Event-Driven Design Approach to the QML Wrapper for SmartSlog Tool

Nikolai Lebedev
Petrozavodsk State University
Petrozavosk, Russia
lebedev@cs.karelia.ru

***Abstract*—SmartSlog Double API library is a tool for a knowledge processors (KP) and KP programming interfaces development. QML is declarative language based on JavaScript and suitable for cross-platrofm mobile application development. With the use of these two instruments it may be possible to implement QML wrapper for SmartSlog library to develop KP for different platforms. Moreover, such wrapper may be extended with some features providing implicit subscription control and therefore data-binding between different SIBs and KP. In this way such architecture will be based on different kinds of events. This event-driven approach may simplify the developer's work saving him from controlling different kinds of subscriptions, queries and connections. The purpose is to provide KPI wrapper for QML developers. The design of wrapper proposed in this paper may be used for cross-platform mobile and web development written in QML and JavaScript because these tools are common for a lot of platforms.**

I. Introduction

Nowadays, the most popular application platforms are mobile and web ones. Both can use JavaScript and its extensions for development. For example, JavaScript application may be compiled for mobile platforms using some specific frameworks such as PhoneGap or Cordova [1]. QML is declarative language based on JavaScript. Now it may be successfully applied for fast cross-platform mobile development with a preferable application performance. Moreover it's common developer's tool for some mobile operation systems, e.g. Sailfish OS, Android OS and others.

Internet of Things concept and SmartSpaces technology can be used in a lot of cases [2], [3]. According to SmartSpace approach, applications are represented with different distributed modules called knowledge processors (KP) and shared storage or semantic information broker (SIB). Single application can contain either one or several KPs. In Petrozavodsk State University the SmartSlog Double API library was developed [6]. It allows to create both KPs and KP programming interface using high-level and low level APIs.

SmartSlog library is written in ANSI C. It's suitable for most of platforms. However, often it is supposed to run KP on some kind of mobile devices. In this way there is an issue of integrating native ANSI C code into specific target environment. Sometimes such integration may cause a lot of difficulties related to data transfering from one programming language to another, complicated structure of applications or hard compilation process for different environements. The application updating process is complicated because a developer needs to keep all SmartSlog dependecies within his application.

As QML and Qt Quick is used for cross-platform application development it is possible to use them to create KP for several mobile devices using the same codebase. In this case QML and SmartSlog may be combined in a single module. The proposed module can be extended with some features that allows use some KP functions like subscriptions [8], SPARQL queries and others implicitely. In this way the event-driven [7] framework for SmartSpace can be introduced. Actually the simpliest implementation of QML wrapper may be integrated into any application from scratch. However the goal of this work is to implement event-driven QML wrapper to free developers from routine work with data transfering, ontology converting and subsctiptions control tasks.

In this paper the detailed design approach and some suggested issues of QML SmartSlog API implementation will be described. The data exchange formats and ontologies usage is examined too. The proposed component has modular structure, so all modules are described severally. In section II the high-level design of the proposed QML wrapper is presented. Section III exposes the managers layer that is responsible for connection and subscription control. Section IV contains description of the objects layer that represented by ontology parser, entity interfaces and QML models definitions.

II. Wrapper design

The proposed wrapper has a layer and modular design as it is shown on the Fig. 1. The first and higher level is API for QML developers. It's represented with object types, their properties and methods. The second layer is a core of proposed component. This layer consists of different mamgers that responsible for subscription, connection and ontology parsing control. The third layer is SmartSlog library that provides a back-end of proposed component. The wrapper uses low-level SmartSlog API that allows working with triples and triple queries.

The proposed wrapper is designed according to event-driven architecture. The main concept is to provide application work with different semantic information brokers (SIB). For example, when the triple or the individual is created, it is bound to SIB end-point automatically. The same situation appears when we fetch some data from SmartSpace. If triple updates either in SIB or in application, both sides get changes implicitly for a developer. This approach gives an opportunity to push triples to different SmartSpaces and support their actual state.

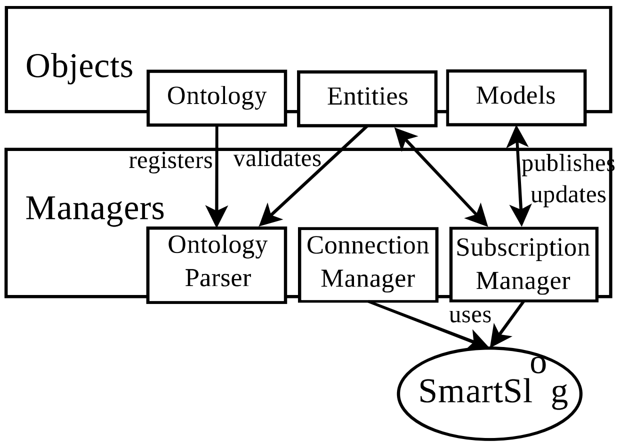

Fig. 1. High-level design of QML Wrapper for SmatSlog Tool

III. Managers layer

A. Connection Manager

SmartSlog Double API library provides an oportunity to create several nodes or end-points for SmartSpace connections. It's a flexible solution and it's supposed to be implemented in the QML wrapper, however with some extensions. Every node is presented by independent object in QML file that sets a connection configuration.

Connection Manager gives an opportunity to define SmarSpace Nodes and Sessions at high-level. Every node has methods to join or leave it on demand but in most cases this functions are called implicitely with the manager. It helps to organize optimal connections structure and handle all possible issues more carefully.

The example of the "Node" declaration is shown in the listing 1

Listing 1. Node Declaration Example

```
Node{
  id: "nodeId"
  address: "smartSpaceAddress",
  name: "smartSpaceName"
  port: "smartSpacePort"
  session: "sessionId"(optional)
}
```

B. Subscription Manager

One of the general features of SmartSlog Tool is a subscription for entity changes watching. Using ANSI C developer should define subscrition query and handlers to initialize the subscription process. The typical problem is to pass data from one module to another using such kind of handlers. QML application is suggested to be modular, therefore the difficulties with standart subscription definititions take place.

There is the subscription manager that is responsible for all subscribing processes during the application work. When a developer defines new individual or triple, it's bound to chosen node automatically. It's needed for notification process providing and to incapsulate subscription creating from QML. In point of fact, this manager is the main module of the proposed wrapper. It implements design pattern "Mediator" to keep map of all "entity-node" bindings. When entity is declared and checked with Ontology Parser Subscription Manager publish it to SmartSpace followed by a subscription creating. Further, when triples in SIB are changed, Subscription Manager finds corresponding entity and changes their property directly. If triple is bound to several nodes, in other words - to several SmartSpaces, Subscription Manager does all the same for different SIBs.

There is an issue of subscription intersection when we create several copies of the same individual. It may trigger intensive memory usage when a lot of suscriptions will be run in a background. So these issues may affect to a target mobile device energy saving and an application performance. The manager analyses subscriptions to similar entities and combines them into one saving memory and increasing performance.

IV. Objects layer

This layer is an API for developers. It consists of set of objects and their methods to provide working with SmartSpace using QML. The objects layer is represented by different types of entities:

- Individual
- Triple
- Subscription Model

Individuals may be represented like a nested structure of objects linked with properties. The main structure of data representation is SmartSpace is triple that consists of a subject, a predicate and an object. This is a common way of data exchange in SmartSlog low API. Triples may be combined into more complex structures, e.g. individuals with properties, using ontologies.

It's not convinient for QML Developer work with triples directly and more suitable representation model of entities should be introduced. As the most general data format in JavaScript is JSON, it will be used for Ontology Representation in QML API. Ontology is a set of classes, individuals and properties, in other workds it's more high-level way for data view than triples. Ontology may be presented in JSON-LD format [4]. The proposed wrapper is followed by JSON Ontology Generator Tool that allows to create JSON ontology from regular RDF format [5].

The purpose of JSON Ontology Generator is to introduce a map of ontology entities to JSON fields. The design of such map is shown in the listing 2.

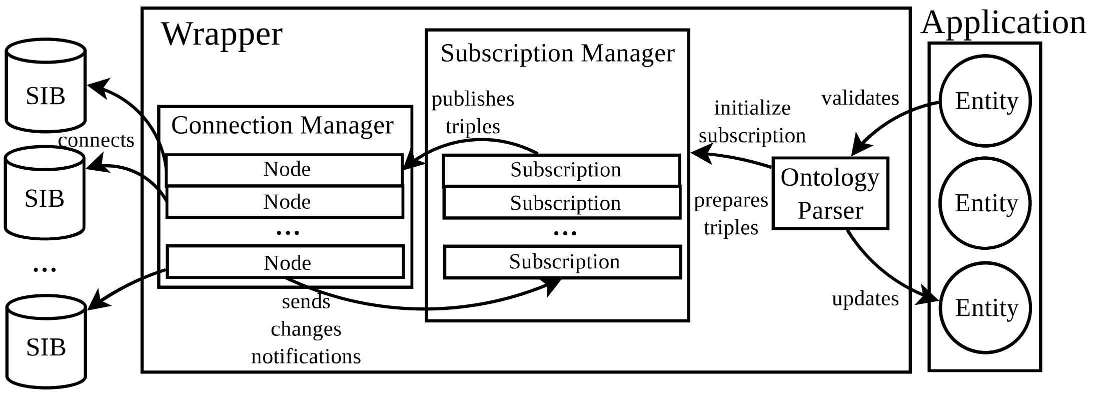

Fig. 2. The Wrapper Workflow

Listing 2. JSON Ontology Structure
```
Individual{
 "id": "objectId"
 "@type": "entityType",
 "name": "entityName"
 "objectProperties": [
  "propertyOne": {
   "name": "propertyName",
   "relatedClass": "className"
  }
 ],
 "dataProperties": [
  "propertyOne",
  "propertyTwo",
  "propertyN"
 ]
}
```

This object structure will be used as a prototype for strict SmartSpace entities declaration in QML. During implementing KP, developer has an opportunity to create both complex individuals and simple triples. Then all these objects will be transformed to clearable for SmartSpace format based on triples in Ontology Parser Model. Otherwise, this parser is supposed to convert data from triples to individuals according to given ontology.

The primitive triple declaration doesn't require any ontology, so it is defined as it is shown in the listing 3

Listing 3. Triple Declaration Example
```
Triple {
 id: "objectId"
 subject: "subjectString",
 predicate: "predicateString",
 object: "objectString"
}
```

Both individuals and triples support following set of methods:

- *bindToNode(node)* — to set data binding with concrete SmartSpace endpoint;
- *unbindFromNode(node)* — to unset data binding with concrete SmartSpace endpoint;
- *getProperty(property)* — to get concrete property value;
- *setProperty(property, value)* — to change concrete property value;

All objects have "onUpdate" callbacks to provide specific actions running on object updating in SmartSpace or in Application (e.g. by another module).

The default node and property values may be set in entity properties during the initialization. Entities may be combined into collections and subscription may be called on all elenements of collection. When entity is beeing unbind from all nodes, it becomes "detached". Detached entities are supposed to be stored locally and no their changes are sent to SmartSpace in this way.

Sometimes it may be convenient to bind entities to nodes from "Node" side. So the "Node" objects support following set of methods:

- *bindEntity* — to set data binding with concrete entity or collection of entities;
- *unbindEntity* — to unset data binding with concrete entity or collection of entities;

The workflow of the wrapper during object creation is shown on the Fig. 2.

A. Data models

In QML there are some data models for providing datasets for different structures such as lists, tables, grids, etc. The views and delegates are responsible for visualizaion and preparing data, but it's necessary to implement models for data fetching from SmartSpace. The proposed wrapper introduces subscription model for representing of such data. The example of the subscription model declaration is shown on the Fig. 4

Listing 4. Subscription Model
```
SubscriptionModel{
 id: "objectId"
 node: "nodeId",
 query: "queryId"
}
```

Such model is based on specific parameter that represent the query for data loading from SmartSpace. The Query may be passed with the use of a specific QML Query Object introduced with the describing wrapper. The Query Object contains several fields that define a type of query, a query body, “onUpdate” callback. The supported types are the same like in SmartSlog: triple template, ontology class or property (entity template) or SPARQL. Examples of query objects declaration are shown on the Fig. 5.

Listing 5. Query Object Examples

```
SPARQLQuery{
 id: "objectId"
 text: "queryRow"
}

TripleQuery{
  id: "objectId"
  subject: "subject"
  predicate: "predicate"
  object: "object"
}

OntologyQuery{
  id: "objectId"
  EntityTemplate:{
    class: "Class Name" (optional)
    uri: "Uri" (optional)
    properties: [
      OntologyProperty {
        name: "value"
      }
    ] (optional)
  }
}
```

V. Conclusion

In this paper the design aproach and possible implementation issues of QML wrapper for SmartSlog library were described. This wrapper is supposed to be full-fledged event-driven framework for development SmartSpace KPs for mobile platforms. The major features of proposed wrapper are an automatic data-binding between QML application and SmartSpace, an intelligent subscription control and JSON as data representation format.

The described techniques may be implemented not only for QML, but for JavaScript application either. In this way, it may have to drop SmartSlog and implement SSAP protocol support, because browsers cannot run ANCI C code. But on the other hand server-side JavaScript tools like Node.js allow to integrate C/C++ code into custom modules. So the event-driven wrapper for SmartSlog may be used in different KPs for mobile and web platforms that written with usual for these environments tools.

Acknowledgment

This research is financially supported by the Ministry of Education and Science of the Russian Federation within project # 14.574.21.0060 (RFMEFI57414X0060) of Federal Target Program “Research and development on priority directions of scientific-technological complex of Russia for 2014–2020”.

The study is financially supported by Russian Foundation for Basic Research 16-07-01289.

References

[1] A. Charland, B. LeRouxMobile “Application Development: Web vs. Native”, in *Communications of the ACM No. 5*, Vol. 54, May 2011 pp. 49-53

[2] D. G. Korzun, S. I. Balandin, A. V. Gurtov “Deployment of Smart Spaces in Internet of Things: Overview of the design challenges”, in *Internet of Things, Smart Spaces, and Next Generation Networking*, Springer Berlin Heidelberg, 2013. pp. 4859.

[3] D. G. Korzun, S. I. Balandin, V. Luukkala, P. Liuha, A. V. Gurtov. ”Overview of Smart-M3 principles for application development.” *Proc. Congress on Information Systems and Technologies (IS&IT11), Conf. Artificial Intelligence and Systems (AIS11)*, Vol. 4, 2011.

[4] “JSON-LD 1.0 A JSON-based Serialization for Linked Data” *W3C Recommendation*, 16 January 2014

[5] O. Lassila, R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation*, February 1999.

[6] D. G. Korzun, A. A. Lomov, P. I. Vanag, S. I. Balandin, J. Honkola “Multilingual Ontology Library Generator for Smart-M3 Information Sharing Platform”, in *International Journal on Advances in Intelligent Systems No. 3 & 4*, Vol. 4, 2011 pp. 68-81.

[7] Brenda M. Michelson “Event-Driven Architecture Overview”, Patricia Seybold Group Boston, 2 February 2006

[8] A. A. Lomov, D. G. Korzun, “Subscription operation in Smart-M3”. *Proc. 10th Conf. of Open Innovations Association FRUCT and 2nd FinnishRussian Mobile Linux Summit*, 2011, pp. 83-94.