An IoT platform for civil infrastructure monitoring

Seongwoon Jeong and Kincho H. Law
Engineering Informatics Group
Department of Civil and Environmental Engineering
Stanford University
Stanford, CA, USA
{swjeong3, law}@stanford.edu

Abstract— **IoT technology can have a huge impact in engineering by leveraging state-of-the-art information and communication technologies (ICT). In practice, however, it is challenging for IoT platforms to handle domain-specific engineering information (e.g., geometric model, engineering simulation model, etc.) along with sensor data of different types. Engineering information and sensor data need to be integrated for effective data utilization and shared and interoperable among a wide variety of software tools. This paper describes an IoT platform that is tailored to engineering applications and adopts an information modeling approach to facilitate data interoperability and to integrate engineering information with sensor data. In addition, a decentralized data management framework is employed so that the data owned by different project participants can be shared among authorized users and software agents. The IoT platform is demonstrated using a civil infrastructure monitoring scenario which involves various types of sensor data, as well as engineering models. The result shows that the IoT platform can facilitate information sharing and data utilization, in particular, for the civil infrastructure monitoring application.**

Keywords- Internet of Things, IoT platform, cloud computing, distributed data management, information modeling, interoperability, data integration

I. INTRODUCTION

With the advances in information and communication technology (ICT), sensors have been deployed widely in many engineering domains. The increasing use of sensors will realize the concept of Internet of Things (IoT) and cyber-physical system (CPS) that physical systems and computational systems are tightly integrated [1]. Physical systems can be monitored, analyzed and controlled with or without human intervention. Furthermore, the massive data collected by sensors offers promising opportunities to find new insights about the physical systems. In practice, however, sensor data needs to be integrated with domain-specific engineering information to support decision-making. A data management platform that can effectively manage sensor data and engineering information is essential before IoT technology can find useful in engineering applications.

An IoT platform refers to a system that can connect physical objects via a network, and receive, manage, store and analyze data generated from the physical objects. As IoT technology matures and becomes increasingly prevalent, many generic IoT platforms have been developed. However, there is no one-size-fit-all platform: an IoT platform has to be tailored to meet domain-specific requirements [2]. Engineering applications, which often involve diverse types of information ranging from heterogeneous sensor data (e.g., high-frequency time-series data and video and camera images) to domain-specific engineering information (e.g., geometric model, engineering simulation model, etc.) [3, 4], impose additional requirements for supporting data and software interoperability. First, engineering information needs to be integrated with sensor data to enable effective data retrieval and utilization. In addition, efficient information sharing and data exchange are required because engineering projects typically involve a wide variety of software tools, as well as *ad hoc* analysis modules [5]. Lastly, engineering applications often involve multiple parties, each of which may own certain types of data. An IoT platform needs to be designed to meet these requirements in order to effectively support engineering IoT applications.

In engineering domains, information modeling has gained enormous attention as a vehicle to support integrated project delivery process. Information modeling enables information sharing and integration, as well as seamless data exchange among software agents based on interoperability standards. However, information modeling community seldom pays attention to issues involving scalable sensor data management or standardized communications with client devices, which are being emphasized in most generic IoT platforms. This study illustrates that information modeling and IoT platforms can beneficially complement each other for effective data management in IoT-driven engineering applications.

In this paper, an IoT platform tailored to engineering applications is presented. For data integration and interoperability, an information model that defines sensor entities and their relationship to the engineering model is employed. Database schema and web interfaces are designed to support the management and sharing of semi-structured information model, as well as the large amount of sensor data. In addition, a decentralized data management approach is adopted based on a hybrid cloud computing environment, so that the data owned by different parties can be shared. The proposed IoT platform is demonstrated using a civil infrastructure monitoring scenario which involves various types of sensor data as well as engineering models.

II. Related Works

With the increasing adoption of sensor and IoT technology, many IoT platforms have been developed. Many cloud computing service vendors offer IoT platforms, including AWS IoT by Amazon AWS [6], IoT Hub by Microsoft Azure [7], Watson IoT Platform by IBM cloud [8], AT&T IoT Platform by AT&T [9]. These generic IoT platforms support device connectivity via standard protocols with high scalability by leveraging cloud computing technology. These platforms also provide many tools, such as device management tools, rule engine, event processing module, security tool and software development kit (SDK). While these generic IoT platforms provide basic services, they lack the supporting services for domain-specific applications and data management tools. Instead, these services need to be developed and added by customers or partner companies. There have been some domain-driven IoT platforms, such as PTC ThingWorx [10] and AutoDesk Fusion Connect [11] developed for industrial IoT (IIoT). These IIoT platforms offer some industrial applications and sophisticated functions, such as augmented reality (AR)-enabled user interfaces. Nevertheless, these IIoT platforms are not designed to manage engineering information models and do not support data and software interoperability.

Many research efforts have been spent on the development of IoT platforms for specific application areas, such as healthcare [12, 13], smart city [14, 15, 16] and agriculture [17]. Domain-specific platform handles not only sensor data, but also other relevant information. For example, Lea and Blackstock [15] describe an IoT platform for smart city application to manage a wide array of data, from real-time (e.g. traffic data) to static data (e.g., asset lists). However, this work does not address data integration for linking heterogeneous sensor data and domain information. To allow software agents to easily discover relevant information and to perform analysis, domain information and sensor data need to be properly linked and integrated. Jayaraman *et al.* [17] describe a semantic-driven IoT platform to link sensor data and domain concept based on ontology definitions. In addition, an IoT platform proposed in [14] enables interoperability among heterogeneous information models, such as building information models (BIM) and system information models (SIM), based on sematic web technology. Nevertheless, these studies do not address the data interoperability problem which is critical in engineering projects involving various software tools, each of which may have its own interface and data model. Engineering information needs to be exposed in a platform-neutral and standardized data format that can be easily parsed and used by different software agents, ranging from engineering simulation tools to data-driven analysis modules. To this end, this paper describes an IoT platform based on information modeling to deal with both data integration and data interoperability problem. The IoT platform's components are designed to support the sharing and management of information model data along with sensor data.

III. Overview of IoT Platform

Fig. 1 shows the architecture of the proposed IoT platform designed to manage data collected from various data sources in engineering, including information models and sensor networks. Information models include comprehensive information (e.g., geometry, physical properties, functional characteristics and sensor information) of target systems. Sensor networks, on the other hand, generate heterogeneous sensor data, ranging from high-frequency time series data to video and camera images.

The IoT platform is composed of three basic layers, namely, communication layer, mapping layer and storage layer, to support data store and retrieval. For data store processes, the communication layer handles communication with the data sources. Specifically, the web server and message broker in the communication layer provide

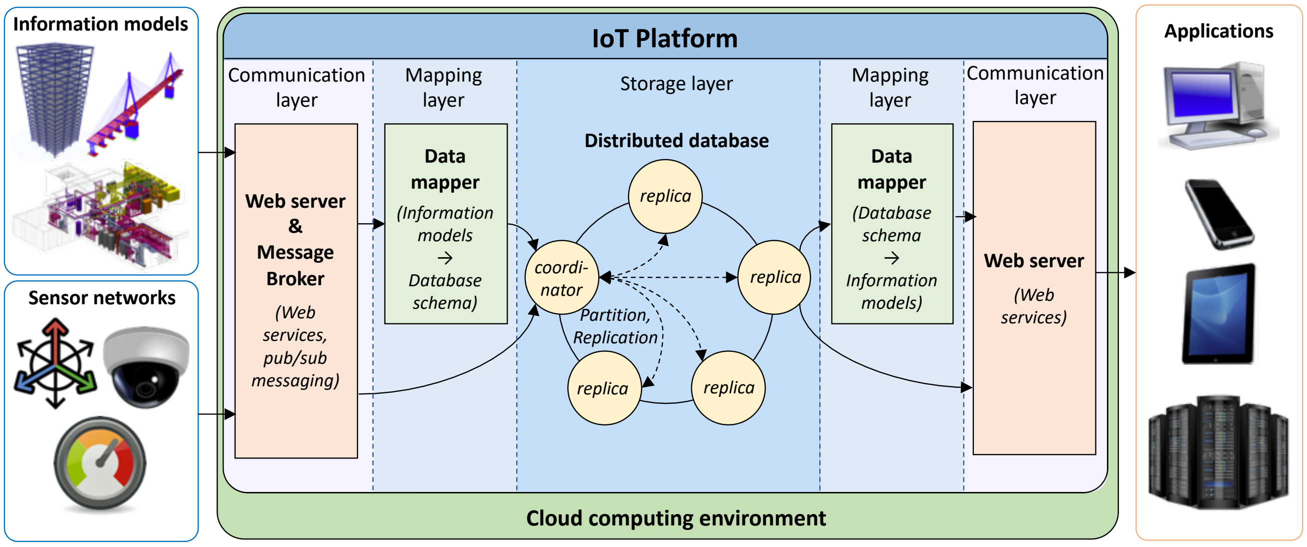

Figure 1. Overall architecture of the IoT platform for engineering

standardized interfaces to receive data from different data sources via the Internet. The mapping layer includes a data mapper that maps the received semi-structured information models onto the database schema. The mapped data is passed to the storage layer which includes a distributed database system that partitions, replicates and stores data.

The data stored in the distributed database of the IoT platform can be accessed and queried via user interface and by various applications, such as data analysis tools, engineering analysis software and 3-D modeling tools. For the data retrieval process, the web server in the communication layer provides standardized web interface through which applications can retrieve data. Receiving a request from an application, the web server retrieves data from the distributed database. If necessary, the data mapper is invoked to map the retrieved data back into information models' standardized data schema that can be parsed and utilized by different applications. Finally, the retrieved data is then delivered to the application.

The proposed IoT platform is deployed on a cloud computing environment for scalability, accessibility and reliability. Specifically, the IoT platform can be deployed on the Infrastructure as a Service (IaaS) layer of cloud (i.e., virtual machines offered by cloud) [18], which assures platform portability across different cloud vendors or among public cloud, private cloud and hybrid cloud.

IV. Data Store Process

This section describes the process of acquiring and storing the data with the IoT platform. Data store requests are processed through three layers: communication, mapping and storage. The communication layer employs web server and message broker built upon standard protocols, so that various data sources can access the platform. The mapping layer defines the mapper to help store the semi-structured information models in the database. Finally, the storage layer employs a distributed NoSQL database to enable scalable data management.

A. Communication Layer

The communication layer is exposed to clients (e.g., data sources) via the Internet to enable remote access. The layer serves as intermediary, and accepts messages with the data from the clients, parses the message to extract the data and passes the data to the appropriate layer. To support different communication protocols often used in IoT applications, this layer includes two systems: a web server based on the Hypertext Transfer Protocol (HTTP) and a message broker based on the Message Queuing Telemetry Transport (MQTT). In the prototype implementation, both systems are deployed using Node.js [19], a server-side JavaScript runtime environment.

The web server is implemented to support a machine-to-machine (M2M) communication based on the client-server model. Specifically, the web server hosts RESTful (Representational state transfer) web services that clients (i.e., data sources) can invoke to deliver messages via HTTP [20]. The proposed web server hosts a set of web services to handle different types of data. For example, a web service for storing an information model works as follows. A client system, such as a local desktop computer, sends an HTTP request specifying the host name and the Uniform Resource Identifier (URI) of the web service for information model store. In addition, the HTTP request includes an information model file written in standard syntax such as eXtensible Markup Language (XML) that the client wants to store in the IoT platform. Upon receiving the message, the web server calls the web service corresponding to the request specified. The web service then parses the HTTP request, extracts the information model file, and delivers the file to the next layer (i.e., mapping layer). Once the process is completed, the web service returns an HTTP response notifying that the request has been processed successfully. If any error occurs during the process, the web service returns an HTTP response with an error message.

The web server processes a sensor data store request in a similar manner. For example, a gateway system connected to a sensor network sends an HTTP request that contains sensor data written in JSON format, as well as the host address and the URI for the sensor data store service. Once the web server receives the request, a web service is invoked, parses the request and extracts sensor data. It should be noted that the sensor data is delivered directly to the storage layer without going through the mapping layer in the current implementation, since sensor data typically does not require complex data mapping.

The message broker is implemented to support M2M communication for sensor data exchange based on the publish/subscribe (pub/sub) paradigm. Publisher clients (e.g., data sources) and subscriber clients (e.g., application programs) can exchange messages in one-to-one, one-to-many and many-to-many communication in real time. This real time messaging can be used not only to store data, but also to enable tasks, such as real time analytics and event triggering, executed in real time. The message broker manages several "topics", each of which is defined for a specific type of sensor data. Data sources can use the topic to specify the type of data. For example, a gateway device connected to sensor network can publish a message containing the sensor data written in JSON and a topic name to the message broker via MQTT protocol. Once published, the message is parsed, and the sensor data is delivered to the storage layer. In addition, any subscriber clients subscribing to the same topic can receive the published message via MQTT protocol.

B. Mapping Layer

The mapping layer receives the information models from the communication layer, maps the model to a database schema, and loads the mapped data to the database. Information models are typically written in object-oriented manner to represent a system as a set of hierarchical objects. Each object includes information about its characteristics, such as physical properties, functional role and relationship with other objects. For interoperability, information models are often written with XML or XML-based syntax adopted by information modeling standards. For the mapping of such information models, the mapping layer includes a data mapper that has information about the relation between the information model schema and the database schema. In the prototype implementation, a Python script is written as a data mapper which can be called by a web service in the communication layer. The data mapper uses an XML parser (e.g., Python xml.etree.ElementTree package [21]) to parse an XML-based information model and a database driver (e.g., Cassandra driver API [22]) to transmit the mapped data to the storage layer.

The data mapper works as follows. Upon receiving an information model, the data mapper is invoked. The data mapper first parses the model written in XML into a hierarchical object structure using the XML parser. Each object in the structure includes information about its properties as well as hierarchical relationship (e.g., parent and child objects information). The mapping script accesses the root object and maps the root object into the database schema. The data mapper then creates an INSERT query request for the mapped object, and delivers the query request to the next layer (i.e., storage layer) using the database driver. This process is conducted recursively for the child objects in the hierarchical object structure until all the leaf node objects are processed.

C. *Storage Layer*

The storage layer provides a data management service implemented using a distributed database. This layer receives information model data (from the mapping layer) and sensor data (from the communication layer), and stores them in the database. For the effective data management, it is critical to choose an appropriate database management system (DBMS). Since the IoT platform aims to manage a large volume of sensor data and engineering information model, scalability is an important factor for choosing a DBMS. The prototype implementation employs Apache Cassandra [23] which is a distributed NoSQL database widely adopted for large-scale distributed data management. Based on the peer-to-peer (P2P) architecture, Cassandra provides high availability and scalability [24]. For example, Cassandra ensures that in the worst-case scenario, the failure of some nodes results in degradation of the database performance but remains able to guarantee a high possibility of availability. Furthermore, a Cassandra database cluster is easily scalable such that the capacity of the system scales linearly as new nodes are added to the cluster.

A Cassandra database cluster is composed of multiple nodes, each of which can be employed on a physical or virtual machine. Cassandra glues the nodes distributed over multiple machines. Furthermore, Cassandra handles data partitioning and replications over multiple nodes according to the defined network topology and replication factor. For instance, a Cassandra database cluster, which has replication factor of two, partitions incoming data into multiple pieces and stores them twice over the distributed database nodes. With partitioning and replication, a Cassandra database cluster can be available even when some of the nodes are down.

Cassandra's data model consists of keyspaces, column families, rows and columns, which are analogous to the common definitions, such as databases, tables, tuples and attributes, respectively, of relational database model [24]. The column-oriented data model of Cassandra offers some advantages for handling object-oriented information model data and the large volume of sensor (in particular, time-series) data. For example, Cassandra data schema is flexible in that every row can have a different set of columns. This flexible data schema is effective when storing information model data, since objects in an information model may have different sets of attributes.

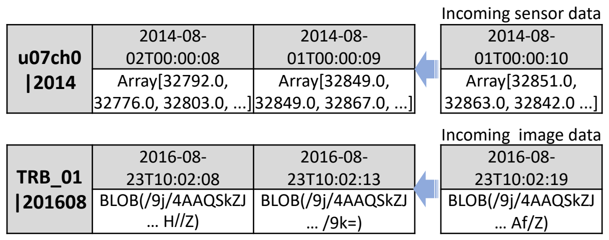
Incoming sensor data
u07ch0 |2014
2014-08-02T00:00:08
Array[32792.0, 32776.0, 32803.0, ...]
2014-08-01T00:00:09
Array[32849.0, 32849.0, 32867.0, ...]
2014-08-01T00:00:10
Array[32851.0, 32863.0, 32842.0 ...]
Incoming image data
TRB_01 |201608
2016-08-23T10:02:08
BLOB(/9j/4AAQSkZJ ... H//Z)
2016-08-23T10:02:13
BLOB(/9j/4AAQSkZJ ... /9k=)
2016-08-23T10:02:19
BLOB(/9j/4AAQSkZJ ... Af/Z)

Figure 2. Data schema definition for high-frequency sensor data (top) and image data (bottom)

Cassandra's data model also improves the performance of range query, which is often used in retrieving time-series sensor data, based on clustering and dynamic column features [25]. For example, time-series sensor data collected by a sensor can be stored in consecutive columns of a single row in a sorted order by assigning the timestamp of the sensor data as the clustering key. The data schema also allows new data collected by the same sensor to be dynamically appended to the end of the row. In this way, the data schema can guarantee high query speed by storing consecutive time-series sensor data in contiguous physical disk locations in a node. Fig. 2 shows the data schema examples for the sensor and image data. Since it is redundant to store every timestamp for the high-frequency data with the same sampling rate, the data schema is defined to store data collected in a certain period of time (e.g., 1 second) in a single column as a numeric array data type. The data schema for image data stores a single image per column in Binary Large Object (BLOB) data type. For both cases, data collected by the same sensor is stored in a single row in a sorted order.

V. Data Retrieval Process

This section describes data retrieval process with the proposed IoT platform. Standardized interfaces are provided so that applications on various systems and devices can access and retrieve data. In a data retrieval process, a request is delivered from an application to the communication layer, to the mapping layer and to the storage layer, whereas data is delivered in reverse order. The retrieved data enables interoperability and integration based on information modeling standards and ontology.

A. *Data Retrieval*

The communication layer handles the communication required by the applications. The communication layer includes a web server that provides RESTful web services for applications to retrieve the data, including (partial and entire) information models and heterogeneous sensor data. For example, an engineering analysis application can download an analysis model, which is a part of an information model, as follows. The application issues an HTTP request which specifies the host name, the URI of the web service for analysis model retrieval and the identifier of the target system. The request is delivered to the web server and the corresponding web service is invoked. The web service then passes the request to the mapping layer. The mapping layer is responsible to query all the relevant objects from the storage layer and to map the objects into a hierarchical model. For this task, the data mapper in the mapping layer first retrieves the root object of the requested model by sending a SELECT query to the distributed database in the storage layer. Once the object is returned, the mapper maps the object from the database schema to the information model schema. In addition, the mapper performs data retrievals for the child objects of the root object using the child object information recorded in the object data. The retrieved child objects are then parsed and added to the root object. The mapper performs this process recursively until there is no further child objects. The retrieved hierarchically structured objects are then saved as an XML file and returned to the communication layer, which then returns an HTTP response containing the file to the application.

Similarly, an application can retrieve sensor data by sending an HTTP request which specifies the host name, the URI for sensor data retrieval web service and the parameters (e.g., sensor ID and target time period) to the web server. For sensor data retrieval request, the corresponding web service sends a SELECT query directly to the data storage, since the retrieved sensor data does not require complex data mapping. The retrieved data is mapped to JSON format by the web service, which then returns an HTTP response containing the data to the application.

B. *Data Interoperability and Integration*

One important feature of the proposed IoT platform offers is the data interoperability. Information models managed by the platform is retrieved as a file written in a platform-neural language, i.e., XML, based on information modeling standards. Therefore, the retrieved model can be easily parsed and converted to other data models for different application software tools. In addition, sensor data is retrieved using another platform-neural format, i.e., JSON, so that the data can be utilized by most computing systems.

Since the proposed IoT platform manages information models and sensor data together, integrated use of data can be facilitated based on ontology or data semantics. For example, in an information model, a sensor object can be assigned to a system component object via a relationship, such as "hasSensor". In addition, sensor data can be assigned to a sensor object via a relationship, such as "measures". These relationships can be used to relate the system components and sensor data and to enable integrated use of information model and sensor data.

VI. Cloud-Based Implementation

This section describes the cloud-based implementation of the proposed IoT platform. The IoT platform needs to be scalable to handle large and increasing amount of sensor data. The use of cloud computing enables scalability of the IoT platform. The prototype implementation leverages Infrastructure as a Service (IaaS) of cloud computing to enable portability. In addition, a hybrid cloud-based decentralized data management is discussed to facilitate information sharing.

A. *Cloud Computing Environment*

According to the National Institute of Standards and Technology (NIST) definition, cloud computing is a "model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal effort or service provider interaction [18]." Cloud computing provides many benefits, including lower upfront investment, lower operating cost, higher scalability, higher accessibility, lower maintenance expenses and reduced business risks as compared to the traditional server-based approaches [26, 27].

Cloud computing service models can be categorized into Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [18]. In the prototype implementation, the IoT platform serves as a PaaS that is built on the IaaS of cloud providers to provide scalable and reliable data management service. Since IaaS cloud service is vendor-independent, the proposed IoT platform can be deployed on any cloud computing environment. Furthermore, the platform can be migrated from one cloud vendor to another, from public cloud to private cloud and vice versa.

IaaS cloud service is often provided as virtual machines (VMs), which can be easily created, configured and managed through cloud service interfaces. The use of IaaS cloud service ensures high scalability, in terms of vertical scaling and horizontal scaling. Specifically, cloud computing offers almost unlimited the horizontal scalability (e.g., by increasing the number of VMs). To leverage the high scalability of cloud computing, the IoT platform is designed to have the ability to run on a distributed computing environment where the number of VMs can be dynamically increased. For example, the platform employs a distributed NoSQL database that is linearly scaled as VMs are added to the platform [24].

B. *Hybrid Cloud-based Decentralized Data Management*

In engineering practice, different types of data are owned by different project participants. Some data owners may hesitate to upload sensitive data to a public cloud computing platform because of security concern; instead, the data owners prefer to store the data in a fully-controlled server. Nevertheless, data needs to be shared among authorized users and application software.

Hybrid cloud, which combines public cloud and private cloud, could be a desirable preference for IoT platform in engineering [28]. Fig. 3 depicts the conceptual framework of the hybrid cloud-based implementation of IoT platform. The public cloud system is employed for the management of voluminous sensor data which requires high scalability and availability. The private cloud, on the other hand, plays a role to handle sensitive data, such as engineering models.

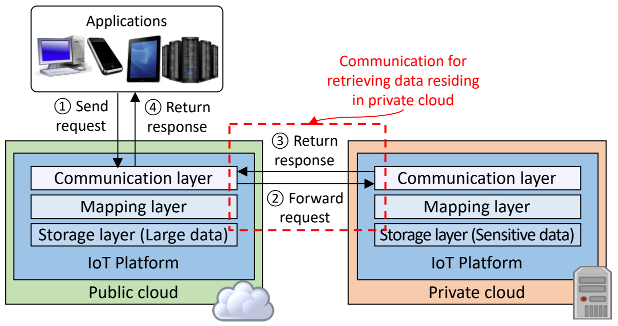

Figure 3. A framework of hybrid cloud-based implementation of IoT platform for decentralized data management

These two separate systems can communicate with each other through web services. For example, when an application program requests sensor data retrieval from the public cloud, the data is directly retrieved from the database on public cloud. On the other hand, when an application program requests for an engineering model from the public cloud, the public cloud forwards the request to the private cloud. The private cloud then retrieves corresponding information from its database and deliver the information to the client via the public cloud. In this way, the hybrid cloud system can abstract the underlying complex structure and provides unified web services to clients.

VII. Case Scenario: Civil Infrastructure Monitoring Application

This section describes a case scenario using a civil infrastructure monitoring application as an example. For demonstration, the Telegraph Road Bridge (TRB), which is a 68-meter long highway overpass located at Monroe, Michigan, is chosen as a target system, as shown in Fig. 4. The TRB has been monitored with a sensor network installed and operated by a research team at University of Michigan since 2011 [29, 30]. The data involved in the TRB monitoring is as follows.

- Sensor data: The TRB is instrumented with 60 sensors, including accelerometers, strain gauges and thermistors. They collect data every two hours for one-minute duration at sampling frequency 200 Hz (accelerometer) or 100 Hz (strain gauges and thermistors).
- Traffic video image: The traffic monitoring system operated by Michigan Department of Transportation (MDOT) collects traffic video images at the TRB every two seconds [31].
- Geometric model: The TRB's geometric model is created for the representation of geometry and physical characteristics.
- Engineering model: An engineering model (i.e., a Finite Element model) is developed for numerical simulation of structural behavior of the bridge.
- Sensor information: The sensor information, including sensor ID, sampling rate, physical characteristic, electrical characteristics, etc., is recorded in a Microsoft Excel Spreadsheet.

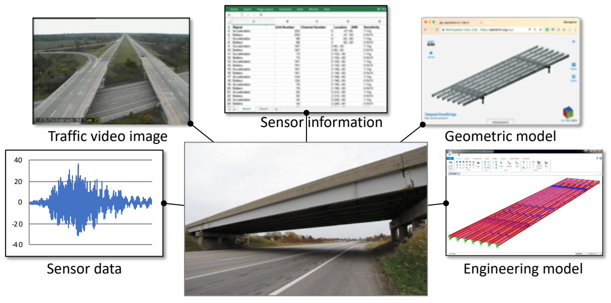

Figure 4. The Telegraph Road Bridge and data involved in the civil infrastructure monitoring application

A prototype IoT platform is implemented using the VMs provisioned on the Microsoft Azure cloud computing service, as well as a private server. Table 1 summarizes the list of computers composing the prototype IoT platform and their role.

A. *Storing Data in IoT Platform*

The IoT platform can store and manage comprehensive data involved in the bridge monitoring application. Fig. 5 depicts the data stored with the IoT platform. Dynamically collected data (e.g., sensor data and video images) are transmitted to the IoT platform continuously and automatically. For this purpose, automation scripts are developed. For example, a sensor data store script is deployed on the onsite computer, which receives data from the sensor network. The script detects new sensor data, parses the data into JSON format, and transmits the parsed data to the IoT platform via HTTP or MQTT. In addition, a video image store script is deployed on a local desktop computer. The script accesses the traffic monitoring system and downloads the image files from the real-time image stream. A downloaded image file is converted into byte array format and stored as a JSON document. The script then sends the JSON document to the IoT platform via HTTP or MQTT.

Static information (e.g., sensor information, geometric model and engineering model), on the other hand, needs to be mapped into the information model schema before storing. For prototyping, an XML-based information model schema for bridge monitoring application based on the OpenBrIM standards is employed [32, 33]. The geometric model is created using the OpenBrIM App [34], which is compatible with the information model schema, and thus, data mapping is not needed. The engineering model is created using CSI Bridge [35], a software tool for structural analysis, which can export the model in a Microsoft Excel Spreadsheet format.

TABLE I. Specification And Role Of Cloud Virtual Machines And Private Server Composing IoT Platform

Type	Spec	Quantity	Role
Public cloud VM	Azure Standard_A2m_v2 (2 cores, 16 GB memory)	5	Distributed database for large data
	Azure Standard DS2 v2 (2 cores, 7 GB memory)	3	Web server
Private server	Dell PowerEdge T620	1	Local data storage for sensitive data

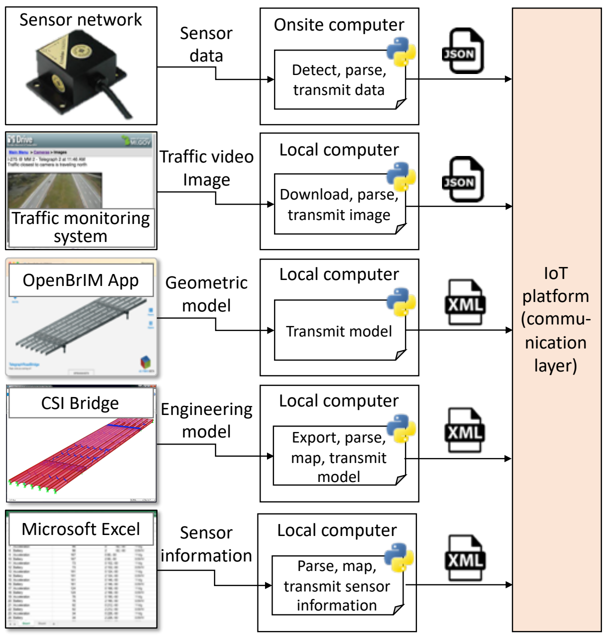

Figure 5. Storing heterogeneous data from different sources

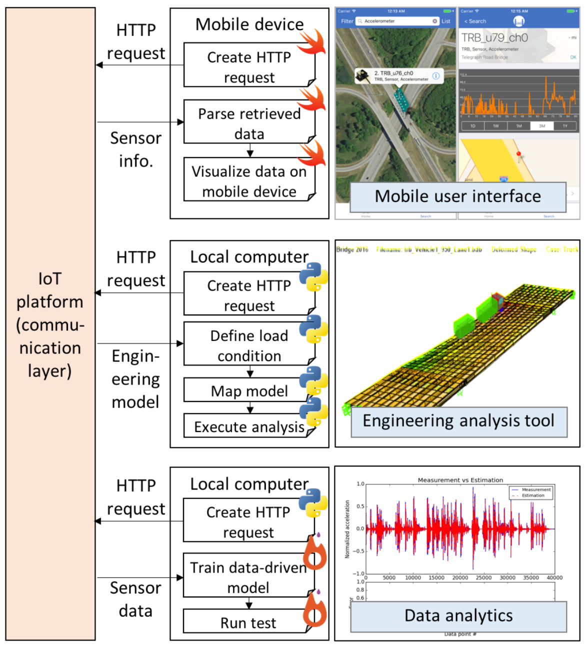

Figure 6. Retrieving heterogeneous data from different devices and applications

The exported file is then parsed and mapped into the information model schema, using an Excel data parser [36], XML parser [21] and a mapping script written for engineering model. Similarly, sensor information recorded in a Microsoft Excel Spreadsheet format is parsed and mapped into information model schema by using an Excel data parser, an XML parser, and a mapping script written for sensor information. The information converted into information model schema is then transmitted to the IoT platform via HTTP, and stored.

B. *Retrieving Data from Applications*

Since the IoT platform offers standardized web interfaces for data retrieval and returns data in a standardized information model schema, different applications and devices can access and retrieve data from the platform, which facilitates data utilization. Fig. 6 shows a few applications that utilize the data retrieval web services of the IoT platform. Firstly, the mobile and web user interfaces can easily retrieve the data from the IoT platform and display the data on mobile devices or web browsers. In addition, domain-specific engineering analysis software tools can also be connected to IoT platform using the web services and the Application Programming Interface (API) of the software tools. For example, Fig. 6 shows a numerical simulation performed using a script that downloads an information model from the IoT platform, defines truck loads, converts model to Microsoft Excel Spreadsheet format, and executes analysis using the API of the CSI Bridge software. Last but not least, a large amount of sensor data residing in the IoT platform can be retrieved and analyzed using data analytic tools. For example, Fig. 6 shows a sensor data reconstruction module triggered by a script that invokes the web service to retrieve accumulated sensor data, trains a data-driven model using a machine learning tool (e.g., PyTorch [37]) and reconstructs sensor data using the trained data-driven model.

C. *Integrated Use of Data*

The heterogeneous data managed by the IoT platform can be integrated via appropriate design of ontology schema. For example, sensor data and engineering model can be retrieved and compared based on the relationship defined between engineering model objects, sensor objects and sensor data. Fig. 7 shows a comparison between measured acceleration and numerically simulated acceleration at the same location of the TRB. The comparison shows that the physical system's response is within the range of numerically simulated response.

Heterogeneous sensor data can also be integrated to understand the target system's behavior. Fig. 8 shows a comparison between traffic video image and the corresponding (time synchronized) acceleration data. This

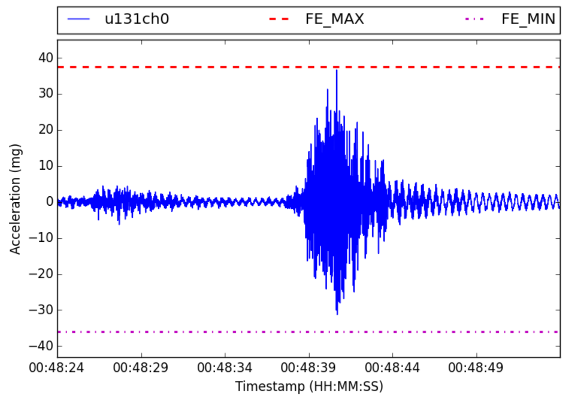

Figure 7. Integration of heterogeneous data: engineering model and acceleration measurement data

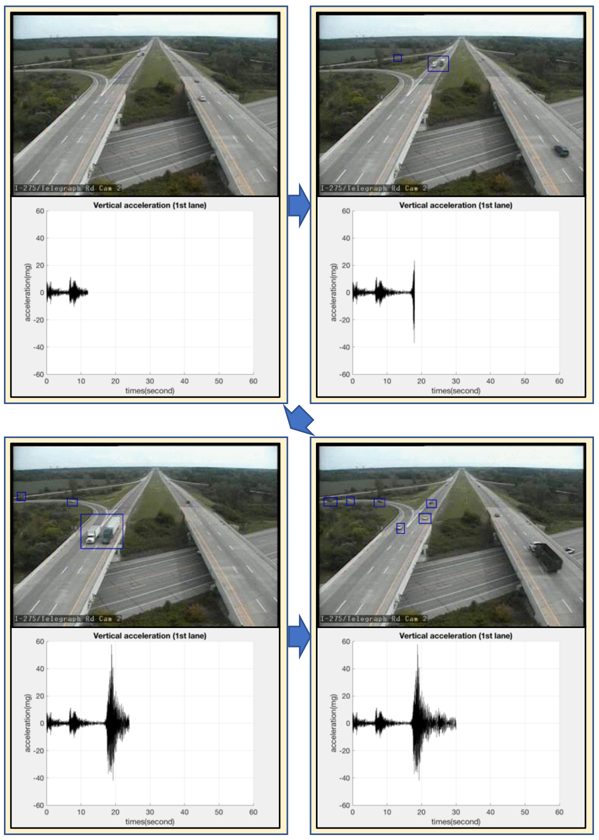

Figure 8. Integration of heterogeneous data: traffic video image data and acceleration measurement data

comparison shows, for example, the cause of high acceleration amplitude with a large truck passing the bridge.

VIII. Conclusion

This paper presents an IoT platform for engineering applications with emphasis on data integration and interoperability. To support various software tools involved in engineering projects, the IoT platform is designed to manage not only the heterogeneous sensor data, but also the standardized engineering information models. In addition, sensor data and domain-specific engineering information are linked and integrated based on relationship definitions among relevant data entities. The platform consists of three basic layers: communication layer, mapping layer and storage layer. The communication layer supports machine-to-machine communication based on standard protocols to allow accesses from different systems and different devices, including data sources and applications. The mapping layer is developed for the data mapping between standardized information model schema and database schema. The storage layer employs a distributed NoSQL database to enable scalable data management. For scalability, reliability and portability, the prototype IoT platform is implemented on the IaaS cloud service offered by a public cloud vendor. In addition, the platform supports hybrid cloud environments, so that the data owned by different parties can be shared in a decentralized manner. The proposed IoT platform is demonstrated with a case scenario of civil infrastructure monitoring, which involves various types of sensor data and engineering models. The result demonstrated that the IoT platform is able to manage heterogeneous data, allow access from various devices and systems, and facilitate data interoperability and integration based on engineering information modeling.

Acknowledgment

The research is supported by a collaborative project funded by the US National Science Foundation (Grant No. ECCS-1446330 to Stanford University and Grant No. CMMI-1362513 and ECCS-1446521 to the University of Michigan). This research is also supported by a Grant No. 13SCIPA01 from Smart Civil Infrastructure Research Program funded by Ministry of Land, Infrastructure and Transport (MOLIT) of Korea government and Korea Agency for Infrastructure Technology Advancement (KAIA). The authors would like to thank Prof. Jerome Lynch of University of Michigan and Prof. Hoon Sohn of KAIST, Korea for their collaboration. While certain commercial systems are identified in this paper, such identification does not imply recommendation or endorsement by the authors, NSF, MOLIT, or KAIA; nor does it imply that the products identified are necessarily the best available for the purpose. Furthermore, any opinions, findings, conclusions or recommendations expressed in this paper are solely those of the authors and do not necessarily reflect the views of NSF, MOLIT, KAIA or any other organizations and collaborators.

References

[1] R. Baheti and H. Gill, "Cyber-physical systems," The impact of control technology, vol. 12, pp. 161-166, 2011.

[2] F. Wortmann and K. Flüchter, "Internet of things," Business & Information Systems Engineering, vol. 57, no. 3, pp. 221-224, 2015.

[3] S. Kim, C.Y. Kim and J. Lee, "Monitoring Results of A Self-Anchored Suspension Bridge," in Sensing Issues in Civil Structural Health Monitoring, Dordrecht, Springer, 2005, pp. 475-484.

[4] R. Hou, S. Jeong, Y. Wang, K. Law and J. Lynch, "Camera-based Triggering of Bridge Structural Health Monitoring Systems using a Cyber-physical System Framework," in Structural Health Monitoring, 2017.

[5] C. Eastman, P. Teicholz, R. Sacks and K. Liston, BIM handbook: a guide to building information modeling for owners, managers, designers, engineers, and contractors, Hoboken, NJ: John Wiley & Sons, Inc., 2011.

[6] Amazon Web Services, "AWS IoT Services Overview - Amazon Web Services," [Online]. Available: https://aws.amazon.com/iot/. [Accessed 1 February 2018].

[7] Microsoft, "Azure IoT Hub | Microsoft Azure," [Online]. Available: https://azure.microsoft.com/. [Accessed 1 February 2018].

[8] IBM, "IoT Platform - IBM Watson IoT," [Online]. Available: https://www.ibm.com/internet-of-things/spotlight/watson-iot-platform. [Accessed 1 February 2018].

[9] AT&T, "AT&T IoT Platform - Build Solutions for the Internet of Things," [Online]. Available: https://iotplatform.att.com/. [Accessed 1 February 2018].

[10] PTC, "ThingWorx Industrial Innovation Platform | PTC," [Online]. Available: https://www.ptc.com/en/products/iot/thingworx-platform. [Accessed 1 February 2018].

[11] Autodesk, "Autodesk Fusion Connect. Enterprise IoT Software Platform," [Online]. Available: https://autodeskfusionconnect.com/. [Accessed 1 February 2018].

[12] G. Yang, L. Xie, M. Mäntysalo, X. Zhou, Z. Pang, L. Xu, S. Kao-Walter, Q. Chen and L. Zheng, "A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box," IEEE transactions on industrial informatics, vol. 10, no. 4, pp. 2180-2191, 2014.

[13] C. Doukas and I. Maglogiannis, "Bringing IoT and Cloud Computing towards Pervasive Healthcare," in 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2012.

[14] A. Krylovskiy, M. Jahn and E. Patti, "Designing a Smart City Internet of Things Platform with Microservice Architecture," in 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, 2015.

[15] R. Lea and M. Blackstock, "City hub: A cloud-based iot platform for smart cities," in 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, Singapore, 2014.

[16] G. Suciu, A. Vulpe, S. Halunga, O. Fratu, G. Todoran and V. Suciu, "Smart Cities Built on Resilient Cloud Computing and Secure Internet of Things," in 2013 19th International Conference on Control Systems and Computer Science, Bucharest, 2013.

[17] P. Jayaraman, D. Palmer, A. Zaslavsky and D. Georgakopoulos, "Do-it-Yourself Digital Agriculture applications with semantically enhanced IoT platform," in 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2015.

[18] P. Mell and T. Grance, The NIST definition of cloud computing, 2011.

[19] Node.js Foundation, "Node.js," [Online]. Available: https://nodejs.org/. [Accessed 1 February 2018].

[20] R. Fielding, Architectural styles and the design of network-based software architectures, Doctoral dissertation, University of California, Irvine., 2000.

[21] Python Software Foundation, "19.7. xml.etree.ElementTree - The ElementTree XML API," [Online]. Available: https://docs.python.org/2/library/xml.etree.elementtree.html. [Accessed 1 February 2018].

[22] DataStax, "Python Cassandra Driver - Cassandra Driver 3.13.0 documentation," [Online]. Available: http://datastax.github.io/python-driver/api/index.html. [Accessed 1 February 2018].

[23] The Apache Software Foundation, "Apache Cassandra," [Online]. Available: http://cassandra.apache.org/. [Accessed 1 February 2018].

[24] E. Hewitt, Cassandra: the definitive guide, O'Reilly Media, Inc., 2010.

[25] T. Le, S. Kim, M. Nguyen, D. Kim, S. Shin, K. Lee and R. da Rosa Righi, "EPC information services with No-SQL datastore for the Internet of Things," in 2014 IEEE International Conference on RFID (IEEE RFID), Orlando, FL, 2014.

[26] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of Internet Services and Applications, vol. 1, no. 1, pp. 7-18, 2010.

[27] A. Zaslavsky, C. Perera and D. Georgakopoulos, "Sensing as a service and big data," in International Conference on Advances in Cloud Computing (ACC-2012), Bangalore, 2013.

[28] X. Huang and X. Du, "Efficiently secure data privacy on hybrid cloud," in 2013 IEEE International Conference on Communications (ICC), Budapest, 2013.

[29] S. O'Connor, J. Lynch, M. Ettouney, G. vander Linden and S. Alampalli, "Cyber-Enabled Decision Making System for Bridge Management Using Wireless Monitoring Systems: Telegraph Road Bridge Demonstration Project," in Structural Materials Technology 2012, 2012.

[30] S. O'Connor, Y. Zhang, J. Lynch, M. Ettouney and P. Jansson, "Long-term performance assessment of the Telegraph Road Bridge using a permanent wireless monitoring system and automated statistical process control analytics," Structure and Infrastructure Engineering, vol. 13, no. 5, pp. 604-624, 2017.

[31] State of Michigan, "MDOT - Mi Drive Interactive Map," [Online]. Available: https://mdotnetpublic.state.mi.us/drive/. [Accessed 1 February 2018].

[32] M. Bartholomew, B. Blasen and A. Koc, "Bridge Information Modeling (BrIM) Using Open Parametric Objects," Federal Highway Administration., 2015.

[33] S. Jeong, R. Hou, J. P. Lynch, H. Sohn, and K. H. Law, "An information modelling framework for bridge monitoring," Advances in engineering software, 114, pp.11-31, 2017.

[34] Red Equation Corp, "OPEN BRIM V3," [Online]. Available: https://openbrim.org/www/brim/. [Accessed 1 2018 February].

[35] Computers and Structures, Inc., "Structural Bridge Design Software | CSiBridge," [Online]. Available: https://www.csiamerica.com/products/csibridge. [Accessed 1 February 2018].

[36] E. Gazoni and C. Clark, "Openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files," 24 January 2018. [Online]. Available: http://openpyxl.readthedocs.io/en/default/. [Accessed 1 February 2018].

[37] PyTorch, "PyTorch," [Online]. Available: http://pytorch.org/. [Accessed 1 February 2018].