links defined by iStarML:

$$Atype \in \{association, dependency, means\_end, decomposition, contribution\}$$

In contrast to the other types of links, where one link involve two nodes, dependencies are represented by three nodes (depender, dependee and dependum) and two links, one from the depender to the dependum and other from the dependum to the dependee.

[4] `http://json-schema.org/`

Fig. 1 show a partial view of the iStarJSON schema. [5]

```
{
  "$schema": "http://testoneosseco.azurewebsites.net/json/schemas/istar/schema#",
  "properties":      { "diagram": {"type": "string", "name": "diagram"},
                       "modelType": {"type": "string","enum": ["rationale","dependence"]
                     },

  "nodes":           {"items": {"properties": {"id": {"name": "id"}, name": {"name": "name"},
                      "elementType": {"enum": ["actor", "goal", "task", "resource", "softgoal",
                      "belief"],"name": "elementType"},"boundary": {"name": "boundary"}}}
                     },

  "edges":           {"items": {"properties": {"source":{"name": "source"},"target": { "name":
                      "target"}, "linkType": {"enum": ["association", "dependency",
                       "means_end", "decomposition", "contribution"]}}}}
                     }
}
```

Fig. 1: partial view of the iStarJSON schema

3.1 Representing i* Models in iStarJSON

In order to illustrate how the iStarJSON structure deals with i* representations, two examples are shown inf Fig. 2. The first example, illustrates a basic Strategic Dependency diagram with 3 actors and 4 intentional elements. All of them are mapped into iStarJSON array property *nodes*. The property *boundary* of each node has the default value *"none"* indicating that the elements are in the general environment of the ecosystem. The second example corresponds to an Strategic Rationale diagram, the iStarJSOn file is similar to that of the first example, but in this case, the property boundary of each node has as value the name of the actor where the node is inside.

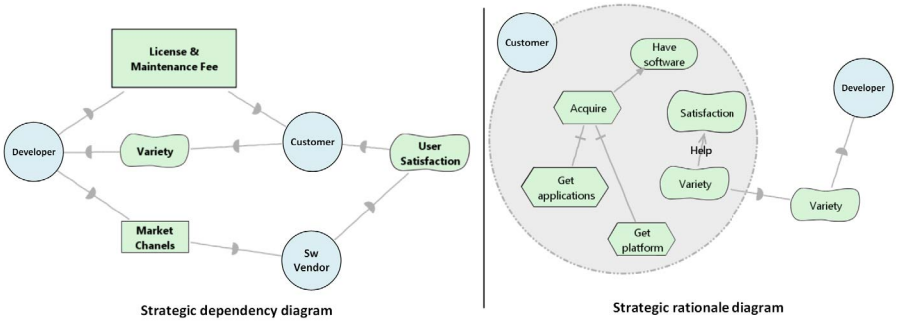

Fig. 2: An example of software ecosystem modeling using i* (excerpted from [11])

3.2 The iStarJSON Tool

iStarJSON exposes two main services. The service *textiStarJSONValidator*, uses the iStarJSON schema for the validation of the JSON document with the i* model, other

[5] iStarJSON schema can be downloaded from: `http://pastebin.com/Z6CTPgMF`

The iStarJSON code for the *i** strategic dependency diagram The iStarJSON code for the *i** rationale dependency diagram

Fig. 3: Representing i* Models in iStarJSON

service, creates a valid iStarJSON document from a valid iStarML file. Additionally, there are a set of specific services that provide functionality for the i* graph-model[6]. Fig. 4 shows the architecture of iStarJSON

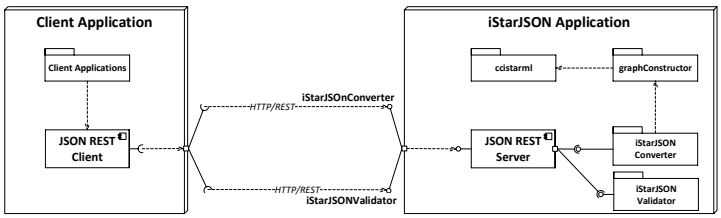

Fig. 4: Overview of the iStarJSON architecture.

We have implemented a simple program to transform a iStarJSON document to *DOT-code*[7]. This allows to draw directed graphs as hierarchies (see Fig. 5). iStarJSON is composed by the following components:

- *JSON REST server component:* handles REST request from the client application and calls internal services.
- *ccistarml package:* [6] allows creating, reading and modifying iStarML files.
- *graph construct package:* creates a dynamic graph structure from the iStarML file.
- *iStarJSON converter package:* transforms the dynamic graph structure into a valid JSON file.
- *iStarJSON validator package:* validates the JSON input document using the iStarJSON schema.

[6] availables in `http://testoneosseco.azurewebsites.net/iStarJSONServiceREST/istar/`

[7] DOT is a plain text graph description language. It is a simple way of describing graphs that both humans and computer programs can use

iSarJSON is an open source software project and its source code is freely available as a GitHub repository[8].

4 Proof of Concept: Graph Visualization

In this section we show a proof of concept of the iStarJSON tool. We have developed a REST web service that creates a visual representation of i* models by transforming iStarJSON files into a variety of output formats (e.g., PostScript, PDF, SVG, PNG, GIF) and using different types of layouts (i.e., dot, neato, sfdp,fdp, twopi and circo). To do that, we have created a *DOT* program from the iStarJSON document. For drawing the models we have used GraphViz an open source graph visualization software with a heterogeneous collection of graph drawing tools [9].

Fig. 5 shows the DOT programs and the graphics representation of the iStarJSON example 2.

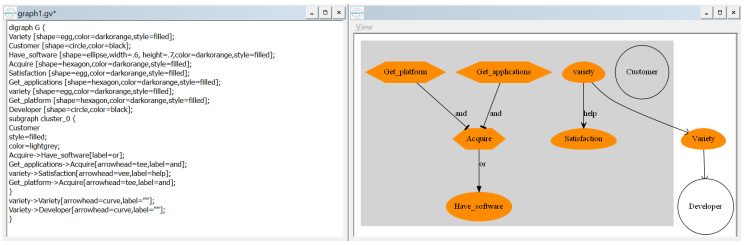

Fig. 5: Representing i* Models in DOT

5 Conclusions and Future Work

In this paper we present iStarJSON a light-weight technical contribution supporting i*models in JSON. Our proposal, allows to represent iStarML models into JSON, also we have provided a set of RESTful services for transforming iStarJSON files into a variety of output formats (e.g., PDF, SVG, PNG). The iStarJSON approach supports interoperability and flexibility of i* models, similar to iStarML, but with the advantages of JSON. Table 1 shows a comparison of XML and JSON.

Table 1: Comparison of XML & JSON

Criteria	XML	JSON
Undestandability	Machine readable	Machine & human readable
File size	Min. number of lines for object=3	Min number of lines for object:1
Mapping	Tree structure	Graph structure
JavaScript	Not subset	subset of JS

[8] `https://github.com/UPC-gessi-oscar-franco/iStarJson`

[9] http://www.graphviz.org/

Next steps on iStarJSON include extending the functionality to support PLATEOSS a framework for OSS ecosystems (OSSECOs) analysis. We plan to use i* for modelling OSSECOs and iStarJSON graph representation for enabling the power of network analysis statistics and reasoning tools of visual analytics (e.g., identify main business goals automatically, analyze quality of dependences between actors, calculate goals centrality measures).

ACKNOWLEDGMENTS

This work is a result of the EOSSAC project, founded by the Ministry of Economy and Competitiveness of the Spanish government (TIN2013-44641-P).

References

1. Yu, E.: Modelling strategic relationships for process reengineering. MIT Press (2011)
2. Cares, C., Franch, X.: A Metamodelling Approach for i* Model Translations. In: Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 337–351
3. Cares, C., Franch, X., Perini, A., Susi, A.: Towards interoperability of i* models using iStarML. Computer Standards & Interfaces **33** (2011) 69 – 79 Special Issue: Secure Semantic Web.
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems **8** (2004) 203–236
5. Cares, C., Franch, X.: iStarML: principles and implications. In: iStar 2011: proceedings of the 5th International i* workshop: 29-30th August, 2011, Trento, Italy, CEUR Workshop Proceedings (2011) 8–13
6. Cares, C., Franch, X., Colomer, D., López, L.: Tool interoperability using istarml. In: iStar 2011: proceedings of the 5th International i* workshop: 29-30th August, 2011, Trento, Italy, CEUR Workshop Proceedings (2011) 166–168
7. Goncalves, A.: XML and JSon Processing. In: Beginning Java EE 7. Apress, Berkeley, CA (2013) 387–416
8. Lee, D.: JXON: an architecture for schema and annotation driven json/xml bidirectional transformations. In: Proceedings of Balisage: The Markup Conference. (2011)
9. Gerhart, M., Bayer, J., Höfner, J.M., Boger, M.: Approach to define highly scalable meta-models based on json (2015)
10. Robie, J.: XQuery, XSLT and JSON adapting the XML stack for a world of XML, HTML, JSON and JavaScript. In: proceedings of the Balisage: The Markup Conference, Mobntreal, Canada, CEUR Workshop Proceedings (2012) 8–13
11. Designing Software Ecosystems: How Can Modeling Techniques Help? In: Proceedings of 16th BPMDS 2015 and 20th a EMMSAD 2015, Springer International Publishing (2015)