On M2M Software Platforms

Dmitry Namiot, Manfred Sneps-Sneppe

Abstract—**This paper provides an overview for existing and upcoming system software platforms for M2M applications. In this article we discuss system software models from the developer's point of view, rather than network related aspects. The primary goal is to find the common and reusable aspects across existing models as well as discuss their possible coexistence. Can we extract the common elements for the different M2M software models? Are there some reusable patterns? What should developers and system architects pay attention to? These are the main issues addressed in this article.**

***Keywords*—M2M, communications, software standards, middleware.**

I. Introduction

Machine-to-Machine (M2M) is a category of Information and Computing Technology that combines communications, computer and power technologies that enable remote iterations with physical, chemical and biological systems and processes [1]. Simply, M2M traditionally refers to technologies that allow both wireless and wired systems to communicate with other devices of the same ability. M2M uses a device (such as a sensor or meter) to capture an event (such as temperature, inventory level, etc.), which is relayed through a network (wireless, wired or hybrid) to an application (software program), translates the captured event into meaningful information [2]. In other words, M2M is the flow of data between network connected devices without the need for human interaction. Connected devices are vehicle tracking devices, parking meters, billboards, etc. The absent (presence) of human interactions sets the difference between M2M and Internet of Things (IoT).

The above mentioned paper [2] presented our first attempt to classify M2M software projects. It summarized our previous attempts, like [3][4]. Actually, the main conclusion presented in the paper devoted to the standards of M2M software is the statement about the lack of standards. We think that there is no common standard for M2M software right now, and we will not see such standards in the future.

We are seeing the future of M2M programming as a perfect example of micro-service architecture [5]. In the micro-service architecture a large number of very small services are deployed and linked up to build systems. Each individual service is focused on the one clearly specified business problem. So, the full set of services is very understandable from the business point of view and very scalable (horizontal scale) in the same time.

So, by our opinion, in M2M (IoT) applications we will deal with many different APIs simultaneously. The key factors are integration (in general, it is what micro-services architecture is about) and time to market (as a key indicator for the developers).

In this paper, we are trying to cover problems described in [2] from another side. More precisely, we are going to describe main exiting services and APIs with idea to discover the common set of micro-services (the kernel of M2M services). It will let us present the requirements for integration of micro-services.

II. M2M platforms

There are three types of platforms that cover the M2M Service Delivery platform market: Connected Device Platforms (CDP), Application Enablement Platforms (AEP) and Application Development Platform (ADP).

Connected Device Platforms are software elements that help to facilitate the deployment and management of connected devices for M2M applications over cellular networks. Some of the vendors may use the abbreviation DCT (Device Connection Platform) [6].

Application Enablement Platforms are designed to provide the core features for multiple M2M applications. They ease the data extraction and normalization activities, so M2M applications and enterprise systems can easily consume machine data. These platforms also assist in device and machine management.

Application Development Platforms provide a standardized service layer and APIs, as well as common frameworks for cellular operators, service providers and device manufacturers.

Of course, as for the most "classifications" in the programming area, the boundary between platforms is not so strong and clean. But in general, this classification lest split systems by their core functionality.

Let us present some simple explanation for the functionality.

CDP is about device management in the first hand. Mostly, it is for telecom providers. Device management is one of the first priorities for them.

AEP is shortly about clouds. As per the modern view, it is about cloud-based data store and processing.

ADP is, probably, the most explainable and targets M2M developers. Actually, the boundary between AEP and ADP

Article received Jul 20, 2014.

D.Namiot is senior researcher at Open Information Technologies Lab, Lomonosov Moscow State University. Email: dnamiot@gmail.com

M. Sneps-Sneppe is with Institute of Mathematics and Computer Science, University of Latvia. Email: sneps@mail.ru

is very limited sometimes. AEP could include rich development tools. So, some of the sources mention two M2M platforms only: CDP and AEP.

The whole line from the top to bottom is: Application – AEP – CDP

Technically, on the AEP level, we should deal with a wide set of tasks. E.g., we can mention also API and SLA exposure, Data management, service integration, ecosystem for applications, etc. Another generic explanation for AEP's functionality is M2M middleware.

CDP, as we wrote above, is a service portal, at the first hand. This portal should cover billing and policy control, bearer service, service ordering and subscription, SIM-cards management, etc.

We should note, that ETSI [7] suggests, practically, the similar sub-division (Figure 1)

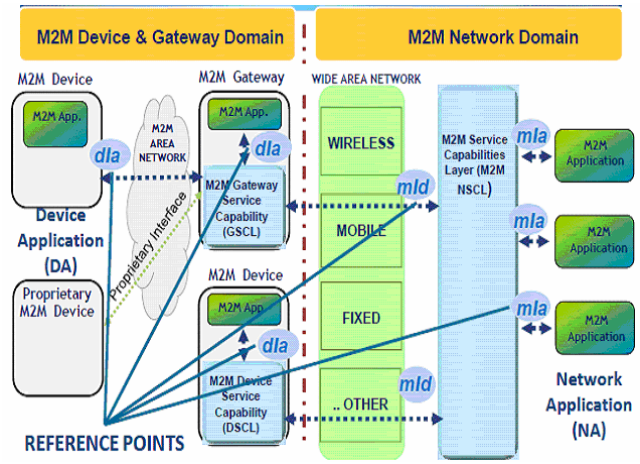

Figure 1. M2M ETSI [7]

Here M2M Device & Gateway Domain is an analogue of CDP, and M2M Network Domain is an analogue of AEP.

III. Connected Device Platforms

Let us see the typical functionality for CDP.

There are three main tasks:

- *Connectivity management*
- *Subscription management*
- *OSS/BSS*

CDP should allow the automation of the business processes between the operator and enterprises. The typical example is Ericsson DCP [8]. It is a secure portal offering businesses a comprehensive suite of tools to manage connected devices across their operations.

Typically, device management should control M2M devices in the real time and usually cover the following areas:

- Centralized Control for the devices. It should allow customers to edit configurations, update firmware, download software and monitor the status and location of user's remote assets via a web browser.
- Groups Control for the devices. Customers should be able to group (organize) devices in order to perform various business tasks or in order to simplify the network.
- Scheduled Operations. It should automate tasks, such as firmware updates, reboots, polling, uploading, etc. The scheduler executes scripts on a one-time or recurring basis.
- Alarms and Notifications. This functionality lets receive immediate notification when a device enters a particular state and take corrective action. Note, that it can cover geo-fence applications too [9].
- Carrier Subscription Management. This functionality lets, for example, activate or deactivate cellular lines and monitor data usage

The typical example for the CDP is Etherios Device Cloud [10]. Figure 2 illustrates the typical M2M dashboard from Bell M2M [11]

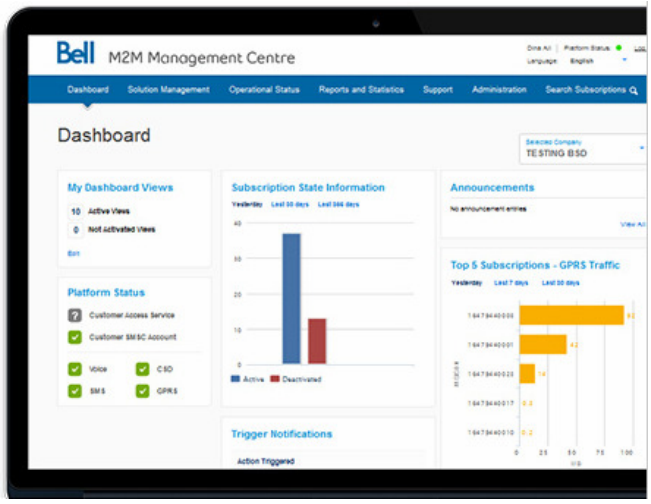

Figure 2. M2M dashboard [10]

CDP lets connect any device to Device Cloud. The connecting software should be available for the different platforms (e.g., Android, Java SE/ME, etc.).

One of the features, usually supported by CDP, is two-way messaging for full cloud-to-device messaging and control. Device management and troubleshooting tools from CDP should include configuration edits, firmware updates and device reboots [12].

IV. Applications Enablement Platforms

According to ABI Research [13], application enablement platforms (AEPs) enable quicker and less expensive application development as well as granular remote device management for developers and solution providers as they consider the optimal approach for building and deploying their M2M solutions.

For example, Axeda AEP (Figure 3):

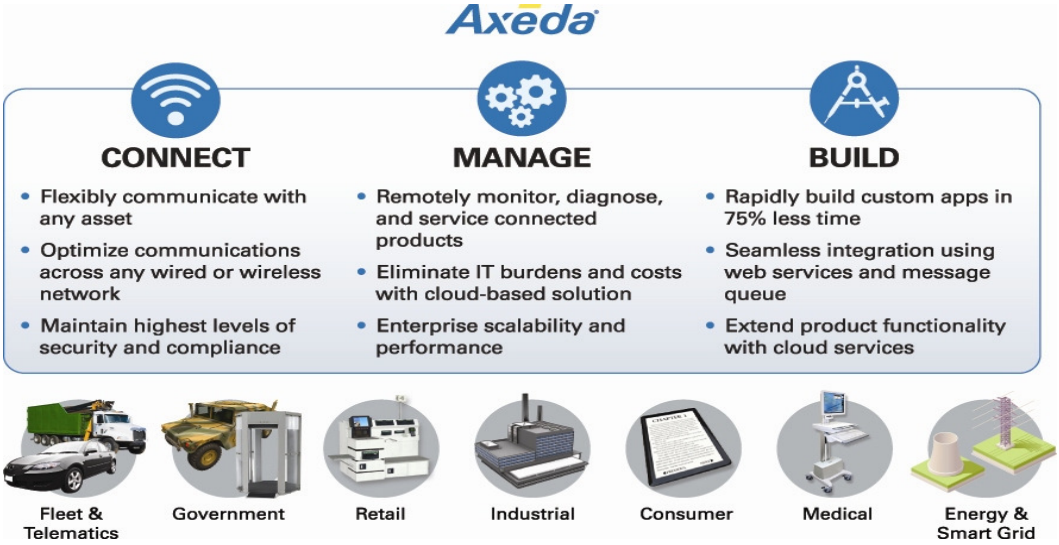

Figure 3. Axeda

It includes:

IoT Connectivity tools. They let connect any product using any device, over any communication channel for any application.

Scalable and Secure Data Management. It lets manage, process, and store millions of daily transactions as well as users and device groups in an intuitive and completely secure environment.

Fast Application Development tools. Axeda provides powerful development tools and flexible APIs accelerate custom application development. So, the boundary between AEP and ADP is not so strong.

Simplified Enterprise Systems Integration. Axeda provides standards-based integration framework accelerates integration between the Axeda Platform and enterprise systems, including ERP, CRM, billing, and data warehouses.

Data management, on the first hand, provides a Data Model. It is designed for storing M2M data and managing device and asset types, M2M data items, locations, alarms, and files. Connected asset attributes include default attributes such as organization, location, contacts, groups, and conditions. Models are easily enhanced with extended database objects to accommodate customization. Note, that this area is a subject for many academic papers (M2M ontology [15][16]), but it looks like de-facto (vendors initiated) standards will prevail.

Data management supports a flexible rules engine for processing incoming data, responding to events and alarms, and triggering actions on the Axeda Platform. Note, that rules engine includes an intuitive UI too. It lets rapidly implement sophisticated rules with thresholds and expressions.

We should mention two main elements here: orientation to mashups and fast prototyping. Developers can extend the rules engine using Axeda's Scripting API to "mash up" platform capabilities with other cloud-based services. Another interesting moment is the distributed engine. Developers can run expression rules on the Axeda Platform or threshold rules remotely and natively on the edge device to reduce communication costs. Platform's rule timers allow developers to create a timer to execute rules on a schedule. The scheduler could be customized too.

The middleware (processing engine) handles device data, files, alarms, events, locations, geo-fences and all processing for the platform. It includes extensive built-in security capabilities to manage, users, roles, user groups, and device groups. A configuration console enables you to manage rules and model definitions, asset grouping, notifications, alarms, user groups, and permissions.

As per developers API many implementations follow to Web Services model. At the same time, we should mention the shift towards REST API and more developers-friendly formats: JSON (JSONP) vs. XML [17].

Another hot trend is build-in statistical processing. This approach correlates with the common interest to big data. But it could have the specific for M2M applications. Real time data flow processing is more preferable for M2M. So, we think that stream processing should prevail in big data solutions for M2M [18].

V. Application Development Platforms

Figure 4 illustrates ETSI-compatible software development platform [19]. Service-oriented architecture should be platform-agnostic and provide a smooth binding with HyperText Transfer Protocol (HTTP) and Constrained Application Protocol (CoAP).

A RESTful architecture is usually based on some hierarchical resource tree. Here we should mention again the efforts in M2M ontology and repeat again, that most vendors follow to the own solutions. Note that constrained M2M devices present the biggest challenge for the developers. So, ADP should support a RESTful and lightweight device

management approach via M2M Gateways that cater to M2M requirements from constrained M2M Devices.

The same is true for resource manipulations for a broad range of M2M Devices, especially constrained M2M Devices.

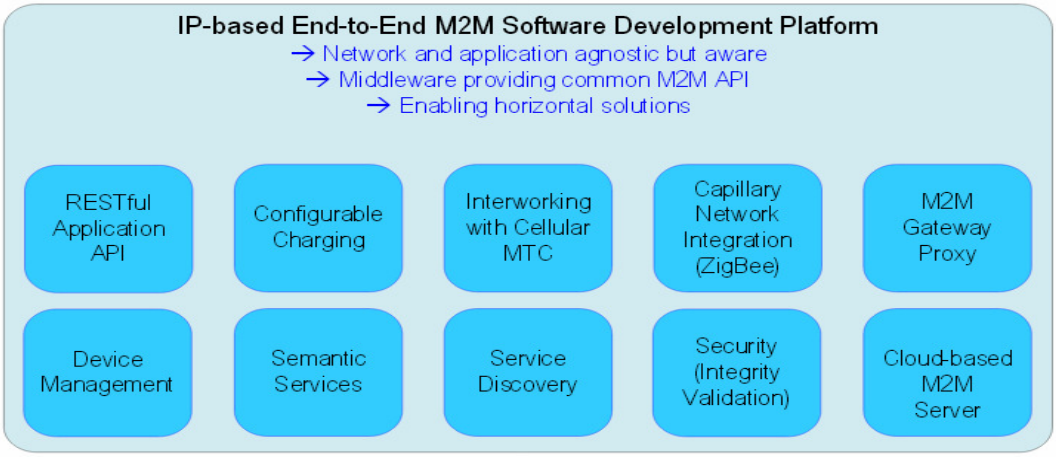

Figure 4. Software Development Platform InterDigital

Classically, RESTful architecture is based on a hierarchical resource tree and standard resource manipulation methods, including so called CRUD group: CREATE/RETRIEVE/UPDATE/DELETE.

RESTful architecture is stateless and based on Client/Server model. The CRUD group presents the uniform interfaces. Requests and responses in this client/server model are built around the transfer of representations of resources. Each resource can have “unique address”, “attributes”, and “sub-resources”. Traditionally, CRUD methods could be extended with one additional function – Subscription.

Note, that in the unified model, nodes in resource tree to present not only sensors and devices. Mobile users, groups of users, access rights, etc. are also “nodes”. As per ETSI requirements, we have several points related to the access rights [20]:

- A M2M device should be able to register its capability information (e.g. access technology, its serial number, its accessible address, allowed user list, etc.) to the M2M System.

- M2M devices and M2M gateways should be able to perform access control that checks the access right of the end-user.

- M2M devices should be alternatively able to perform the access control of M2M devices.

Traditionally (as it is borrowed from telecom), ADP could provide a special functionality for billing. In the terms of IMS telecom architecture, M2M server is so called Service Capability Server (SCS). With a configurable charging architecture, the Service Capability Server (SCS) (e.g., an M2M Server) can access charging records generated within 3GPP networks as well as charging functions within the 3GPP network can access and leverage charging records generated within the SCS [21].

The unified web-based API (SDK) requires, obviously, some intermediate nodes (proxies) for legacy M2M devices. It is required by ETSI model. For example, Figure 5 illustrates the common architecture [21]:

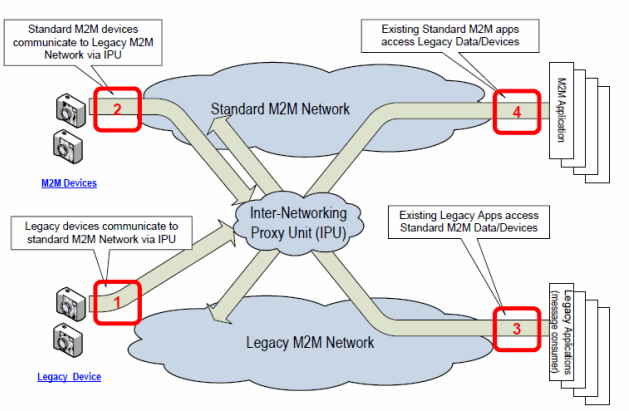

Figure 5. M2M internetworking

The central node in Figure 5 is Inter-Networking Proxy for legacy devices.

On the one hand, for the end developer, everything looks pretty transparent and simple. It is yet another REST API, where ADP (AEP) leverages the details. On the other hand, concerns for the efficiency will result, in our opinion, to the attacks on the above described model. We see, at least, two directions for changes. The figure 6 describes data model for FI-WARE project.

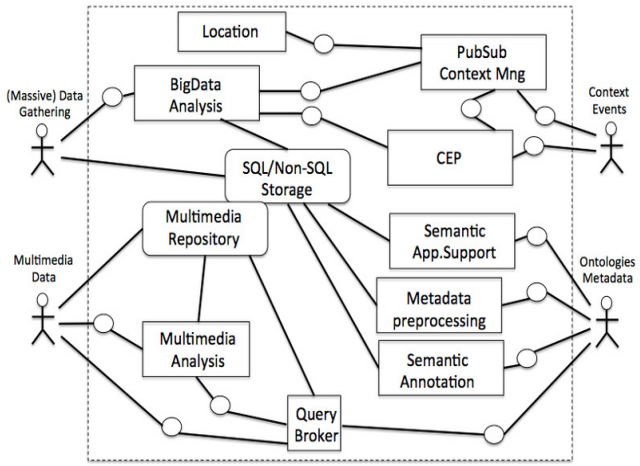

Figure 6. FI-WARE data model

It is a much-more spaghetti-like model with the different interfaces.

The second direction for changes (by our opinion) is the cloud based model itself. Actually, cloud based access to the devices (sensors) has got own weaknesses. Firstly, it can face security concerns. Secondly, it could sometimes be more costly than direct access to M2M (IoT) devices. And sometimes the development for cloudless system could be faster too. Just because any cloud-based solution is always some generalized model and it may lose the convenience details, comparing with the libraries, especially oriented to the particular device (class of devices). So, it could be faster in terms of convenience for the developers, at the first hand. The typical example is Bluetooth Low Energy. As per current model from Apple, iBeacons (BLE devices) do not assume any cloud. End-user devices should obtain the data (advertisement) right from beacons. And this model (for iBeacons only, of course) is faster to develop with than, for example, FI-WARE.

As per our opinion, it is the future of M2M development tools. We think that unified toolbox will be very difficult to maintain – think about creating support for every new device. We think that devices will provide partial APIs and M2M "frameworks" will combine them. That is why we wrote about micro-services at the beginning.

V. Conclusion

In this paper, we describe the current state of M2M platforms. At this moment we have a system with the reasonable, logical sub-division of the implemented functions. At the same time, we think that the high diversity in the devices (sensors) used in M2M will change this picture in the nearest time. We predict that instead of the unified systems developers will deal with micro-services oriented to the particular devices (classes of devices). ISo, the true developers-oriented stack for M2M is yet to be created.

References

[1] Brazell, J. B., Donoho, L., Dexheimer, J., Hanneman, R., & Langdon, G. (2013). M2M: the wireless revolution.

[2] Namiot, D., & Sneps-Sneppe, M. (2014). On M2M Software. International Journal of Open Information Technologies, 2(6), 29-36.

[3] Schneps-Schneppe, M., Namiot, D., Maximenko, A., & Malov, D. (2012, October). Wired Smart Home: energy metering, security, and emergency issues. In Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on (pp. 405-410). IEEE.

[4] Sneps-Sneppe, M., & Namiot, D. (2012, April). About M2M standards and their possible extensions. In Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on (pp. 187-193). IEEE.

[5] Blum, N., Boldea, I., Magedanz, T., Staiger, U., & Stein, H. (2009, July). A service broker providing real-time telecommunications services for 3rd party services. In Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International (Vol. 2, pp. 85-91). IEEE.

[6] Cackovic, V., & Popovic, Z. (2012, September). Device Connection Platform for M2M communication. In Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on (pp. 1-7). IEEE.

[7] "ETSI Machine-to-Machine Communications info and drafts" http://docbox.etsi.org/M2M/Open/ Retrieved: Jul, 2014.

[8] Ericsson DCP http://www.ericsson.com/ourportfolio/products/device-connection-platform Retrieved: Jul, 2014

[9] Namiot, D. (2013). GeoFence services. International Journal of Open Information Technologies, 1(9), 30-33.

[10] Mazhelis, O., & Tyrvainen, P. (2014, March). A framework for evaluating Internet-of-Things platforms: Application provider viewpoint. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 147-152). IEEE.

[11] Bell M2M http://mobilebusiness.bell.ca/industries-and-solutions/machine-to-machine/ Retrieved: Jul, 2014

[12] Etherios http://www.etherios.com/products/devicecloud/ Retrieved: Jul, 2014

[13] ABI Research https://www.abiresearch.com/market-research/product/1005785-m2m-software-platforms/ Retrieved: Jul, 2014

[14] Castro, M., Jara, A. J., & Skarmeta, A. F. (2012, July). An analysis of M2M platforms: challenges and opportunities for the Internet of Things. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on (pp. 757-762). IEEE.

[15] Chen, M., & Shen, B. (2011, October). A semantic unification approach for M2M applications based on ontology. In Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on (pp. 265-271). IEEE.

[16] Gronbek, I., & Biswas, P. K. (2009, October). Ontology-based abstractions for M2M virtual nodes and topologies. In Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on (pp. 1-8). IEEE.

[17] Sneps-Sneppe, M., & Namiot, D. (2012). M2M Applications and Open API: What Could Be Next?. In Internet of Things, Smart Spaces, and Next Generation Networking (pp. 429-439). Springer Berlin Heidelberg.

[18] Neumeyer, L., Robbins, B., Nair, A., & Kesari, A. (2010, December). S4: Distributed stream computing platform. In Data Mining Workshops (ICDMW), 2010 IEEE International Conference on (pp. 170-177). IEEE.

[19] Lu, G., Seed, D., Starsinic, M., Wang, C., & Russell, P. (2012, April). Enabling smart grid with ETSI M2M standards. In Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE (pp. 148-153). IEEE.

[20] Wu, G., Talwar, S., Johnsson, K., Himayat, N., & Johnson, K. D. (2011). M2M: From mobile to embedded internet. Communications Magazine, IEEE, 49(4), 36-43.

[21] InterDigital http://www.interdigital.com/wp-content/uploads/2012/10/InterDigital-M2M-white-paper_Oct2012.pdf Retrieved: Jul, 2014-07-14

[22] Namiot, D., & Schneps-Schneppe, M. (2013). Smart Cities Software from the developer's point of view. arXiv preprint arXiv:1303.7115.