26th Annual INCOSE International Symposium (IS 2016)
Edinburgh, Scotland, UK, July 18-21, 2016

Making Smart Cities Smarter – MBSE Driven IoT

Matthew Hause
PTC Engineering Fellow
MHause@PTC.com
140 Kendrick St., Needham, MA, USA

James Hummell
Consultant, MBSE
JHummell@MBSE.solutions
Phoenix, Arizona, USA

Abstract. The Internet of Things (IoT) is a system of systems (SoS) in every sense of the definition. A.P. Sage and others list five common SoS characteristics: operational independence of the individual systems, managerial independence, geographical distribution, emergent behavior and evolutionary development or independent life cycles. Typical examples include smart houses, the electric grid, complex military systems and so-called smart cities. The future of IoT success, including technology advancements and revenue generating potential across the business spectrum, is dependent on the application of solid Systems Engineering and Model Based Systems Engineering (MBSE) principals. Without MBSE, the complexity involved in the design, development, and deployment of IoT systems would consume both system and operational providers. Absent of any industry standards, IoT systems cannot be built in a vacuum and their success will only be realized through application of modern day systems engineering processes, methods, and tools. With the potential of 28 billion "things" connected to the Internet by 2020, it's not too difficult for anyone paying attention to this emerging technology trend to envision the massive scale of social, economic, and technological changes that will need to occur to realize this prediction. Technology advancements in consumer products will continue to evolve to facilitate connection to larger and larger IoT networks. This will be the catalyst that will drive entire infrastructures changes to: Federal, State, City, and local governments; product development companies; utility and service providers; and even to consumers and their homes in order to support the growing demand for connected products. The infrastructure and management will need to be established prior to, or in conjunction with, the smart systems that support them. This paper will show a traffic management system and connected systems in a large city and how an MBSE and SoS approach will help guide development.

Introduction

Kevin Ashton, cofounder of the Auto-ID Center at the Massachusetts Institute of Technology is known for inventing the phrase "Internet of Things". The phrase started as the title of a presentation Kevin made at Procter & Gamble in 1999; linking the new idea of RFID in P&G's supply chain to the then-red-hot-topic of the Internet. [2][3] Little did anyone know back then that less than 16 years later, in the age where the Internet and smart devices are now common-place, the "Internet of Things" would be emerging as the third wave in the development of the Internet. [1] This has resulted in the emergence of what are called "smart connected products."

What Are Smart, Connected Products?

Once composed solely of mechanical and electrical parts, products have become complex systems that combine hardware, sensors, data storage, microprocessors, software, and connectivity in myriad ways. These "smart, connected products" have been made possible by vast improvements in processing power and device miniaturization and by the network benefits of ubiquitous wireless connectivity. Smart, connected products have three core elements: physical components, "smart" components, and connectivity components. Smart components amplify the capabilities and value of the physical components, while connectivity amplifies the capabilities and value of the smart components and enables some of them to exist outside the physical product itself such as in the cloud. Smart, connected products require a rethinking of design. At the most basic level, product development shifts from largely mechanical engineering to true interdisciplinary systems engineering. [11], [12]

Physical components comprise the product's mechanical and electrical parts. Smart components comprise the sensors, microprocessors, data storage, controls, software, and, typically, an embedded operating system and enhanced user interface. In many products, software replaces some hardware components or enables a single physical device to perform at a variety of levels. Connectivity components comprise the ports, antennae, and protocols enabling wired or wireless connections with the product. Connectivity can be one-to-one, one-to-many, and many-to-many, and systems can use any or all of them. [11], [12] A typical SoS will use many-to-many connections with multiple products connected to many other types of products and often also to external data sources.

Connectivity allows communication between systems, but also provides additional features for distributed control and autonomy. Functions of the product can be limited to the product, distributed to a local controller, centralized in a federated architecture, deployed as virtual in the product cloud, as well as a combination of all of these. The correct configuration will depend on a variety of stakeholder and system drivers, and choosing the correct configuration will require complex systems engineering trade-off analysis. In the case of a smart city, this may involve single intersection controlled traffic signals, coordinated traffic signals through a major traffic thoroughfare, coordinated traffic signals through multiple major traffic thoroughfares, centrally controlled predictive and adaptive traffic management systems, and so forth. Coordination of traffic signals could be used to both speed up and slow down traffic. For example, US federal regulations impose speed limits on flow of traffic in order to limit fuel consumption and to support the Clean Air Act. So in addition to speeding up traffic flow, there is a financial incentive to slow down traffic as well. They may also wish to slow traffic flow as a means of encouraging people to use public transportation. All of these capabilities are now technically available. The "correct" choice will be made depending on priorities, costs, funding, phasing of funding, perceived and actual benefit as well as which solution delivers the benefits required by the city. Finally, the "correct" solution will change over time as requirements, infrastructure and enabling technologies change. Therefore, the architecture of the system of systems needs to be planned not only for the immediate future and initial installation, but how the system will need to change over time to address future needs.

Enabling Technology

A number of innovations in hardware, software, systems, and communications technology have converged to make smart, connected products technically and economically feasible. These include breakthroughs in the performance, miniaturization, and energy efficiency of sensors and batteries; highly compact, low-cost computer processing power and data storage.

These all make it feasible to put computers inside products. Cheap connectivity ports and ubiquitous, low-cost wireless connectivity allow systems to communicate at low cost in both energy and communications. Tools that enable rapid software development and big data analytics provide a means to process the tsunami of data that is continually arriving. Finally, the new IPv6 internet registration system will open up 340 trillion trillion trillion potential new internet addresses for individual devices, with protocols that support greater security, simplify handoffs as devices move across networks, and allow devices to request addresses autonomously without the need for IT support. All of these advancements together have enabled the IoT to develop and eventually explode into consumer devices and the world. [11], [12]

IoT Infrastructure

Systems deploying the IoT require a different infrastructure, consisting of a series of layers known as a "technology stack". This includes modified hardware, software applications, and an operating system embedded in the product itself. Network communications support connectivity between systems and sensors and a product cloud containing the product-data database. A platform for building software applications and a rules engine provides user interface and interaction as well as common and exceptional behavior. The analytics platform, and smart product applications that are not embedded in the product provide a means of offsetting or "off-sourcing" much of the required memory and data processing requirements, limiting power consumption. Cutting across all the layers is an identity and security structure, a gateway for accessing external data, and tools that connect the data from smart, connected products to other business systems (for example, ERP and CRM systems). [11], [12]

In some cases, such as a smart traffic management system in a large city, the infrastructure will need to be established prior to, or in conjunction with, the smart connected vehicle technologies. The sheer number and types of devices, and their relationships within this connected system will require the application of Systems Engineering principals and MBSE tools to drive down and manage the complexity of the entire system of systems (SoS).

Once again, the "correct" configuration and deployment of the "system intelligence" will likely change over time. Starting en mass the 1970's, distributed control systems changed the dynamics of SCADA and other types of control systems. [13] By deploying control capabilities to locations other than the main control system, processing was off-loaded. Pre-processing as well as filtering of the data could take place, which cut down on communications interfaces. And finally, increased computer power meant that local autonomous systems could take over from the main control system when communications were down, or reaction time to strategic events was limited. Computing power and communication links have increased substantially since then. However, these resources are not infinite. Therefore the same sort of trade-of analysis will still need to be made regarding what data to collect, at what rate and what types of processing can be made. In addition, local autonomy decreases communication requirements, but increases the costs of the deployed systems and has the potential to increase the security risks. The more powerful that distributed smart devices become, the more at risk they potentially become of doing greater harm. Once again, this needs to be considered in the overall design. More on this later.

How IoT Influences Product Architecture

Smart, connected products require a whole set of new design principles, such as designs that achieve hardware standardization through software-based customization, designs that enable

personalization, designs that incorporate the ability to support ongoing product upgrades, and designs that enable predictive, enhanced, or remote service. Expertise in systems engineering and in agile software development is essential to integrate a product's hardware, electronics, software, operating system, and connectivity components. Product development processes will also need to accommodate more late-stage and post-purchase design changes quickly and efficiently. Systems engineers have adapted to this requirement and agile systems engineering initiatives are taking place within INCOSE and within industry. [9]

Security.

More than most products, smart, connected products create the need for robust security management to protect the data flowing to, from, and between products; protect products against unauthorized use; and secure access between the product technology stack and other corporate and private systems. This will require new authentication processes, secure storage of product data, protections against hackers for product data and customer data, definition and control of access privileges, and protections for products themselves from hackers and unauthorized use. Every smart, connected device may be a point of network access, a target of hackers, or a launchpad for cyberattacks.

Smart, connected products are widely distributed, exposed, and hard to protect with physical measures. Because the products themselves often have limited processing power, they cannot support modern security hardware and software. Smart, connected products share some familiar vulnerabilities with IT in general. For example, they are susceptible to the same type of denial-of-service attack that overwhelms servers and networks with a flood of access requests. However, these products have major new points of vulnerability, and the impact of intrusions can be more severe. Hackers can take control of a product or tap into the sensitive data that moves between it, the manufacturer, and the customer. The risk posed by hackers penetrating aircraft, automobiles, medical equipment, generators, and other connected products could be far greater than the risks from a breach of a business e-mail server. Organizations such as CIPR and INFRAGARD are examining security breaches to critical infrastructure including the smart grid, all manner of public utilities and transportation and communication systems. [14, 15] Therefore security needs to be designed in from the ground up. With everything starting off as connected we need to ensure that security is designed in. Either by using the appropriate tools that can ensure the data is being protected or creating the infrastructure in what we build to have the concept of securing the information.

There are also different regulations for data security, protection, logging, tracking, storage, and anonymity depending on location. The UK has the Data Protection Act [16] which requires that people are informed of what data is being collected, what it will be used for, and how long it can be kept. Adherence to these laws is essential as fines can be quite substantial for each data breach. Therefore, security as well as data tracking and possibly case management need to be integrated into the architecture to ensure that the system is compliant with the law.

Model-Based Systems Engineering (MBSE)

Nine years prior to the birth of the phrase "Internet of Things", the National Council on Systems Engineering (NCOSE) was founded in the United States. Its goals had been the development and promotion of systems engineering in the United States. After only 5 years these goals were extended to the international level when NCOSE became INCOSE. [4] Model-Based Systems Engineering (MBSE) is becoming the de facto means of systems

engineering within companies. MBSE requires an integrated approach. The INCOSE 2025 Vision states that, “The systems engineering tools of 2025 will facilitate systems engineering practices as part of a fully integrated engineering environment. Systems engineering tools will support high fidelity simulation, immersive technologies to support data visualization, semantic web technologies to support data integration, search, and reasoning, and communication technologies to support collaboration. Systems engineering tools will benefit from internet-based connectivity and knowledge representation to readily exchange information with related fields. Systems engineering tools will integrate with CAD/CAE/PLM environments, project management and workflow tools as part of a broader computer-aided engineering and enterprise management environment. The systems engineer of the future will be highly skilled in the use of IT-enabled engineering tools.” [10] This integrated vision is being actively developed by a number of standards, industry, university, and government organizations to enable the INCOSE 2025 vision.

The Systems Modeling Language (SysML)

The state of the art language for supporting these activities is the Systems Modeling Language (SysML). For systems of systems (SoS) this is done using the Unified Profile for DoDAF and MODAF (UPDM). UPDM implements the Department of Defense Architecture Framework (DoDAF), the Ministry of Defence Architecture Framework (MODAF) and the NATO Architecture Framework (NAF) using SysML. This provides a means of performing systems engineering on the SoS rather than simply capturing the architecture as a collection of models. Recent versions of UPDM architectures can now take the fourth dimension (time) into account. The paper “Architecting in the Fourth Dimension - Temporal Aspects of DoDAF”(Hause, Kihlstrom 2013) captures many of these aspects. The evolution of the system over time can be shown, as well as the dynamic interactions within the architecture as well as simulation of the architecture.

Elements of SysML

SysML is more than a diagramming notation. It also defines relationships between and properties of the elements which are represented on those diagrams. The SysML diagrams can be used to specify system requirements, behavior, structure and parametric relationships. These are known as the four pillars of SysML. The system structure is represented by Block Definition Diagrams and Internal Block Diagrams. A Block Definition Diagram describes the system hierarchy and system/component classifications. The Internal Block Diagram describes the internal structure of a system in terms of its Parts, Ports, Interfaces and Connectors. Parts are the constituent components or “Parts” that make up the system defined by the Block. Interfaces define the access points by which Parts and external systems access the Block. Connectors are the links or associations between the Parts of the Block. Often these are connected via the Ports. The parametric diagram represents constraints on system parameter values such as performance, reliability and mass properties to support engineering analysis.

Smart City Example Problem

The federal government has an initiative to reduce traffic on the highways and thoroughfares. They have decided to acquire a traffic management system to help them identify areas and times of high traffic density so they can take measures to alleviate the effects of it. Systems will include controlled parking facilities, availability monitoring and dissemination, emergency management, traffic control and prediction, and support for electric vehicles.

During the preparation of this paper, the authors built a complex model and simulation with more than 70 different diagrams. Of course, documenting all of these is beyond the scope of this paper, so we will limit ourselves to a few of the useful techniques in MBSE that can be used throughout the system and system development lifecycle.

What Are Smart Cities?

A smart city uses digital technologies or information and communication technologies (ICT) to enhance quality and performance of urban services, to reduce costs and resource consumption, and to engage more effectively and actively with its citizens. Sectors that have been developing smart city technology include government services, [5] transport and traffic management, energy, [6] health care, [7] water and waste. Smart city applications are developed with the goal of improving the management of urban flows and allowing for real time responses to challenges. [8] A smart city may therefore be more prepared to respond to challenges than one with a simple 'transactional' relationship with its citizens.

System Context

There are a number of ways to go about the analysis and design of this type of system. One could start from the bottom up, by researching the different types of sensors involved in sensing traffic flow, evaluating existing traffic patterns, investigate transportation demand management, and so forth. As we are trying to implement a system of systems, it would be best to get the Big Picture. This involves a number of techniques: context definition, stakeholder needs elicitation, use case definition, vision, goal and needs statements, capability definition and so forth. Any and all of these are useful in determining what we are trying to achieve, prior to defining how we are going to achieve it. Figure 1 contains the high-level context diagram.

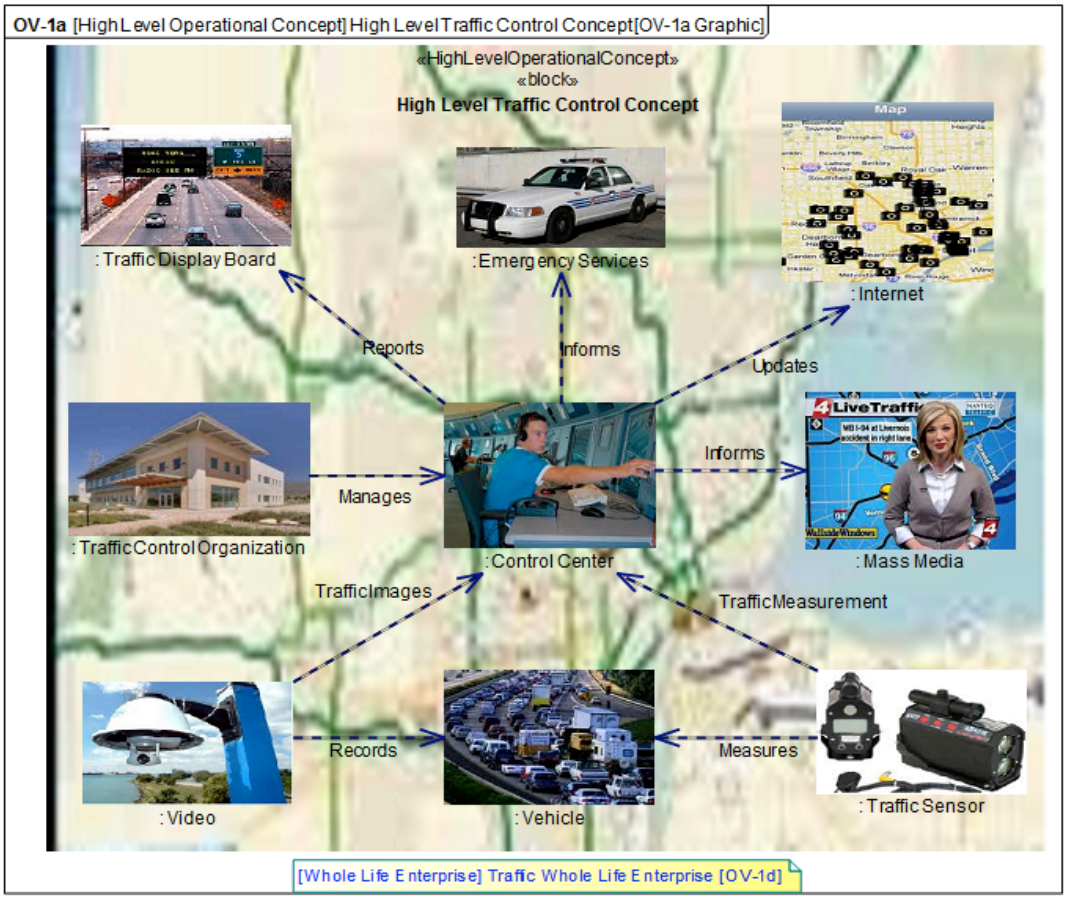

Figure 1. High level System Concept

At the center of the diagram is the traffic management control system. This type of diagram is normally presented as a set of boxes with lines between them. However, as we are going to use this diagram to communicate with non-technical stakeholders, the boxes have been replaced with graphics in order to more clearly communicate their intent and purpose. The main elements in the system context have been identified such as traffic display boards, emergency services, the internet, traffic sensors and videos, etc. A simple diagram such as this can be used to communicate with the stakeholders and as a means of identifying missing elements. Another means of identifying stakeholders and their goals is the use case diagram shown in Figure 2. Stakeholders can investigate different aspects of the systems such as: What goals are we trying to achieve? What is the purpose of the system? What improvements do we want?

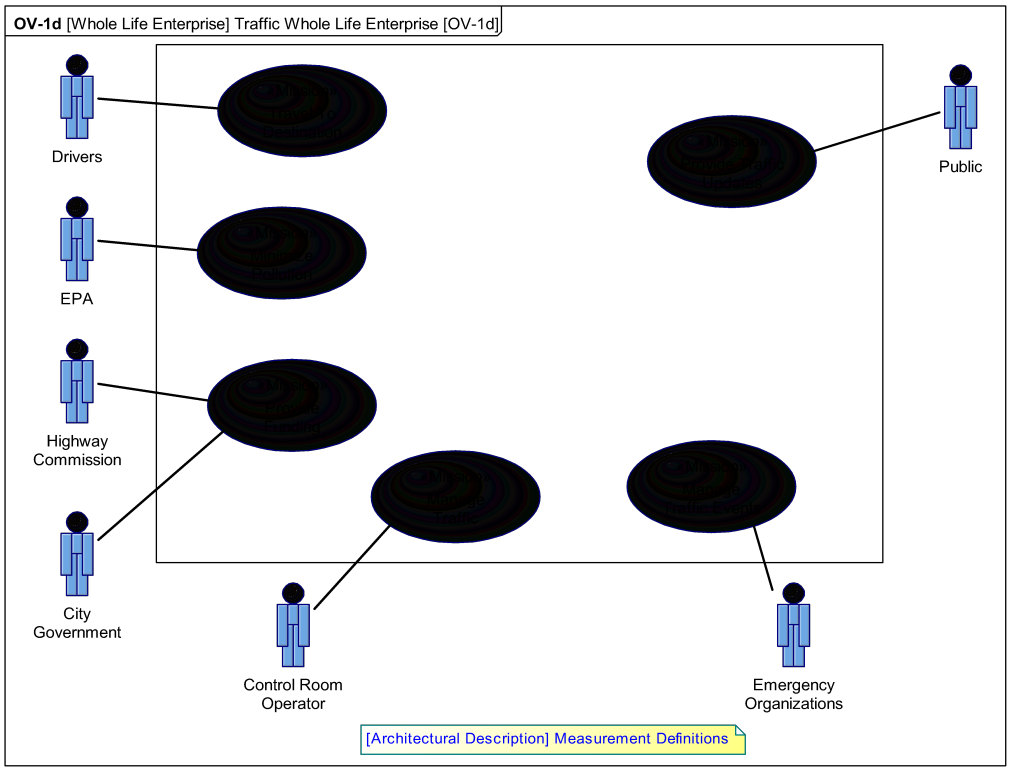

Figure 2. Stakeholders and their Use Cases

Another aspect we can capture using Modeling is to list what capabilities we want the different systems to have as shown in Figure 3. This helps to answer the question: What are we trying to achieve? Capabilities define the ability to achieve a desired outcome. For example, a parking control system can consist of the deployment of several people who keep track of the number of people entering and exiting a parking lot and communicating totals via radio. It could also be achieved via a fully automated control system connected via wireless devices. We need to define what we are trying to achieve prior to how we will achieve it in order to perform adequate trade-off analysis. Starting with this approach help you to think of what you want and not how it will be handled. In other words, if we don't know what we want to do, how do we know what will be the best way to achieve it? It's a different mindset that city planners and engineers need to do in order to not solve a problem with a given solution but think up new and innovative ways to solve the smart city's needs. Figure 3 documents some of the capabilities as well as the software and systems that will provide the capabilities such as Calculate Traffic Levels by the Traffic Flow Calculation SW. Realizing resources are normally added later in the development lifecycle once the architecture has been defined. They are included on the diagram for reasons of space.

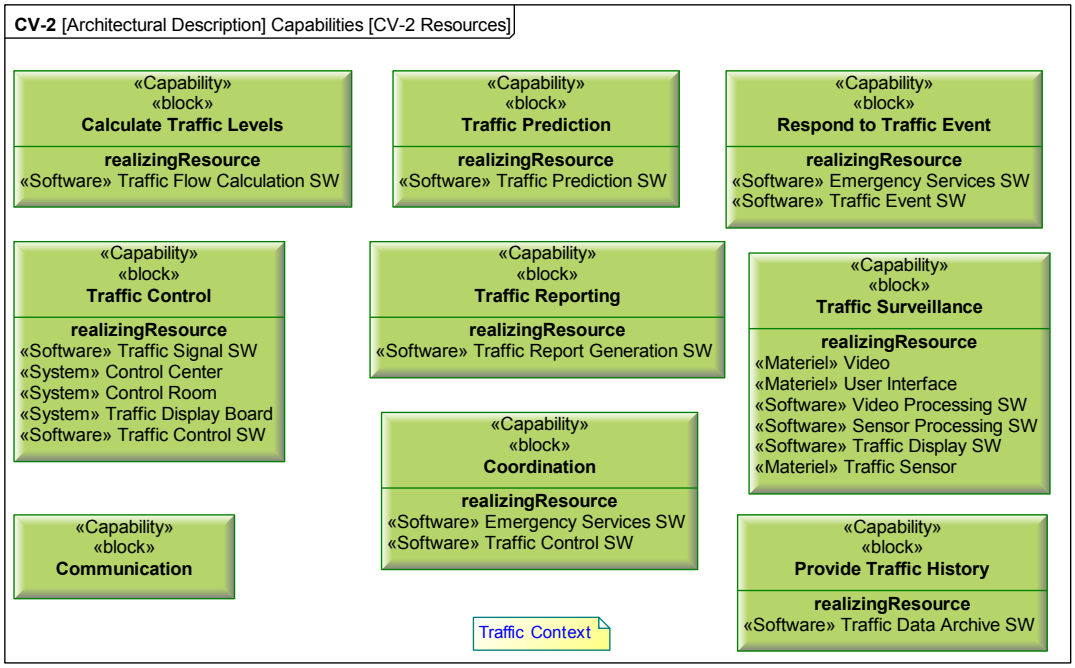

Figure 3. Traffic Management Capabilities

The Operational View

Operational views document the different behavior and logical infrastructure that are involved in the enterprise. This information starts us on the path of understanding what we may need to build, design, create, and document, etc. to enable communication between differing parts of the government, industry, and commercial sectors of the Smart City. Figure 4 shows the activities that have been defined to support the capabilities.

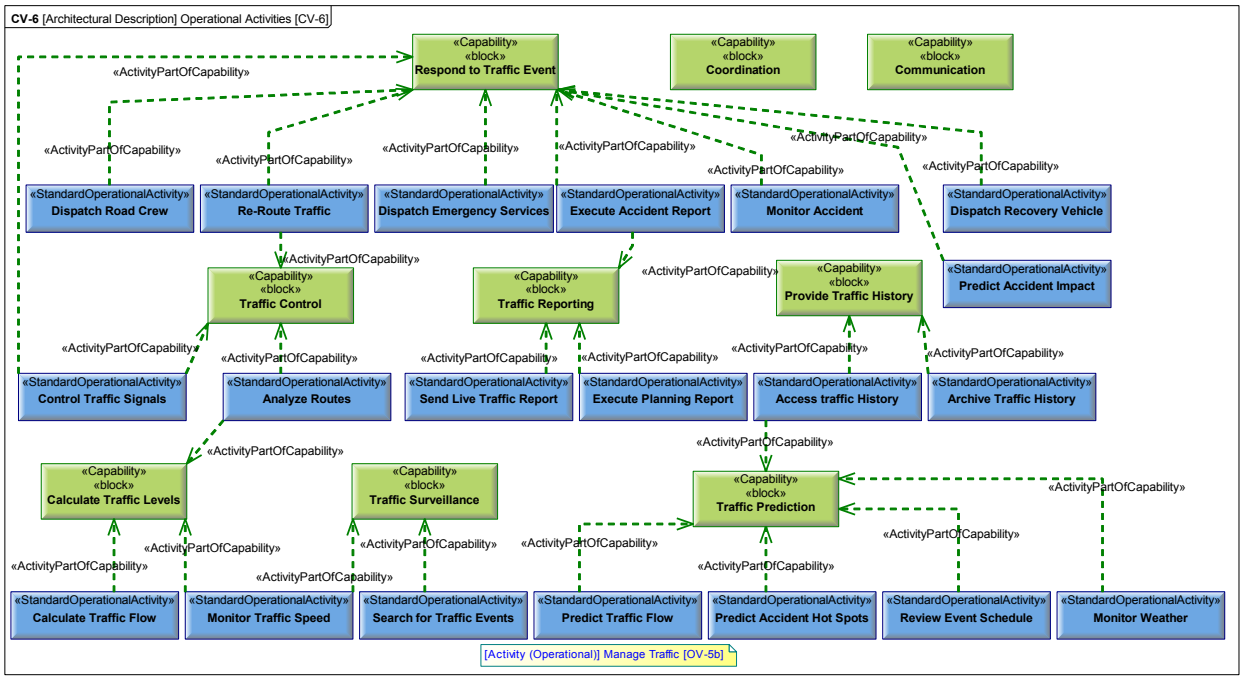

Figure 4. Operational View of the Traffic Context

The Operational activities support/implement the capabilities. For example, activities for the Traffic Prediction capability are Predict Traffic Flow, Predict Accident Hot Spots, Review Event Schedule, Monitor Weather and Access Traffic History. Each of these activities would be further decomposed into activity diagrams to define more detailed functional requirements and to understand what will be required. This will describe logical and physical flows in the system and can be simulated as well to discover timing analysis and other functional characteristics. Having defined these activities, a logical architecture is created corresponding to the entities that will implement the activities. The data flow defined in the activities will correspond to the data flows defined for the architecture. Figure 5 contains a logical architecture for the traffic flow system.

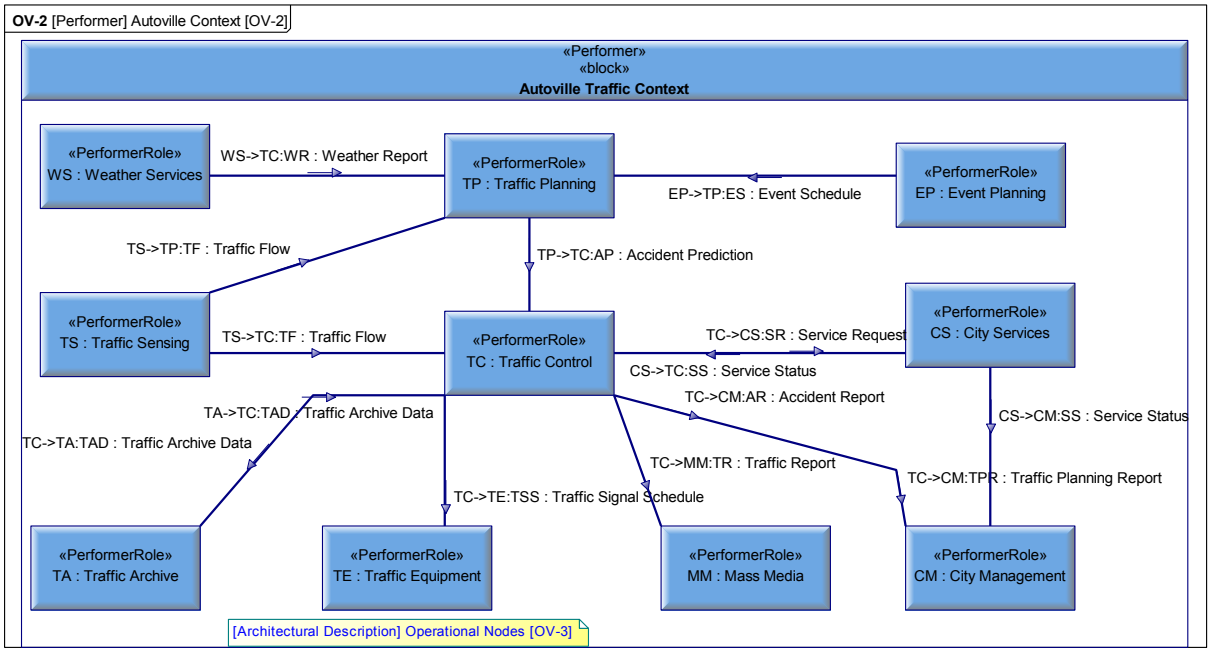

Figure 5. Operational View of the Traffic Context

The elements at this level are called performers in the DoDAF version of UPDM. They define abstract entities that encompass the logical groups of behaviors. These will define the requirements and interactions that the implementing systems, services and organizations will perform.

The Systems View

As we narrow our focus down to traffic control of the city we start to document the sub-systems involved in the communication of the information from traffic control to the different services and systems they need to talk with as shown in Figure 6. With this we can know that two different parts of the organization need to have a communication path and we can even document what type of data and/or materials will be exchanged between the different departments. The system interface view defines the different systems and required interfaces in the system without necessarily specifying how they will communicate. Figure 6 show the control system, Control Room Operators and User Interfaces. Interfaces to external entities such as Event Venue, Mass Media, Internet, weather Services, etc. are show. System entities such as Traffic Display Boards, Traffic Signals and Traffic Sensors are also shown.

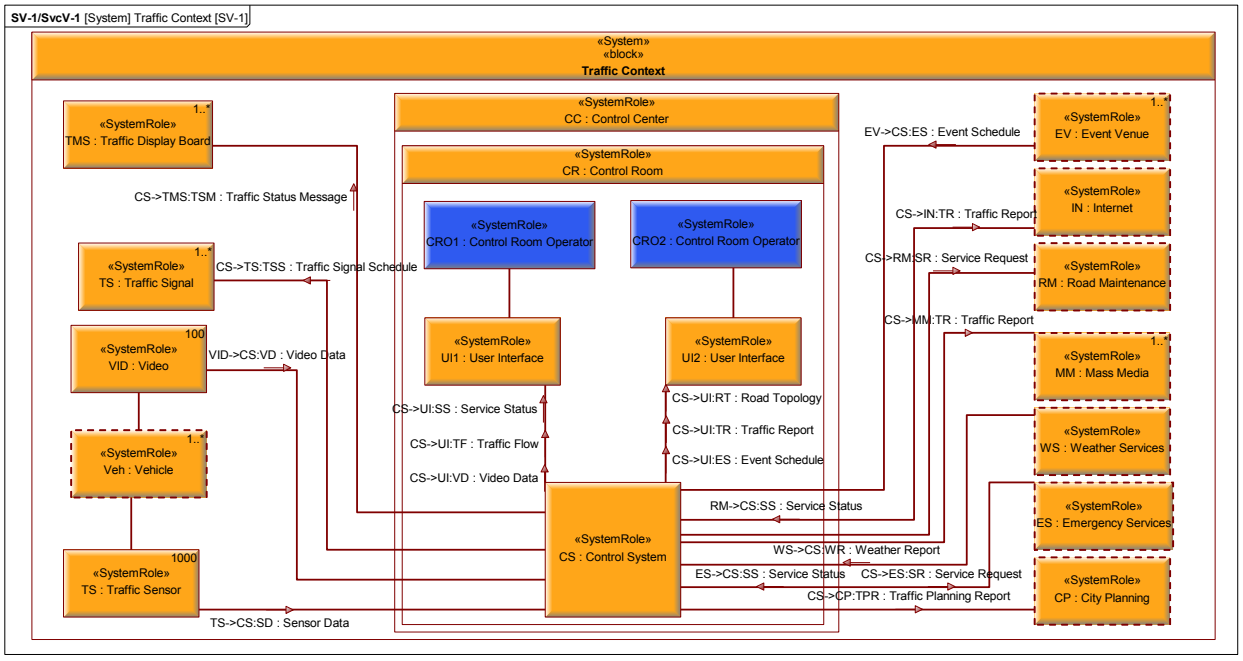

Figure 6. System Diagram of the Traffic Context

Even more detail can be given to show different pieces of data and its format that needs to be exchanged. At this point we can start to look at is the data being transferred via paper memos, over a network with automated reporting to other systems, via a telephone call between two individuals or a meeting to discuss that days information. We can start to look at best approaches to transfer information, and know what infrastructure may need to be purchased to achieve this. Figure 7 shows the software deployed on the control system.

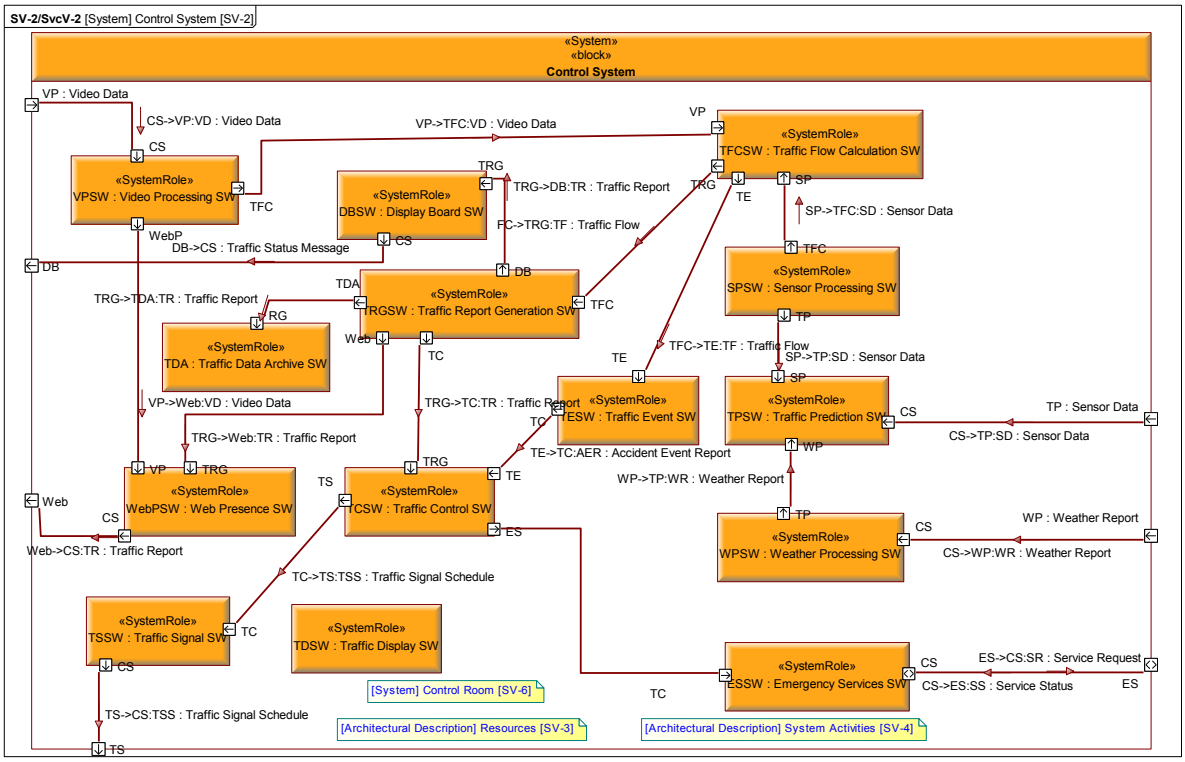

Figure 7. Traffic Management Software and their Interfaces

Figure 7 shows a number of software elements defined for the control system, their data interactions and external data requirements. For example, video data is received by the video processing software and send to the web presence and traffic flow calculation software, which also receives sensor data from the sensor processing software. Traffic flow calculations are sent to the traffic event software and the traffic report generation software, and so forth. The purpose of this diagram is not to design the software, but to define the requirements for the necessary software packages in terms of required data, interfaces, storage requirements, required behavior, etc. This can be used to specify which software packages will need to be purchased or developed. At this point we can go into a non-Smart City document their current flow and communication paths, perform a gap analysis and then re-design for efficiency to make the City Smarter.

Individual Systems View

Now let's just take one aspect of our Smart City and focus on its individual system. In this example we will look at a parking lot and how we can make it Smarter. Here we can see that we have three Parking Lot Systems that each communicates to three display boards. These display boards are placed throughout the city so people can view them and know where available spaces are located as shown in Figure 8.

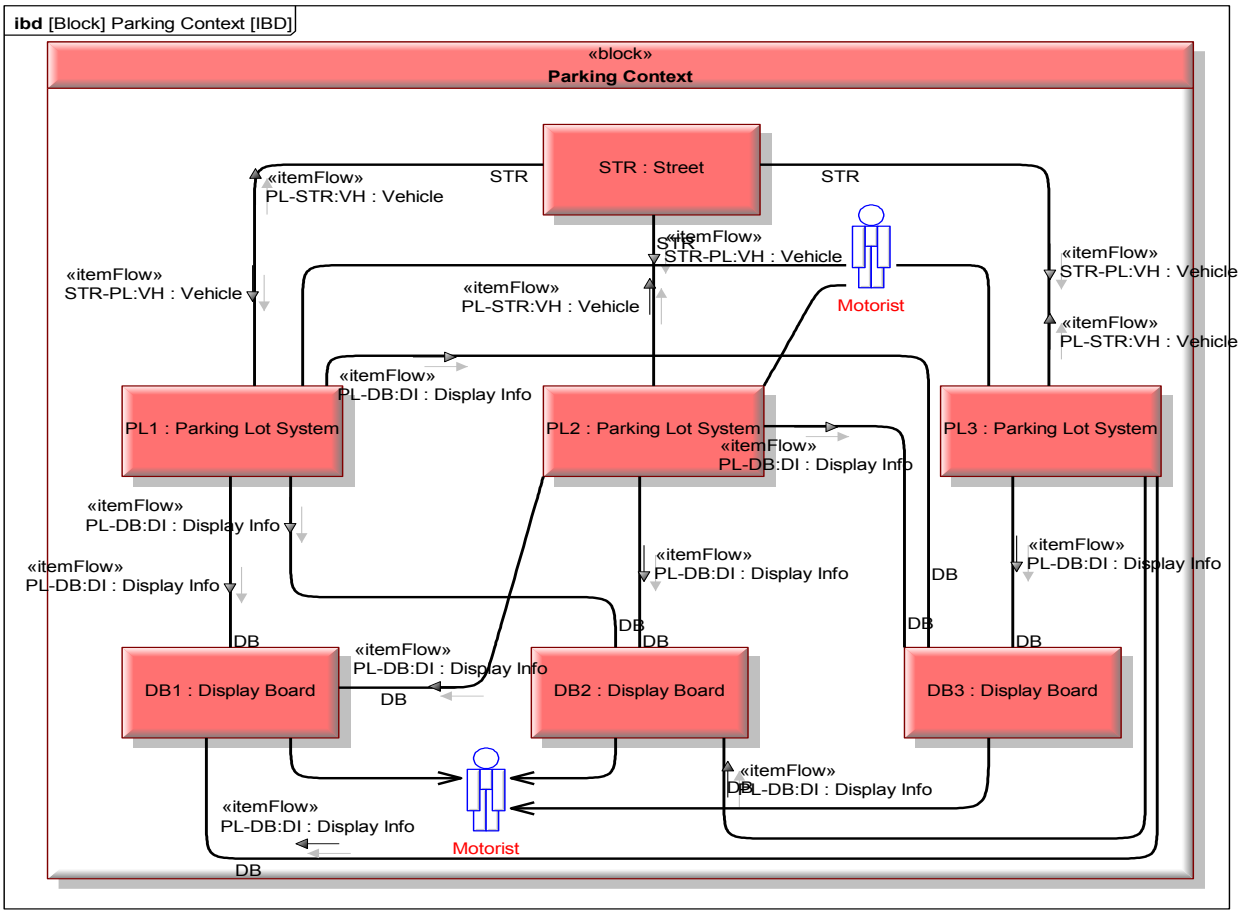

Figure 8. Parking Lot System Context

This diagram now shows us focusing on how may parking spaces we can have for a given parking lot. We can use these parts of the system in conjunction with parametrics to calculate how many spaces we can have. This also starts to address the need for electric car charging

stations. Current requirements are to calculate available parking. To support electric vehicles, we will also need to indicate the number of available charging spaces.

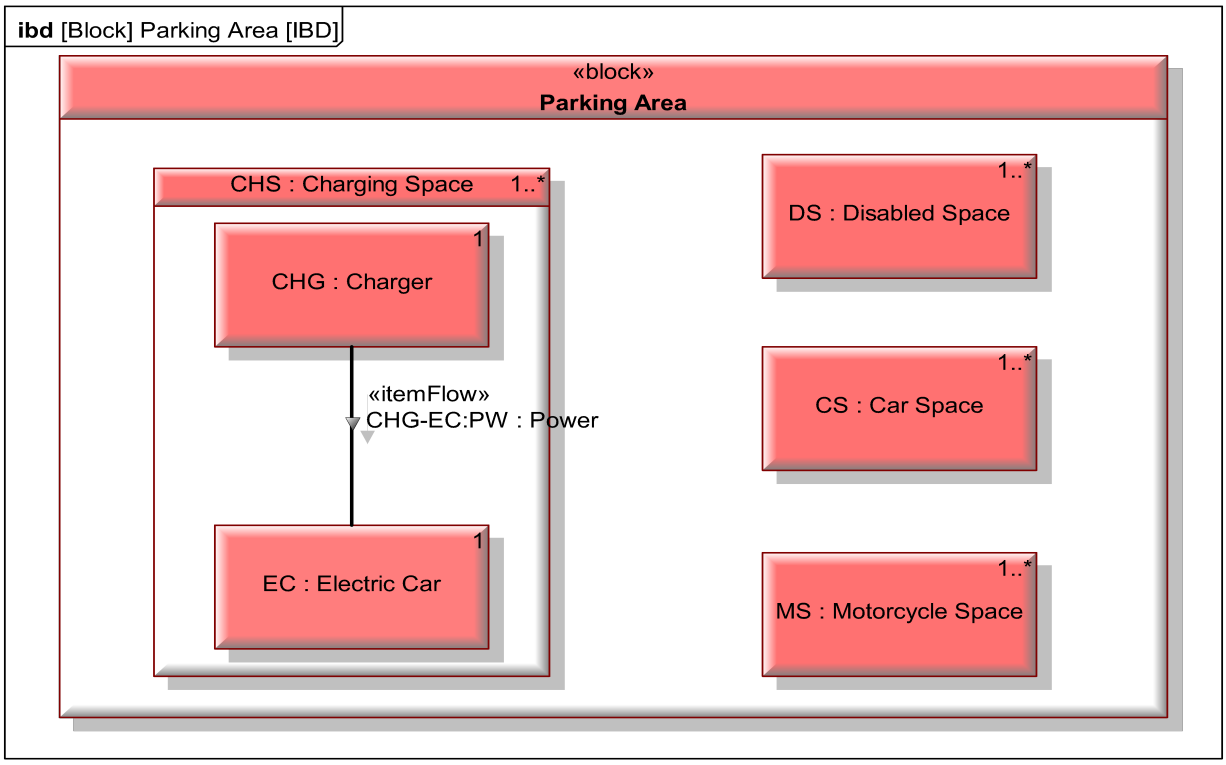

Figure 9. Parking Space Elements

SysML models can use model based generation capabilities to automate the generation from the embedded systems to capture data and send it onto an IoT platform.

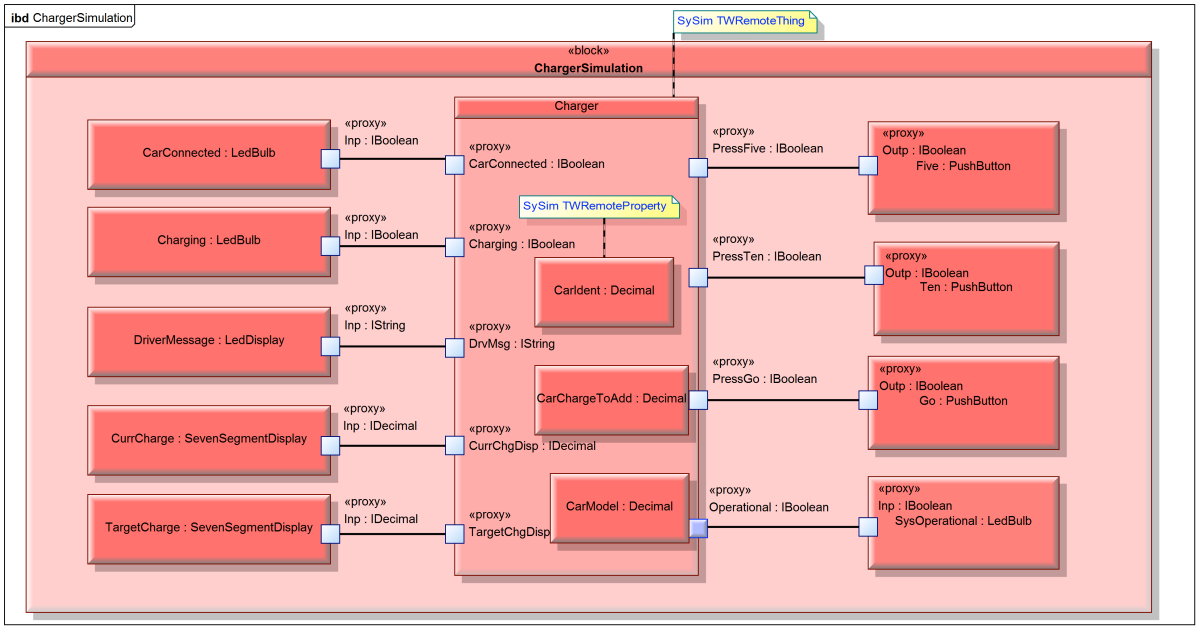

Figure 10. Design of a System where listening ports send on IoT information.

The example shown in Figure 11 is PTC ThingWorx. This not only makes it extremely easy to build up devices and control them by a click of a switch so we can generate all that is necessary for a device to send all of its data along to be read by a central office to manage parking spaces, collect revenue, and manage traffic flow. These systems can be co-simulated with varying levels of software in the loop, hardware in the loop, and system in the loop prior to deployment to determine whether or not the system is fit for purpose. This will save time and money and ensure that resources are spent efficiently.

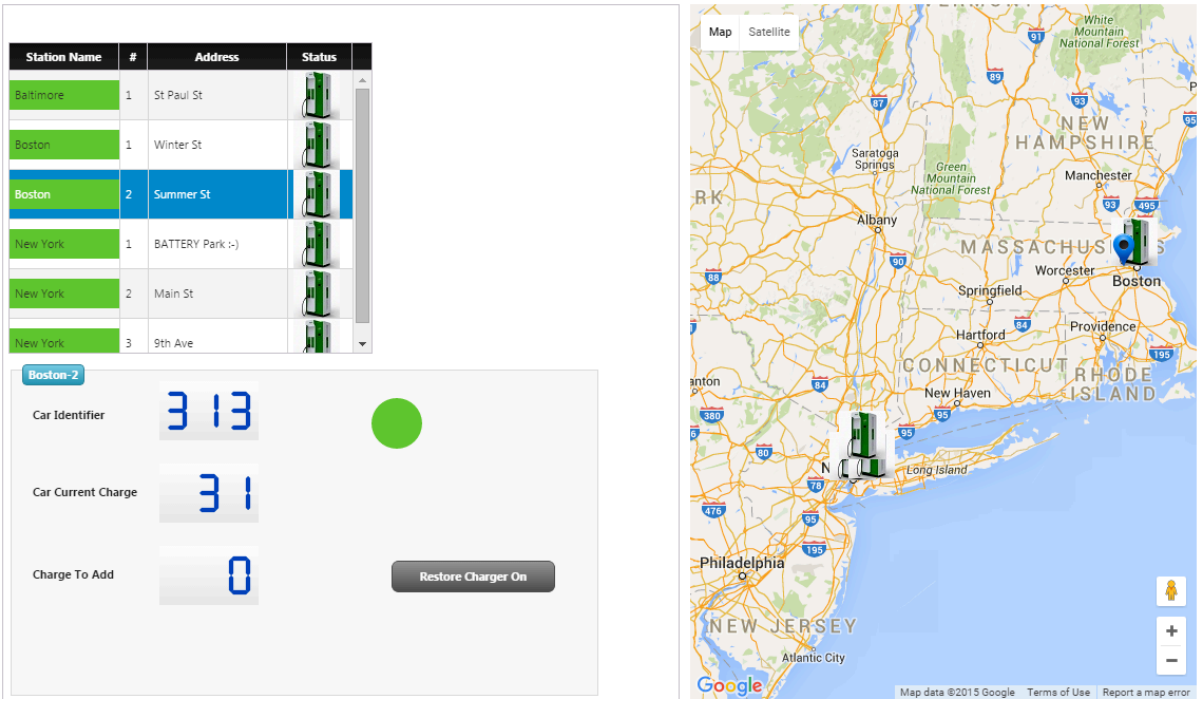

Figure 11. Dashboard view of the IoT device data.

We could also investigate creating traffic web applications, or adding interfaces to existing GPS navigation systems to support navigating to available free parking. This will cut down on driver frustration and ease traffic congestion.

Conclusion

The Internet of Things has the potential to transform the way we do business, work, live, play, and even exist. It is ubiquitous and growing exponentially. A system of systems of this size needs to be controlled, managed, guided, and above all planned and engineered. Without the necessary planning and engineering, it will result in insecure and potentially dangerous systems, and in the worst case scenario devolve into anarchy. Systems engineering has the ability to guide, engineer and plan these systems. MBSE coupled with IoT tools have the ability to implement and execute these designs in a clear documented form. Together, they have the potential to truly change the world by making it smarter, one system at a time.

References

[1] Goldman Sachs, “The Internet of Things: The Next Mega-Trend”, September 2014

[2]Wikipedia.com, biography for Kevin Ashton; http://en.wikipedia.org/wiki/Kevin_Ashton#cite_note-RFID_Journal_2009-3

[3] Kevin Ashton, “That ‘Internet of Things’ Thing”, RFID Journal, 22 June 2009

[4] Weilkiens, T. Systems Engineering with SysML/UML: Modeling, Analysis, Design, MK/OMG Press, Feb. 12, 2008

[5] Southampton City Council. "SmartCities card". Retrieved2015-05-30.

[6] Amsterdam Smart City. "Amsterdam Smart City ~ Projects". Retrieved 2015-05-30.

[7] Solanas, A.; Patsakis, C.; Conti, M.; Vlachos, I.; Ramos, V.; Falcone, F.; Postolache, O.; Perez-Martinez, P.; Pietro, R.; Perrea, D.; Martinez-Balleste, A. (2014). "Smart health: A context-aware health paradigm within smart cities". IEEE Communications Magazine 52 (8): 74. doi:10.1109/MCOM.2014.6871673.

[8] Komninos, Nicos (2013-08-22). "What makes cities intelligent?". In Deakin, Mark. Smart Cities: Governing, Modelling and Analysing the Transition. Taylor and Francis. p. 77. ISBN 978-1135124144.

[9] INCOSE Agile Working Group Mission and Objectives Page, http://www.incose.org/ChaptersGroups/WorkingGroups/knowledge/agile-systems-se

[10] INCOSE Vision Document, http://www.incose.org/AboutSE/sevision

[11] Porter, Heppleman, How Smart, Connected Products Are Transforming Competition, https://hbr.org/2014/11/how-smart-connected-products-are-transforming-competition

[12] Porter, Heppleman, How Smart, Connected Products Are Transforming Companies, https://hbr.org/2015/10/how-smart-connected-products-are-transforming-companies

[13] Stout, T. M. and Williams, T. J. (1995). "Pioneering Work in the Field of Computer Process Control". IEEE Annals of the History of Computing

[14] INCOSE Critical Infrastructure Protection and Recovery (CIPR) Working Group (WG), http://www.incose.org/docs/default-source/wgcharters/critical-infrastructure.pdf?sfvrsn=6

[15] INFRAGARD, https://www.infragard.org/

[16] UK Data Protection Act, http://www.legislation.gov.uk/ukpga/1998/29/contents

Biography

Matthew Hause is an Engineering Fellow at PTC, the co-chair of the UPDM group and a member of the OMG SysML specification team. He has been developing multi-national complex systems for almost 35 years. He started out working in the power systems industry and has been involved in military command and control systems, process control, communications, SCADA, distributed control, and many other areas of technical and real-time systems. His roles have varied from project manager to developer. His role at PTC includes mentoring, sales presentations, standards development and training courses. He has written a series of white papers on architectural modeling, project management, systems engineering, model-based engineering, human factors, safety critical systems development, virtual team management, systems development, and software development with UML, SysML and Architectural Frameworks such as DoDAF and MODAF. He has been a regular presenter at INCOSE, the IEEE, BCS, the IET, the OMG, DoD Enterprise Architecture and many other conferences.

James Hummell is an MBSE consultant at MBSE.Solutions. He is an expert in software and systems engineering, specializing in modeling and simulation analysis using UML and SysML, with extensive experience in embedded systems for safety critical systems (Do178b Level A), Configuration Management, the Software Life Cycle, and Process Engineering Development. He has been developing software and systems in model based design engineering (UML and SysML) for over 19 years. A member of the RTCA SC-205 subgroup developing Do-178C Model-Based Development and Verification Supplement. Working with OMG and INCOSE on many specifications and working groups. Has a degree in Computer Engineering from the University of Arizona.