Towards Agile Application Integration with M2M Platforms

Menghan Chen[1] and Beijun Shen[2]
[1,2]High Reliable Software Lab, School of Software
Shanghai Jiaotong University
Shanghai, China
[e-mail: {chenmenghan_1986, bjshen}@sjtu.edu.cn]

Received August 30, 2011; revised November 11, 2011; revised November 24, 2011; accepted December 26, 2011; published January 31, 2012

Abstract

M2M (Machine-to-Machine) Technology makes it possible to network all kinds of terminal devices and their corresponding enterprise applications. Therefore, several M2M platforms were developed in China in order to collect information from terminal devices dispersed all over the local places through 3G wireless network. However, when enterprise applications try to integrate with M2M platforms, they should be maintained and refactored to adapt the heterogeneous features and properties of M2M platforms. Moreover, syntactical and semantic unification for information sharing among applications and devices are still unsolved because of raw data transmission and the usage of distinguished business vocabularies. In this paper, we propose and develop an M2M Middleware to support agile application integration with M2M platform. This middleware imports the event engine and XML-based syntax to handle the syntactical unification, makes use of Ontology-based semantic mapping to solve the semantic unification and adopts WebService and ETL techniques to sustain multi-pattern interactive approach, in order to agilely make applications integrated with the M2M platform. Now, the M2M Middleware has been applied in the China Telecom M2M platform. The operation results show that applications will cost less time and workload when being integrated with M2M platform.

Keywords: M2M, M2M middleware, syntactical and semantic unification, ontology

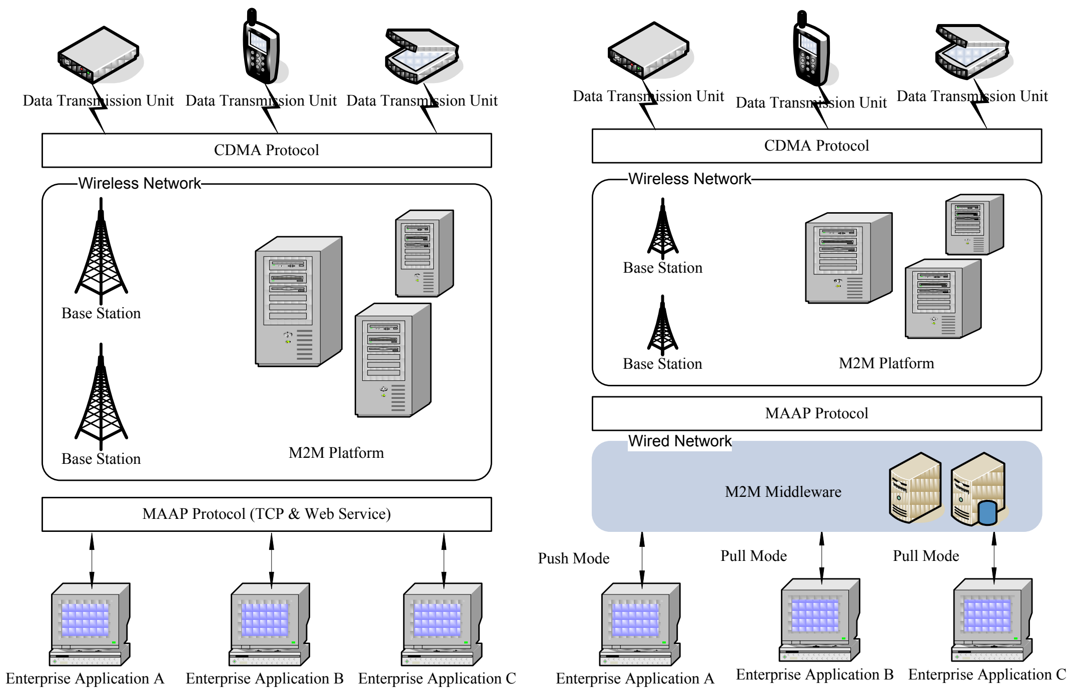
This research was supported by Project “Research on M2M Middleware for Enterprise Application’s Rapid Integration” of China Telecom and Schneider Sunshine Co.Ltd

DOI :10.3837/tiis.2012.01.005

1. Introduction

M2M (Machine-to-Machine) refers to technologies that allow both wireless and wired system to communicate with other devices of the same ability [1]. Nowadays, since quite a lot distinguished and heterogeneous appliances and terminal devices have been engaged in the M2M domain, M2M Platforms, as shown in **Fig. 1-(a)**, were developed and run by telecommunications in China to collect and dispatch raw data through various 3G wireless techs like CDMA (Code Division Multiple Access) or other wireless protocol. The applications catch and dispatch raw data up and down through the M2M platform and some specific internet protocols, for example, the MAAP Protocol put forward by China Telecom. With M2M platform being brought up, several advantages take place to overcome the weakness of the primitive single-established linking mode between the terminal device and the application server. For example, one-to-one communicating link established is hard to satisfy for other heterogeneous terminal devices to be in touch with. Also, details of terminal units are totally exposed to applications without any hardware abstraction which may bring about obstacle for a definite application to share different terminal units. The most troublesome factor lies in the large-scale increasing numbers of applications which are ready to connect with various devices.

Although M2M Platform mostly solves the problems above, there still exists a series of new challenges when enterprise applications try to integrate with the platform.



(a) without M2M Middleware **(b)** with M2M Middleware

Fig. 1. Application Integration with M2M Platform

Firstly, the maintaining and refactoring work of enterprise application is overwhelming and tedious though integrated interfaces like SDK should have been provided because of the heterogeneous properties of different application systems whatever the implementation language and operating environment are considered. Once enterprise applications would like to be involved in the platform, consuming time on modification of codes and even architecture is inevitable. Next, syntactical unification is still lack because of the factor that data noise, redundancy, unexplained segment meaning and unformatted interactive message are commonplace. Compared with applications in the domain of RFID (Radio Frequency Identification) Tech in Internet of Things, EPCGlobal [2] ALE (Application Level Event) -like syntactical standard is urgently needed to solve the problems above. Thirdly, information sharing in modern IoT is more significant than ever before but no technology has been applied in business vocabulary mapping to make data commonly known. For example, in the domain of automobile, speed (km/s) and velocity (m/s) are both the business vocabularies which interpret and depict "How fast the auto is going", but the value and dimension are hard to be shared if vocabularies are not semantically mapped. Last but not the least, multi-patterned interaction between enterprise applications and M2M Platform are defected, like both Push & Pull, and direct ETL (Extraction-Transformation-Loading), in the process of adapting distinguished data transmission requirement.

Corresponding to the challenges above, in this paper, we propose our scheme which based on a middleware, as shown in **Fig. 1-(b).** Overall, the interfaces to enterprise applications and M2M platforms which are based on Web Service and Message Middleware, the syntactical analysis due to event engine, EPL (Event Processing Language) and XML (Extensible Markup Language) Specs, the semantic analysis combined with ontology theory and WordNet, and the database extraction and mapping are four major modules considered to constitute the middleware. After we researched and developed the M2M middleware, Schneider Sun Shine's Air Compressor System has been put into practice to verify the feasibility of our scheme.

The rest of the paper is organized as follows: Section 2 will introduce some related work. In section 3, we go deep into the architecture of M2M Middleware and its mechanism. In section 4 & 5, how to implement data syntactical and semantic unification will be exhibited. Section 6 will explain our patterns in data interaction between enterprise applications and M2M middleware. Section 7 will interpret our implementations and related experiments in practice. Last, we conclude and look into the future.

2. Related Work

Among several researches on middlewares in IoT, though, most of which are based on the technology of RFID instead of M2M, offer valuable reference to our research on M2M Middleware. [3] has exhibited a complete implementation of EPCGlobal ALE Module which contains reading tags, writing tags, access control and logical control APIs, but less semantic work has been done. [4] considers a novel approach to provide an AOP (Aspect Oriented Programming) based framework of RFID ALE middleware. However, this framework is mainly aimed to RFID domain and there is not a specific syntactic unification mechanism in that enterprise applications have to maintain very details of their heterogeneous devices. [5] puts forward an integral RFID Middleware combined with HAL (Hardware Abstraction Layer), DF (Data Filtering) and AI (Application Interface), the most important perspective of which lies in the binding of RFID and antenna to be pair of <R, A> in order to constitute events through logical combination based on channel, helping to subscribe solutions to complex events. Compared with solutions to complex events, to utilize the enterprise-leveled event engines are more secure and have high extensibility. What's more, event engines like Esper providing SQL-like inquiring syntax is easy for programmers to quick understand. [6] went deep into the ALE events considering it necessary to classify events into simple and complex ones in order to lead to high flexibility and responsiveness. [7] proposes that two ways of syntactical solution including ALE based and WinRFID-XSLT based mechanisms, both of which are compared in performance. Both [7][8] have put forward high performance ways to solve ALE process, nevertheless, they ignored how to semantically map the events to different applications in the same business domain for information sharing. The above are novel ideas in data filtering and processing, but they are against RFID. Moreover, there also have been several researches on M2M applications. [8] takes advantage of the M2M techs to implement an automation of facility management process, which departs the business service from the basic communication one and put forward a distributed data management proposal. The most advantage is the CEP (Complex Event Processor) pulled in and the consistence with EPCGlobal Standard on ALE, but it is merely suitable for a definite monitoring system without any information share with others. [9] gives out an M2M proxy communication method, which establish a proxy server maintaining a synchronized message queue for poll mode. The performance of data transmission is satisfied but no flexible data interaction mode and syntactical rule for multiple applications to integrate with such a server. So, above all, we absorbed some merits and summarize our own M2M Middleware.

3. Architecture of M2M Middleware

According to the challenges proposed above, we suggest and exhibit the M2M Middleware to face with them.

The M2M Middleware is generally required to satisfy the following requirements:

A) Common Interfaces to M2M Platforms for basic raw data up and down.

As the M2M middleware will be established over the M2M Platforms, the MAAP protocol have to be encapsulated by a series of common interfaces which are enabled to make the basic raw data interaction between M2M platforms and our middleware.

B) Platform-Independent and Multi-Pattern Interfaces to Applications for flexible integration.

Because of the tendency of increasing number of heterogeneous enterprise applications, whatever in the aspect of business or implementation are distinguished when trying to be integrated with the M2M platform, platform independent interfaces will flexibly for applications to access our middleware whatever CPP\JAVA\.NET is used to implement. Moreover, multi-pattern interfaces like pull and push etc are efficient in solving different data interaction mode.

C) Syntax Mechanism of logic subscription and raw data processing for encapsulation of complex details of business logics and terminal devices.

The business related logic becomes more and more complicated even the enterprise have to get the very specific knowledge and details of the syntax of their terminal devices. So syntax subscription and raw data encapsulation will lead to a better syntactical unification when terminal devices' physical details are departed from the applications' business logics.

D) Semantic Map to unify core business vocabularies for the sake of easy information sharing.

Business Events should be shared with different applications who belong to the same business domain if necessary. However, the vocabularies adopted by different applications are various though the same meaning is summed up. So, to semantically map the Core Business Vocabularies (CBV) as well as to unify their dimensions, are essential for information sharing.

The architecture of M2M Middleware, as shown in **Fig. 2**, can generally be departed into several important modules including:

1. MAAP Protocol Interface Module. This module is responsible for the job of communication with M2M platform. MAAP Protocol is implemented by China Telecom which provides a unified protocol to communicate with applications. However, MAAP Protocol has exposed too much communicative details like TCP sockets, SOAP and heartbeat checking etc, which are great burdens for applications to implement when the integration occurs. Moreover, MAAP SDK has not been released. So, this module generally encapsulates the MAAP and exposes several callback functions for our middleware to implement the business logic.

2. Raw Data Syntactical Analysis Module. This module's duty includes parsing and filtering the raw data or binaries from M2M platform corresponding to the syntax cached and parsed in Syntax DB. Because of the factor that the raw data may contain invalid message and duplicated information, which also has not mapped to the definite meaningful segments, this module analyzes the syntax subscribed by domain experts who have full knowledge about the characteristics of terminal devices and finally transforms encapsulated and meaningful data into POJO [10] events.

3. Syntax Subscription & XML-POJO Mapping. This module generally makes interaction with the Syntax DB for syntax parsing and mapping the XML-formatted syntax to POJO classes which is ready to be dynamically loaded in order to instantiate the POJO events. The syntax subscription is responsible for parsing the syntactical meaning in the XML-formatted syntax, for example, interpreting the address ranges of segments according to the specification of terminal devices. XML-POJO Mapping is a runtime mapping process that the POJO classes are dynamically compiled by POJO sources which are parsed from the syntax subscription and ready for instantiate to be POJO events.

4. Core Business Vocabulary Mapping. This module takes the task of data semantic unification through core business vocabulary mapping. Domain experts use the domain ontology model language like RDF [11] to model the domain ontologies and map relationship among ontologies [12], through which an automated mapping task will be processed to establish the mapping relationship of business vocabularies like "speed (km/h)" and "velocity (m/s)". One of another major duty of CBV Mapping is about to make the dimension unified for example, 3.6 times should be considered when "km/h" and "m/s" are mapped.

5. Esper Event Engine [13][14]. Esper Event Engine is a complex event process engine that accepts the logic rules subscribed by enterprise applications with the language EPL [15] and provides listener interface for customer to implement their own event processing, according to the events happened if the specific logic rules have been satisfied. Esper Event Engine can provide a slide window not only being able to embrace a specific span of time but being capable of involving a specific volume of events that have occurred.

6. Enterprise Application Interface. This interface is in charge of the communication to enterprise application which provides both Push and Pull patterns. Web Service based techniques and Message Middleware techs are utilized to shield the heterogeneity among different applications when implementing the Pull and Push.

7. Database ETL Tool. This tool additionally supplies a database extracting and loading mechanism for applications for the sake of large quantity of history data and analysis.

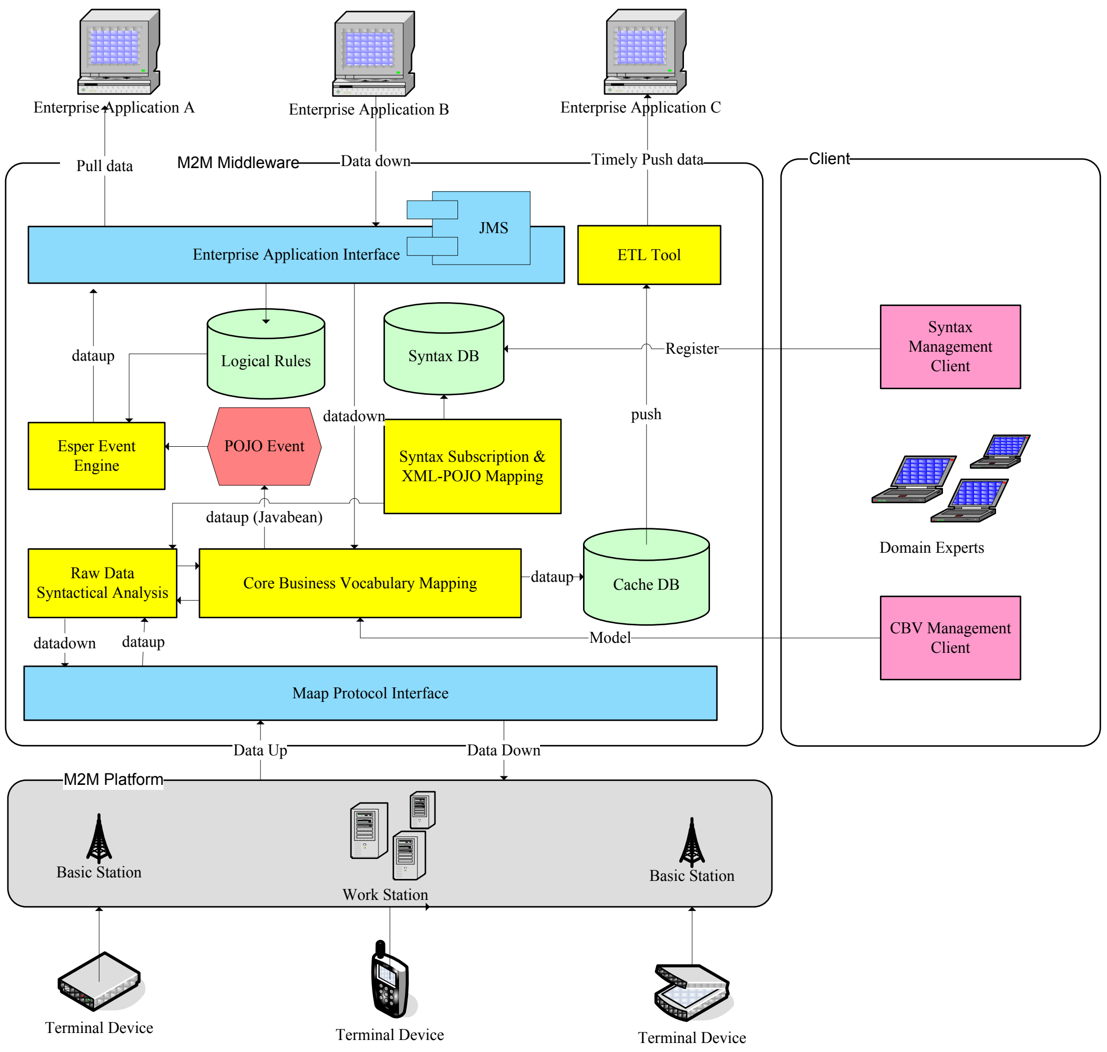


Fig. 2. The Architecture of M2M Middleware

4. Data Syntactical Unification

In our M2M Middleware, the Data Syntactical Unification, the Data Semantic Unification and the Multi-Pattern Data Interaction are the three pivot techniques which will be explained in the following three sections.

Data syntactical unification is considered as the technique including the data filtering and formatting, encapsulation according to specific and meaningful segments, business logics injection and processing. **Fig. 3** indicates how the syntactical unification works.

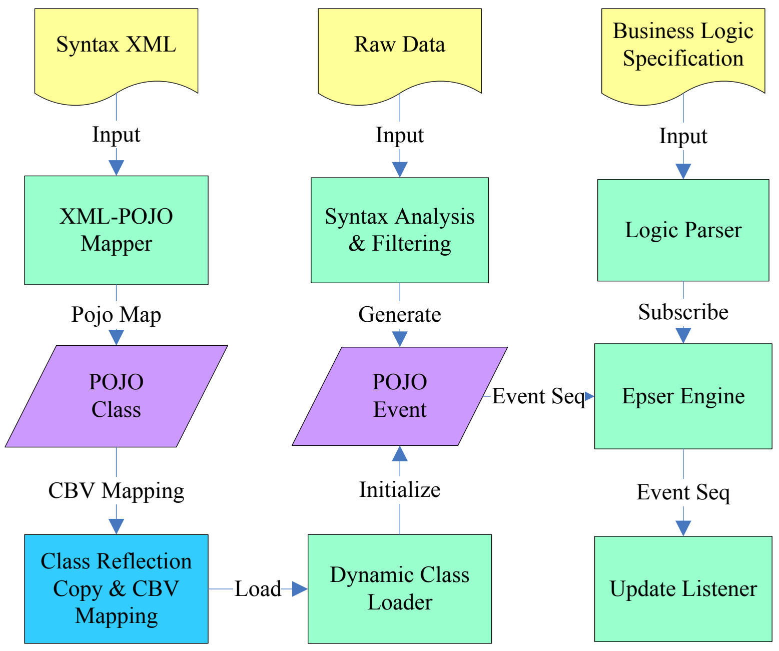
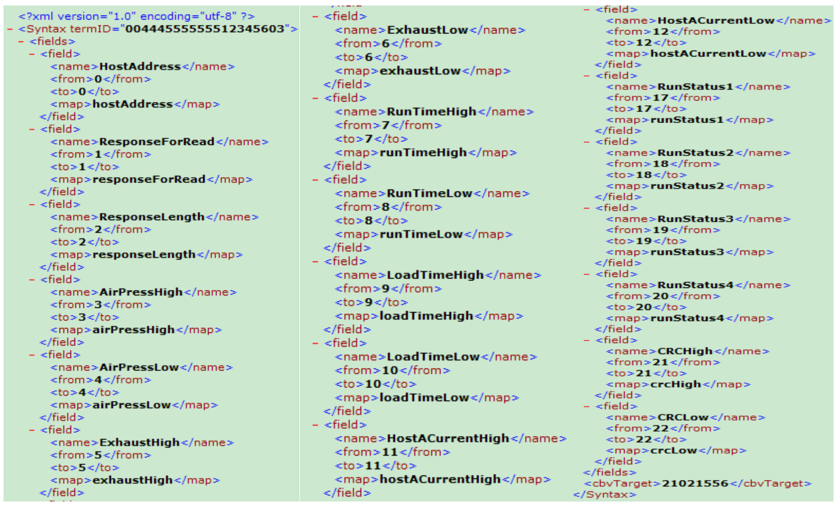


Fig. 3. The Work Flow of Syntactical Unification

Syntax XML specification, as shown in Fig.4 represents the interpretation containing the segments' range and meaning from device experts who have full knowledge of terminal specifications. XML-POJO Mapper accepts the specification and transfers it into a POJO class, after which a copy of POJO class will be generated and go through the CBV (Core Business Vocabularies) Mapping based on ontology by reflection mechanism if the definite mapping is required by applications. Dynamic Class Loader will load the POJO classes and initialize a POJO object which is called POJO event. Raw Data is dispatched to Syntax Analysis and Filter Module in order to split the data duplicated and then map definite data to its relative segment. After the events are generated, Esper Event Engine accepts the sent objects and triggers the update listener to finish the relevant event processing work corresponding to the logic rules subscribed by applications.



```
  <?xml version="1.0" encoding="utf-8" ?>
- <Syntax termID="00444555555512345603">
  - <fields>
    - <field>
        <name>HostAddress</name>
        <from>0</from>
        <to>0</to>
        <map>hostAddress</map>
      </field>
    - <field>
        <name>ResponseForRead</name>
        <from>1</from>
        <to>1</to>
        <map>responseForRead</map>
      </field>
    - <field>
        <name>ResponseLength</name>
        <from>2</from>
        <to>2</to>
        <map>responseLength</map>
      </field>
    - <field>
        <name>AirPressHigh</name>
        <from>3</from>
        <to>3</to>
        <map>airPressHigh</map>
      </field>
    - <field>
        <name>AirPressLow</name>
        <from>4</from>
        <to>4</to>
        <map>airPressLow</map>
      </field>
    - <field>
        <name>ExhaustHigh</name>
        <from>5</from>
        <to>5</to>
        <map>exhaustHigh</map>
      </field>
    - <field>
        <name>ExhaustLow</name>
        <from>6</from>
        <to>6</to>
        <map>exhaustLow</map>
      </field>
    - <field>
        <name>RunTimeHigh</name>
        <from>7</from>
        <to>7</to>
        <map>runTimeHigh</map>
      </field>
    - <field>
        <name>RunTimeLow</name>
        <from>8</from>
        <to>8</to>
        <map>runTimeLow</map>
      </field>
    - <field>
        <name>LoadTimeHigh</name>
        <from>9</from>
        <to>9</to>
        <map>loadTimeHigh</map>
      </field>
    - <field>
        <name>LoadTimeLow</name>
        <from>10</from>
        <to>10</to>
        <map>loadTimeLow</map>
      </field>
    - <field>
        <name>HostACurrentHigh</name>
        <from>11</from>
        <to>11</to>
        <map>hostACurrentHigh</map>
      </field>
    - <field>
        <name>HostACurrentLow</name>
        <from>12</from>
        <to>12</to>
        <map>hostACurrentLow</map>
      </field>
    - <field>
        <name>RunStatus1</name>
        <from>17</from>
        <to>17</to>
        <map>runStatus1</map>
      </field>
    - <field>
        <name>RunStatus2</name>
        <from>18</from>
        <to>18</to>
        <map>runStatus2</map>
      </field>
    - <field>
        <name>RunStatus3</name>
        <from>19</from>
        <to>19</to>
        <map>runStatus3</map>
      </field>
    - <field>
        <name>RunStatus4</name>
        <from>20</from>
        <to>20</to>
        <map>runStatus4</map>
      </field>
    - <field>
        <name>CRCHigh</name>
        <from>21</from>
        <to>21</to>
        <map>crcHigh</map>
      </field>
    - <field>
        <name>CRCLow</name>
        <from>22</from>
        <to>22</to>
        <map>crcLow</map>
      </field>
    </fields>
    <cbvTarget>21021556</cbvTarget>
  </Syntax>
```

Fig. 4. A Sample of XML Syntax

5. Data Semantic Unification

After syntactical unification, data semantic unification including CBV mapping and dimension transformation should be followed for the information sharing between multiple enterprise applications, as shown in **Fig. 5**. Our semantic mapping approach is based on ontology theory. Facing to the factor that it has difficulty in requiring a consensus domain business vocabulary model in the scope of domain if considered to be quite large, we put forward an ontology mapping algorithm [12] which establishes the mapping relationship among multiple ontology trees and their conception nodes from four levels considering literal, semantic, constraint and structural properties and characteristics.

The literal analysis is based on the comprehensive algorithms of string comparison. The literal analysis combined with the classic algorithm of Levenshtein Distance [16] and LCS (Longest Common String) [17] that always estimate how similar two strings literally looks like.

$$sim_{lex}(c1,c2)=\frac{MinLength(c1,c2)}{MinLength(c1,c2)-LenvenshteinDist(c1,c2)}, if, sim_{lex}<0, sim_{lex}=0$$

And

$$sim_{lex}(c1,c2)=\frac{2*LCS(c1,c2)}{Length(c1)+Length(c2)}$$

This two methods combined solve the situation like “speed_km_h” and “SPD(KM/H)”.

The semantic resolution is found on the semantic web. It solves the semantic similarity if two vocabularies have distinct literal text. WordNet is a large lexical database of English, developed under the direction of George A. Miller (Emeritus). Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Through WordNet, we can estimate how two conceptions are semantically near:

$$sim_{sem}(c1,c2) = \frac{Synset(c1) + Synset(c2)}{Relations(c1,c2)}$$

Here, Relations(c1, c2) indicates the numbers of words when adding up all synsets in the relationship between c1 and c2 if the relationship exists. Synset(c1) indicates the numbers of words when adding up the synsets which belong to the relationship between c1 and c2 if the relationship exists.

The constraint anatomy and match mainly focus on how the properties two conceptions own are similar. [18] considers using a vector to represent the properties' dimensions of a conception. Here we use the formula:

$$sim_{attr}(c1,c2) = \cos(V(c1), V(c2)) .$$

The V(c1) indicates a vector of conception attributes' dimension. Each value in the vector represents the how many the same dimensions included in the attribute. For example, V(c1)={2,1,0,3} means that there are two dimensions of "speed", one dimension of "weight", none of "temperature" and three of "pressure". That is, the higher probability they are resemble in their attributes, the higher the similarity between two conceptions in constraint level is.

The structural interpretation is constituted on the difference among the topology of various conceptions. If an ontology tree rooted by a top conception is much like another one, the structure of two top conceptions might be similar in structure as their sub-tree is like. Structural Matching is mainly focus on how two conceptions are physically similar due to their model structure. [19] recommends that estimating and comparing how two nodes weight in their own hierarchy tree is an efficient and quick way. APS (a-priori-score) score [20] put forward by Schickel-Zuber and Faulting is simple as follow:

$$APS(c) = \frac{1}{n+2} .$$

Here, n indicates number of total descendants of conception node c in the ontology tree. We can calculate the ratio of two conceptions APS in order to estimate the status of two conceptions in the ontology tree is similar.

After we calculate the four level ingredients influencing the mapping degree, a synthesized method can be concluded:

$$MappingDegree(c1,c2) = \sum_{i=0}^{4} P(sim(c1,c2)) * w_i$$

Here sim(c1, c2) means the four-level estimation mentioned above. **Fig. 6** shows how two ontolgoy models automatically generate mapping relationship through the four-level estimation.

The mapped ontology tree models accept the CBV specifications containing how the mapping relationship among applications looks like. Core Business Vocabularies are brought in and analyzed by the ontology tree and finally a mapped template of CBV is generated.

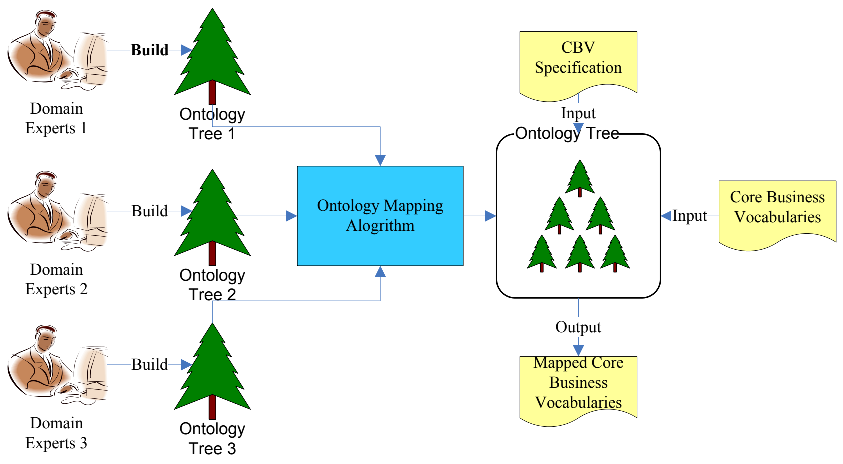


Fig. 5. The Work Flows of Semantic Unification Module

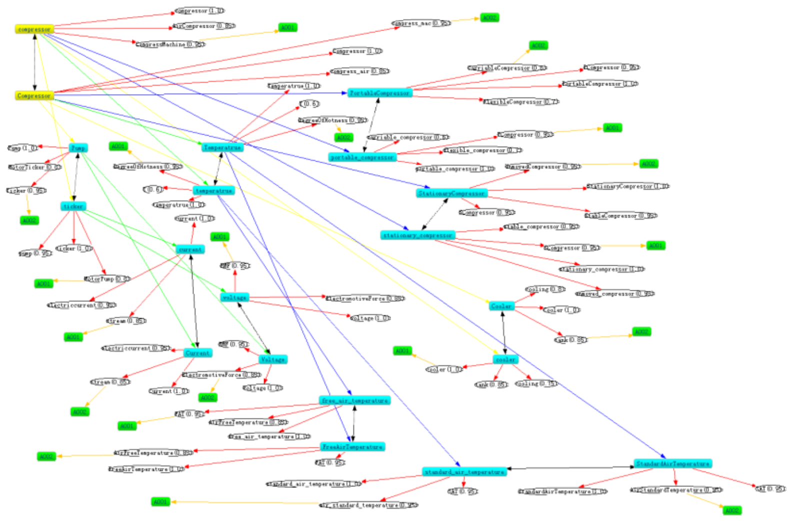


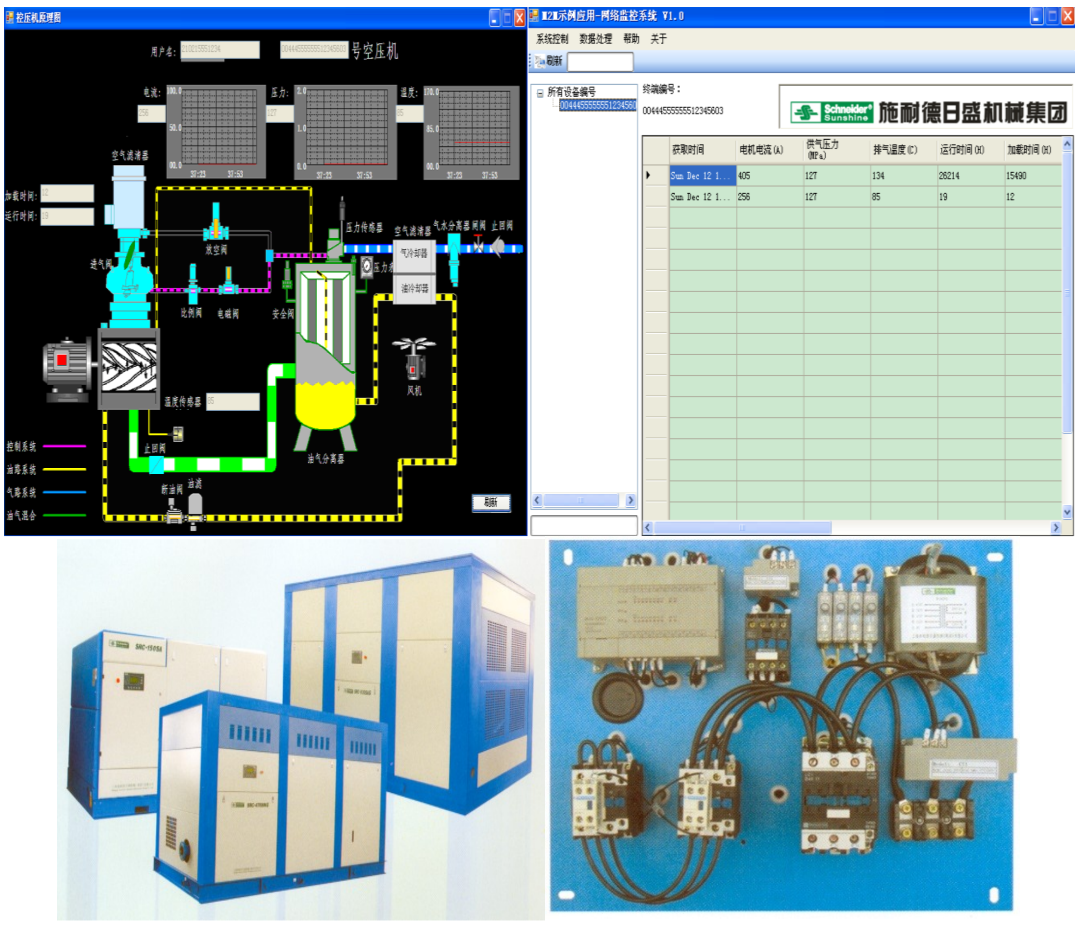
Fig. 6. Ontology Model & Ontology Mapping Relationship in Air Compressor Domain

6. Implementations and Experiments

Based on proposed architecture and key technologies, we have implemented the M2M middleware. The MAAP protocol is implemented by native C code and we virtualized them with JNI (Java Native Interface) for middleware to call. The syntactical unification module is involved with large quantity of XML parsing and encapsulation and the technique of DOM and library of JDOM [21] is utilized. In the part of semantic mapping of Core Business Vocabulary, we choose the WordNet 2.0 [22] developed by Princeton University and the JWNL (Java WordNet Library) which can calculate the semantic resembling degree of two words (Noun) through the steps between them in the net of word. In the implementation of event processing, we tried the CEP Engine - Esper Event Engine and combined with the EPL language. The interfaces to enterprise applications for PULL are generally achieved by the technology of Web Service while the ones for PUSH are realized by JMS of ActiveMQ [23]. And the Timely Push relevant to ETL technique is based on the open source tool of KETTLE

[24].

After implementation, the M2M Middleware has been applied in Schneider Sunshine Air Compressor Monitor System, as shown in Fig. 7. The Schneider Sunshine Air Compressor Monitor System is aimed to monitor the states of multiple air compressor through DTU connected with PLC. The states include basic parameters of temperature, pressure, current of phases, voltage, loading time, oil and lubrication quantity, and cautions. The whole middleware is deployed on the Windows Server 2003 and connected with the MySQL 5 Data base. As to the scalability of M2M Middleware, it is reflected by the factor that Esper Event Engine is not only based on EPL, but also implemented by JAVA Interface with easy code maintaining. Moreover, the platform-independent interface through Web Service has high scalability for extra applications to be integrated with.



终端编号：

004445555555512345603

获取时间	电机电流(A)	供气压力(MPa)	排气温度(C)	运行时间(H)	加载时间(H)
Sun Dec 12 1...	405	127	134	26214	15490
Sun Dec 12 1...	256	127	85	19	12

Fig. 7. The Demonstration Application - Schneider Sunshine Air Compressor Monitor System

We found that the total workload of code maintenance and modification for application integration is sharply decreasing compared to the one of original work if directly integrated into the M2M Platform, as shown in Table 8, comprehensively from 6 main performance indexes.

Table 8. The Comparison of Two Application Integration Ways

	Num of Interfaces	Average Num of Parameters	Code Quantity of Modification (LOC)	Period of Development (Month)	Data Structure of Interaction	Platform Independency
Without M2M Middleware	15	3.9	3500	3	4	N
With M2M Middleware	7	2.1	1500	1.5	0	Y

As the number of terminal devices is limited, we use the average latency instead of the throughput to evaluate the performance difference between the demonstration application with and the one without M2M Middleware. Table 9 shows how the latency was when a binary code of 23 bytes being uploaded and analyzed with the Mode of PUSH. We can summarize that if the application integrates with the M2M Middleware, the latency will increase by 8.3% to the situation without M2M Middleware. The performance testing is based on the M2M Platform of China Telecom whose IP Address is 116.229.239.181 and the server of High Reliable Software Lab in Shanghai Jiaotong University whose IP Address is 202.120.40.98 in education network. The latency of MAAP Protocol will be determined largely on the location of M2M Middleware Sever and the M2M Platform workstation.

Table 9. The Latency of M2M Middleware

Average Latency (Millisecond)	MAAP Data UP	Syntactical Unification	Semantic Unification	Esper Engine for Event Processing	JMS Pushing	Total
With MiddleWare	4358	8.4	21.6	3.3	226	4617.3
Without MiddleWare	4264	--	--	--	--	4264

7. Conclusion and Future Work

Hereby, we have put forward an M2M Middleware as an approach for enterprise applications to be integrated with the M2M Platform agilely and flexibly. The contribution of our work lies in the following aspects: A) An overall architecture of M2M Middleware and its requirement have been proposed to provide an agile integration approach for enterprise applications. B) A new syntactical unification approach has been raised. It combined with Esper Event Engine and unified XML-based syntax subscription to help various applications engaged in without the knowledge of details of devices. C) A novel and semi-auto semantic unification approach has been applied in our M2M Middleware that can promote the information share among applications and terminal devices. D) Interaction with applications are multi-pattern bring about great convenience when different communication requirements arrive. E) We verify our middleware through the real working environment and evaluate the performance which proved less performance loss occurred.

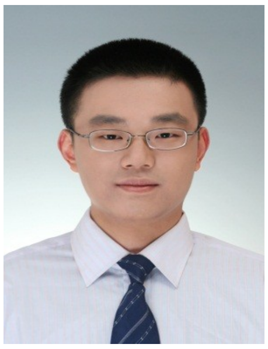
Compared to the current related works, first, our M2M middleware overcomes the obstacles when enterprise applications need to be flexibly integrated with M2M Platform through the technique of XML subscription and CEP which make it possible to syntactically unify the integrating specification and to have the enterprise applications loosely coupled with the M2M platform in the aspect of business logics. Second, it is more convenient and approximate to share information through the semantic unification of CBVs based on domain ontology and mappings which fit for the domain of M2M, if the requirement of information share is needed. Third, the pattern of interaction with enterprise application is diverse rather than simple way in most of the M2M applications so that various interactive modes can be considered according to data whatever is various in format, functionality and quantity.

The future work will inevitably concentrate on, 1) To try to import more terminal devices and applications in order to promote the M2M Middleware's universality as well as improve our experimental quality, for example, throughput testing to diagnose where on earth bottleneck exists. 2) To find more elaborate ways in semantic mapping of CBVs through discussing with domain experts and using mature lexical tools. 3) Unified specifications of integration rules will be researched and documented for the sake of agility and flexibility.

References

[1] http://en.wikipedia.org/wiki/Machine-to-Machine#cite_note-1
[2] http://www.gs1.org/epcglobal
[3] Qiang Wang, Wooseok Ryu, Soohan Kim and Bonghee Hong. "Demonstration of an RFID middleware: LIT ALE manager," in *Proc. of CIKM'2009*. pp. 2071-2072, 2009. Article (CrossRef Link)
[4] Hwang, H.J. and Choi, J.T., "Design of an aspect-based framework to improve the dynamic management of RFID middleware," in *Proc. of CIT*. 2007, pp. 955-960, 2007.
[5] Anirudh Ghayal, Zuber Khan and Rajat Moona, "SmartRF: A flexible and light-weight RFID middleware," in *Proc. of the 2008 IEEE International Conference on e-Business Engineering,* 2008.
[6] Wenhui Hu, Wei Ye, Yu Huang and Shikun Zhang, "Complex event processing in RFID middleware: A three layer perspective," *Third International Conference on Convergence and Hybrid Information Technology*, 2008. Article (CrossRef Link)
[7] Xiaoyong Su, Chi-Cheng Chu, B. S. Prabhu and Rajit Gadh, "Creating an RFID data integration framework for enterprise information system," *International Journal of Internet Protocol Technology*,vol. 4, no. 4, Nov. 2009.
[8] Sudha Krishnamurthy, Omer Anson, Lior Sapir, Chanan Glezer, Mauro Rois, Ilana Shub and Kilian Schloeder, "Automation of facility management processes using machine-to-machine technologies," *The Internet of Things, Lecture Notes in Computer Science*, vol. 4952, pp. 68-86, 2008.
[9] Shinji Kitagami, Yosuke Kaneko, Akihisa Yasuda, Harumi Minemura and Jun Sawamoto, "Communication method for remote device control using the Internet and its evaluation," in *Proc. of the 10th WSEAS international conference on Applied computer science*, 2010.
[10] http://en.wikipedia.org/wiki/Plain_Old_Java_Object
[11] http://www.w3.org/RDF/, W3C
[12] Menghan Chen and Beijun Shen, "A semantic unification approach for M2M applications based on ontology," in *Proc. of the 7th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Oct. 2011.
[13] http://esper.codehaus.org/
[14] Teixeira, P.H.D.S., Clemente, R.G., Kaiser, R.A., and Jr., D.A.V. "HOLMES: an event-driven solution to monitor data centers through continuous queries and machine learning," in *Proc. of DEBS*, pp. 216-221, 2010. Article (CrossRef Link)
[15] http://esper.codehaus.org/esper-2.0.0/doc/reference/en/html/index.html
[16] http://en.wikipedia.org/wiki/Levenshtein_distance, Levenshtein Distance

[17] http://en.wikipedia.org/wiki/Longest_common_substring_problem, Longest Common Substring Problem
[18] Minghua Pei, Kotaro Nakayama, Takahiro Hara and Shojiro Nishio, "Constructing a global ontology by concept mapping using wikipedia thesaurus," in *Proc. of 22nd International Conference on Advanced Information Networking and Applications - Workshops* (aina workshops 2008), Mar. 2008.
[19] Ying Wang, Weiru Liu and David Bell, "A concept hierarchy based ontology mapping approach, school of electronics," in *Proc. of Electrical Engineering and Computer Science*, Aug.2010.
[20] Schickel-Zuber, V. and Faltings, B.: "OSS: A semantic similarity function based on hierarchical ontologies," in *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 2007.
[21] http://www.jdom.org/
[22] http://wordnet.princeton.edu/
[23] http://activemq.apache.org/
[24] http://kettle.pentaho.com/



Menghan CHEN received B.S. degree from department of Computer Science of Shanghai Jiaotong University in 2009. And now, he is going to receive M.S. degree of software engineering from School of Software of Shanghai Jiaotong University. His research area is about M2M and its relevant applications in the domain of Internet of Things

Prof Beijun SHEN received B.S. degree from department of Computer Science in 1991 and M.S. degree in 1994 from Shanghai East China University of Science and Technology. She received Ph.D. degree from Software Institute of Chinese Academy of Sciences in 2002. Her research area is about Software Process, Software Engineering in Cloud Computing, Software Evolution and Maintainance, Model Driven Development and Information Security. Prof Shen is the first IEEE CSDP in China, serves as the chief editor of SWEBOK, member of ISTQB and CSTQB of advanced level, Meanwhile, she also serves as the member of national software engineering committee and the national authorized system analyzer. Prof Shen has already published more than one hundred papers on top leveled international transactions and conferences.