A mediator system for querying heterogeneous data in robotic applications

Rubanraj Ravichandran

Master Thesis Proposal
Master of Autonomous Systems

Department of Computer Science
University of Applied Sciences Bonn-Rhein-Sieg

Supervisors:
Prof. Dr. Erwin Prassler
Prof. Dr Manfred Kaul
Advisors:
Nico Huebel
Sebastian Blumenthal

December 5, 2018

1 Introduction

Robots generate large amount of data from different types of sensors attached to it and also from its hardware components. In our previous research work [7], we have conducted an extensive qualitative and quantitative analysis to find better databases and architectures that effectively store these data and consume it for further operations. Results from our previous work shows that, a single database is not suitable for every robotic scenarios. For example, in terms handling large BLOB data, mongoDB stored them faster but reading the data was slower compared to couchdb [7]. Also, to complete a given task robots depends on multiple source of information from internal sensors, as well as external sources for example world model, kinematic model, etc..

Adoption of multiple databases for robotic applications requires unique way of mediation to view multiple databases as a single federated database. Mediator approach helps to integrate data from different sources and produce a single result back to robots. Mediator abstracts the information of how data is being stored in different data sources from robot, and allows robotic applications stream data to mediator independent of databases used in the back-end.

To Map the data generated by robots with multiple databases, mediator system requires a proper data model predefined in the context of robotic applications. Modeling robot generated data helps to generalize the structure of data and defining relations between different objects in the robot systems. If we have a well defined robotic data models, then mediator will get the ability to mutate or query data from different data sources. Also it is important that, these data models should be extended to any robotic use-cases.

As mentioned in these papers [1, 5, 2, 4, 4, 8], mediators are being used to integrate data from different data sources and few architectures supports single data model (e.g SQL), and others supports for different data models (e.g SQL,NoSQL, document store, etc..). Also, they are differ from query languages, ease of implementation, components used in their architecture. This project mainly focus on defining semantic based models for sensor data to make it more inter-operable with other systems or even in multi robot systems, and implementing a mediator system which act as a middle-ware

between robots and databases.

2 State of the art analysis

2.1 Semantic Data model

Su et al. [9] highlights the interoperability issues in IoT sensor data and also says that these data should be useful for multiple applications rather than dependent on specific domain. To make the machines interpret the meaning of sensor data, author suggested to use Semantic Web technologies such as Resource Description Framework (RDF). Even though SenML is an evolving technique to model to sensor data, but it is lacking reasoning capabilities and interoperability with other devices. To overcome the issues in SenML generic model, author proposes a technique to represent IoT sensors as a Knowledge Based Systems by transforming SenML to Resource Description Framework.

As an advantage, the transformed data can be analyzed and helpful to take more meaningful actions. SenML is specially designed for resource constrained devices, hence additional information to contextually understand the data has be not excluded intentionally. Each entry of SenML data should have the sensor parameter name and other attributes such as time, value, etc. Also it supports custom attribute called Resource Type (rt) to let the users add their own attributes and this allow users to include contextual information.

With the help of transformation we could adopt RDF structure to model robot generated sensor data, But Charpenay et al. [3] points out that RDF data structure is not suitable for resource constrained devices like micro-controllers and also analyzed the issues in RDF such as verbosity and complexity in processing knowledge. To overcome this issues, author carried out an extensive analysis between RDF and JSON-LD structures. JSON-LD was published by W3C, and it serves as an alternative for RDF. Using JSON-LD one can represent the context for the data which is more important in robotics field such that other robots can understand the data based on context.

Results shows that JSON-LD compaction coupled with EXI4-JSON or CBOR outperforms state-of-the-art (HDT) with **50 - 60 %** compaction

ratios.

2.2 Mediator architectures

Fahl et al. [5] proposed an active mediator architecture to gather in formation from different knowledge base and combine them to a single response. AMOS[1] architecture uses Object-Oriented approach to define declarative queries. This distributed architecture involves multiple mediator modules to work collaboratively to collect the required piece of information and produce final result. Primary components of AMOS architecture are,

- Integrator - Gather data from multiple data sources that have different data representations.
- Monitor - Monitor service always watch for any data changes and notifies the mediators. This is helpful in the case where system needs an active updates to change its current task.
- Domain models represents the models related to application which helps to access data easier from any database through a query language.
- Locators helps to locate mediators in the network.

Integrator module is built with two internal components called IAMOS[2] and TAMOS[3]. First Integration AMOS parse the query and send individual requests to Translational AMOS modules which are responsible for heterogeneous data source. Then,all TAMOS modules return the individual results to IAMOS for integrating all the results. To query multi databases from IAMOS, IAMOS servers are mapped with TAMOS servers w+ith the help of Object-Oriented query language.

Ahmed et al. [1] developed a heterogeneous multi-database system called Pegasus that supports multiple heterogeneous database systems with various data bases models, query languages and services. Pegasus predefines its domain data models based on sobject oriented approach and also supports programming capabilities. These objects are created and mapped with

[1] Active Mediators Object System
[2] Integration Active Mediators Object System
[3] Translation Active Mediators Object System

the types and functions with the help of HOSQL[4] statements. HOSQL is a declarative object oriented query language which is used by Pegasus to manipulate data from multiple data sources.

Pegasus system supports two types of data sources, local and native data sources. Whenever a new data source joins Pegasus system, schema integrator module imports schema from data source and update its root schema with the new schema types. The final integrated schema shows the complete blueprint of the different data sources participates in the data integration. Pegasus system work-flow is comparatively similar to AMOS architecture, but they use different query language and data modeling strategies.

Chawathe et al. [4] developed project Tsimmiss extract information from any kind of data source and translates them to a meaningful common object. Unlike AMOS and Pegasus, Tsimmiss follows a straight forward approach to define the data model which is a self-describing object model. Each object must contain a label, type and value itself. Label can be used by the system to understand the meaning of the value and type shows the observed value type. Objects can be nested together to form a set of objects.

Tsimmiss tool offers a unique query language called OEM-QL and this language follows the SQL query language pattern to fetch the data from mediators. Mediators resolves the query and send separate requests to respective data sources to retrieve the information and merge them together to give a single response back to user. During data integration process, Tsimmiss removes possible duplicates to avoid redundancy in the response. Also, Tsimmiss bundles a default browser tool to query data using OEM-QL language.

In the articles discussed above, mediators are targeted to extract information from different data sources that could be different databases or data from file-system. But Rufus system proposed by Shoens et al. [8] focus only on semi structured data stored in file system for example documents, objects, programming files, mail, binary files, images etc. Rufus system classifier automatically classifies the type of file and apply a scanning mechanism on those files to extract the required information and transform them to the appropriate data model which is understandable by Rufus system. Rufus can

[4]Heterogeneous Object Structured Query Language

classify 34 different classes of files. In terms of query language, Rufus can apply simple object predicates and finding text from the extracted information from the documents or files.

Papakonstantinou et al. [6] proposes a Mediator Specification Language that helps the mediator to understand the schema and integrates the data from unstructured or semistructured source. MSL overcomes the major problems in existing mediator systems for example,

- Schema domain mismatch
- Schematic discrepancy
- Schematic evaluation
- Structure irregularities

During translation of original information from different sources to a single object it should be important that, all data sources should have the required attribute and the name of the attribute should be same. Otherwise, mediator system will not be able to process the information to a single answer. External predicates and Creation of the Virtual Objects in MSL solves the problems mentioned above.

Arens et al. [2] built a mediator which is flexible to map domain level query different data-sources and efficient to plan the query execution to reduce the overall execution time. Information source models provides relations between the super class and subclasses, and also the mapping between the domain models and information from heterogeneous sources. SIMS uses Loom as a representational language to make objects and relationship between them. SIMS supports parallel query access plan that makes the mediator to access information independent of data sources and the user will get the final answer as quick as possible.

Many mediator systems developed in the past to support integrating heterogeneous information from different data sources. All of them built with different architectures, query language, and execution optimization. In our mediator approach we focus mainly on,

- How different type of robot generated data will be stored in multiple data sources?

- A unique context based data model to represent the components attached with each robot and data generated by them.
- Semantic query language to communicate with mediator. Unlike traditional query languages we would like to attempt new way of querying data, for example Graphql.
- GUI tool to visualize and analyze the robot generated data in a meaningful way.

3 Problem Formulation

Our previous work results reveals not all databases reacts in a similar fashion for different heterogeneous data from robotic applications. Also, there are no concrete data models has been defined in the context of robotic applications. For example, black box designed for ROPOD project uses mongoDB to store data from different sources such as Ethercat, Zyre, ZMQ, and ROS topic. The data is being transformed in to a simple flatten json document to store the values. These documents are stored under a single collection which is created for each ROS topic or other sources. Each document holds only the information of value generated by the sensors or application itself, but these values are not useful without additional details for example, who created the data, if it is a robot then what type of robot generated this data from which location?, then in what context other system should interpret this data.

For example in the black box, geometry_msgs/Pose ROS topic will be flattened to a simple json document which have the following format.

```
double timestamp
double position/x
double position/y
double position/z
double orientation/x
double orientation/y
double orientation/z
double orientation/w
```

In the above format, 'position/x' is a key and the value will be attached with it. Now only with position x,y,z and orientation x,y,z,w, other system which consumes this data would not be able to say who generated this data or at which location this data is being generated and if the other system is doing mathematical calculation, then this data is missing it own context such as unit, dimensions, etc.

In terms of supporting multiple databases setup for robots, there is no systematic approach to store and retrieve data from external sources and, a well defined data model that can map components and sensors to a robot and also with the world model. In this case, there is no mediator has been developed before to connect between robots and databases with different data model.

4 Approach

To find the best data models for robotic applications and build a scalable mediator, we begin with SOA analysis to find out approaches that has been followed through before for similar data integration applications. After defining the data model, list of recent querying techniques will be collected and reviewed based on the features and possibility of adopting them with the mediator as a base. For review we would like to consider current well known querying techniques such as Graphql, and Falcor. At first, our mediator will support only the databases which are selected based on the results from our previous research work [7] and other data-sources used by ROPOD such as OpenStreetMap. Then, schema's will be defined to map the data being generated by the robot and the data sources. To reduce the complexity of identifying appropriate data-sources by the robot, in our architecture mediator will dynamically choose the data-source respective to the type of data that robots want to store and retrieve. Still the configuration will be adjustable according to the scenarios. Finally to visualize the integrated data from mediator in a meaningful way, a GUI application will be developed based on Node.js stack.

5 Workplan

This work-plan shows the major decomposition of the work-packages, start time and end time. This project starts on October 15^{th}, 2018 for 6 months duration and ends on April 16^{th},2019.

5.1 Work packages

Task Name	Start Date	End Date	Duration
Literature survey (Work package 1)	**10/15/18**	**11/23/18**	**40d**
Analysis of SOA approaches	10/15/18	11/03/18	20d
Identification of mediator architectures & data models	11/04/18	11/08/18	5d
Summarizing deficits in SOA	11/09/18	11/13/18	5d
Identifying usecases	11/14/18	11/23/18	10d
Designing mediator architecture (Work package 2)	**11/24/18**	**12/23/18**	**30d**
Data models creation	11/24/18	12/03/18	10d
Identifying schema wrappers for chosen databases	12/04/18	12/13/18	10d
Identifying components in the mediator	12/14/18	12/23/18	10d
Implementation (Work package 3)	**12/24/18**	**02/21/19**	**60d**
Developing the components from the mediator architecture	12/24/18	01/12/19	20d
Developing schema wrappers to make communication between robots and databases via mediator	01/13/19	02/01/19	20d
Adopting existing FMEA tool to map logs from mediator	02/02/19	02/06/19	5d
GUI implementation to visualize to data from mediator	02/07/19	02/21/19	15d
Experimentation & Testing (Work package 4)	**02/22/19**	**03/23/19**	**30d**
Test the mediator with existing black box module	02/22/19	03/08/19	15d
Functional and reliability testing under different robotic scenarios	03/09/19	03/23/19	15d
Report(Work package 5)	**03/24/19**	**04/16/19**	**24d**
Report writing	03/24/19	04/07/19	15d
Revising report	04/08/19	04/12/19	5d
Finalizing report	04/13/19	04/16/19	4d

Figure 1: Work packages

5.2 Gantt Chart

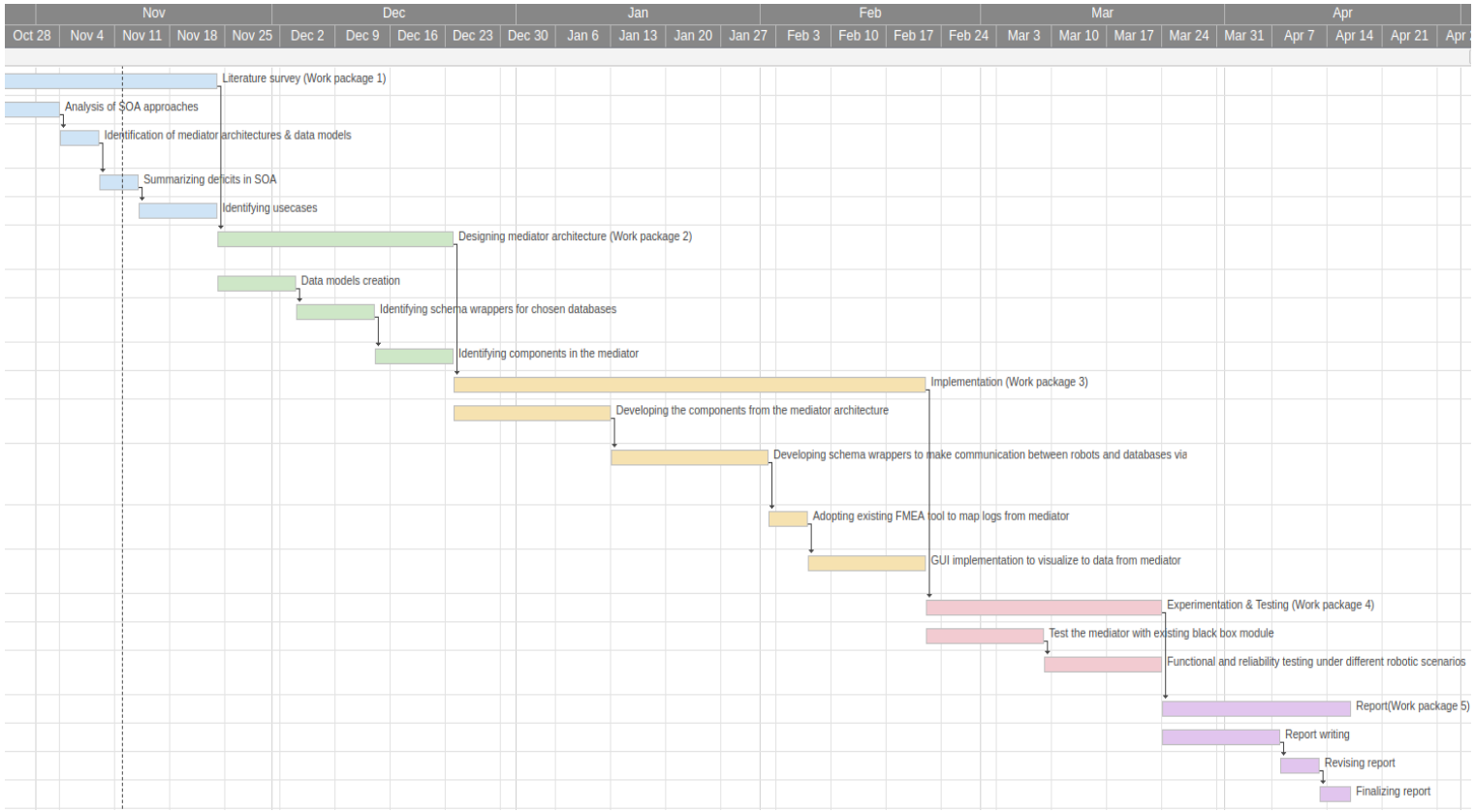

Figure 2: Gantt Chart

6 Deliverables

6.1 Minimum

- Report on state of the art analysis.
- Collection of existing mediator architectures which can be adopted to robotic applications.
- Data modeling in the context of robotic applications.
- Finding unique and semantic query language to communicate with mediator.

6.2 Expected

- Designing and implementing mediator system that supports storing and retrieving robot generated sensor data from different data sources dynamically.
- Suitable data model and relationships for multi robot use-cases.

6.3 Maximum

- Developing a GUI tool to interactively communicate with mediator and visualize the sensor data in a meaning way.

References

[1] Ahmed, R., DeSmedt, P., Du, W., Kent, W., Ketabchi, M. A., Litwin, W. A., Rafii, A. and Shan, M.-C. [1991], 'The pegasus heterogeneous multidatabase system', *Computer* **24**(12), 19–27.

[2] Arens, Y., Hsu, C.-N. and Knoblock, C. A. [n.d.], 'Query processing in the sims information mediator'.

[3] Charpenay, V., Käbisch, S. and Kosch, H. [2018], Towards a binary object notation for rdf, *in* 'European Semantic Web Conference', Springer, pp. 97–111.

[4] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J. [1994], 'The tsimmis project: Integration of heterogenous information sources'.

[5] Fahl, G., Risch, T. and Sköld, M. [1993], 'Amos-an architecture for active mediators'.

[6] Papakonstantinou, Y., Garcia-Molina, H. and Ullman, J. [n.d.], Medmaker: A mediation system based on declarative specifications, *in* 'Data Engineering, 1996. Proceedings of the Twelfth International Conference on', IEEE, pp. 132–141.

[7] Ravichandran, R., Huebel, N., Blumenthal, S. and Prassler, E. [2018], 'A workbench for quantitative comparison of databases in multi-robot applications'.

[8] Shoens, K., Luniewski, A., Schwarz, P., Stamos, J. and Thomas, J. [n.d.], The rufus system: Information organization for semi-structured data.

[9] Su, X., Zhang, H., Riekki, J., Keränen, A., Nurminen, J. K. and Du, L. [2014], 'Connecting iot sensors to knowledge-based systems by transforming senml to rdf', *Procedia Computer Science* **32**, 215–222.