# Towards data exchange interoperability in building lifecycle management

Sylvain Kubler and Manik Madhikermi and
Andrea Buda and Kary Främling
Aalto University, School of Science and Technology
P.O. Box 15500, FI-00076 Aalto, Finland
Email: firstname.lastname@aalto.fi

William Derigent and
André Thomas
Université de Lorraine, CRAN, UMR 7039
Vandœuvre-lès-Nancy F-54506, France
Email: firstname.lastname@univ-lorraine.fr

***Abstract*—Building Information Management systems have started to leverage new kinds of digital information infrastructures that integrate activities related to design, budgeting, scheduling, analysis, material management, and human resources. Companies implementing such systems have no other option today than exchanging information among themselves in a proper and efficient way. Industry foundation classes (IFCs) provide useful information structures for such a data sharing. However they do not specify how to capture, exchange and synchronize the information between distinct entities (i.e. information systems like sensors, servers, smart products...) throughout the building lifecycle. Accordingly, this paper investigates existing messaging protocols in order to identify which one is the most appropriate for supporting building lifecycle management; i.e a flexible protocol that provides sufficiently generic communication interfaces. A platform set up on a university campus based on the selected messaging protocol is then presented. Within this context, the research agenda that should be pursued to develop new techniques and algorithms for optimizing the building energy management is announced.**

***Index Terms*—Building information management; Messaging protocols; Quantum lifecycle management; Intelligent products; Energy management**

## I. Introduction

Advances in IT technologies over the last decades and the arrival of the so-called Internet of Things (IoT) has radically changed the traditional organizational boundaries and space limitations, fostering collaboration and coordination between distinct corporations. New developments in the use of wireless technologies such as RFID, sensor networks, make it possible to retrieve the identity of objects and monitoring items moving throughout their whole life [1]. Such technologies help companies to get real control of their items by "digitizing" multiple aspects of work processes that were previously supported by analogue tools. Digitized work practices can be modularized, integrated, and reconfigured. For instance, in the construction industry, building information management systems have started to leverage new kinds of digital information infrastructures that integrate activities related to design, budgeting, scheduling, analysis, material management, and human resources [2]. Companies at the source of these activities have no other option today than exchanging information among themselves in a proper and efficient way so as to optimize their costs, to respect governmental regulations, to keep track of their products and to improve the customer experience [3], [4].

This paper drafts a project aimed at designing a flexible system for building lifecycle management (BLM), whether in terms of energy management, facility management, maintenance management, or still product/information traceability management. Such a BLM system should be able to help users to integrate and reuse building information and domain knowledge throughout the building lifecycle [5]. BLM often begins with a physical analysis of the building to obtain a numerical representation (e.g. CAD documents). In this context, it is more than necessary to develop an appropriate BLM system at the beginning of its lifecycle in order to avoid information loss acquired during the building construction, during its use/maintenance and disposal [6]. A key ingredient to a successful BLM system rests upon a flexible communication infrastructure that provides sufficiently generic interfaces for exchanging all types of data consumed or provided by the actors, corporations or information systems in general involved in the building lifecycle [7]. Industry foundation classes (IFCs) provide a useful structure for data sharing among applications [8]. However, IFCs only focus on the information structure, they do not specify how to capture, exchange and synchronize information between different information systems. Current integrated IT tools also appear limited in providing such generic communication interfaces, thus hindering the extension of existing monitoring and control systems, even when considering IFCs. This lack of appropriate data exchange solutions currently outweighs the benefit of software interoperability [9]. However, the design of appropriate communication interfaces are the foundation to increase the quality of BLM since different aspects might therefore be correctly addressed, such as the visibility and sustainability of building-related information (i.e. their availability and persistency), or still the data security aspect [10].

In this respect, this paper introduces in section II the two-level challenge to be addressed for data exchange interoperability, and then gives insight into the main messaging protocols. Section III first introduces a framework used to compare the different messaging protocols so as to identify the most appropriate one in the framework of BLM. Section IV provides greater details on the adopted messaging protocol. Section V

presents the platform being set up on the ENSTIB[1] campus using this solution, along with the first proofs-of-concept and results; conclusions follow.

## II. Data exchange interoperability

### A. Two-level challenge

Data exchange interoperability can be achieved at the application level, which is a two-level challenge as illustrated in Fig. 1:

i) *Communication level integration:* necessity to provide infrastructures and mechanisms that support communications and transactions between distinct organizations, networks, applications and IT technologies,

ii) *Data level integration*: necessity to handle the many changes in data media and formats that occur throughout the product and information life cycle,

Over the last decades, techniques, concepts and standards have emerged at both levels [11], [12]. For instance, the ISO 16739 or UDEF are standards have been defined at the level of data integration. The focus of this paper is not given at this level but at the communication level.

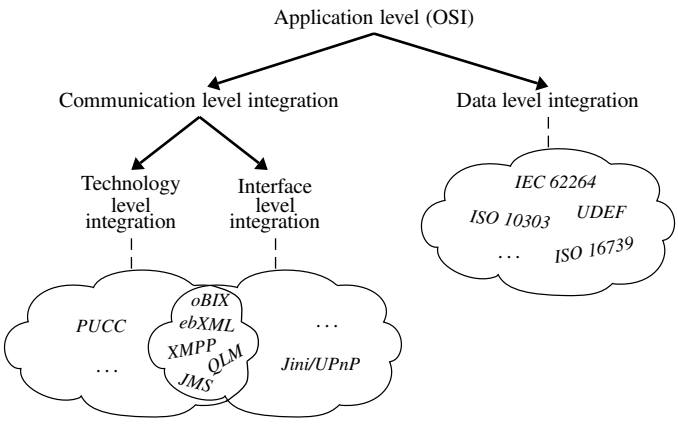



Fig. 1. Data exchange interoperability at the "Application level"

The communication level integration is in turn a two-level challenge as depicted in Fig. 1:

i) *Technology level integration:* necessity to provide a framework for integrating heterogeneous hardware and software platforms and to support the wide range of IT technologies (proprietary solutions, extended networks, *etc.*),

ii) *Interface level integration:* necessity to support different types of transactions that must be based on shared business references [13]. To gain time and efficiency and to avoid re-defining co-operation rules and software supporting it each time, these references must be based on business standards or norms. The business standards must be independent and weakly coupled with the technology level integration so as to support openness.

It is not an easy task to develop solutions supporting both levels of integration. As a matter of example, the Peer-to-peer Universal Computing Consortium (PUCC) [14] mainly focuses on the technology integration aspect by designing a general peer-to-peer networking platform that enhances the communication capabilities of various devices via various networks. The solutions developed by this consortium do not integrate the set of interfaces directly at the communication level but requires the deployment of the PUCC middleware, thus limiting the integration with other middlewares. Another example is the Jini/UPnP interoperability framework developed in [15] that allows clients to use pre-defined interfaces, but requires the use of Java technologies. This makes technology integration with non-Java applications quite daunting. Some solutions have nevertheless emerged for supporting both levels of integration such as oBIX (Open Building Information Exchange), ebXML (Electronic Business XML), XMPP (eXtensible Messaging and Presence Protocol), QLM (Quantum Lifecycle Messaging) or still JMS (Java Message Service). This paper only focuses on such protocols. The next section briefly introduces these five messaging protocols, which are then compared in section III.

### B. Candidate messaging protocols

*1) JMS:* It might be the most known messaging protocol among the five to be compared. The creation of JMS was a joint effort of several corporations supported by SUN Microsystem [16]. JMS is an Application Programming Interface (API) for enterprise messaging that provides the set of interfaces enabling point-to-point communications in distributed enterprise applications. JMS defines a single, unified message API that enables loosely coupled, asynchronous and reliable communications, and support the development of heterogeneous applications that span operating systems, machine architectures, and computer languages. JMS relies on the publish/subscribe (Pub/Sub) model and supports the subscription mechanism. In this model, publishers and subscribers are generally anonymous and may dynamically publish or subscribe to specific data. A centralized system takes care of distributing subscribed data coming from the publishers to all subscribers.

*2) oBIX:* It is an industry-wide initiative in the building area that enables mechanical and electrical control systems to communicate with enterprise applications. oBIX defines XML and Web Services-based mechanisms and the information is represented as a concise object model, where objects with any structure and contents can be freely exchanged [17]. A client-server architecture is used for accessing oBIX information over the network. Software can act as a client, as a server or eventually as both simultaneously. Three types of requests/responses referred to as "oBIX verbs" are defined:

- *Read*: it returns the current state of an object,
- *Write*: it updates the state of an object,
- *Invoke*: it triggers an operation on an object.

oBIX also provides mechanisms called *watch objects* that enable clients to subscribe an object event. More concretely, *watch objects* are created based on client requests, then the server provides the client with a URI to enable it to poll and get the latest status updates. It makes it possible to append

[1]École Nationale Supérieure des Technologies et Industries du Bois, Épinal (France)

new history records, query historical data or do basic roll up of numeric data. Subscriptions are cancelled either when the client unregisters the *watch object* or when the predefined lease time for polling action is exceeded. The oBIX alarm feature is a cornerstone of this protocol, which enables to raise an alarm when an incident occurs and to notify users or related applications. Broadly, oBIX is a suitable communication interface for data acquisition and control of a wide range of devices.

*3) XMPP:* It is a communication protocol for message-oriented middleware based on XML [18]. XMPP was introduced by the Jabber open-source community in 1999 and has been improved by the XMPP Working Group to provide Internet Engineering Task Force (IETF) Instant Messaging and presence technology. The XMPP RFC defines this messaging protocol as a robust, secure, scalable, internationalization-friendly architecture for near real time messaging and structured data exchange. In XMPP, the communication of XML messages is based on the XML stream rather than the XML document between two network entities. The XMPP technology uses decentralized client-server architectures in which clients open a stream to the server, which in turn, opens a stream back.

*4) ebXML:* It is an initiative of the OASIS group and the United Nations Centre for Trade Facilitation and Electronic Business, whose foundations rely on a former Business-to-Business solution called ooEDI (object oriented Electronic data interchange) [19]. This consortium promotes an open and XML-based infrastructure supporting the exchange of electronic business information in an interoperable, secure, and consistent manner. One objective for ebXML is to lower the barrier of entry to *e*-business, particularly in the context of small and medium enterprises. The ebXML specifications cover business processes and heterogeneous data shared between a wide range of actors and corporations.

*5) QLM:* This messaging protocol emerged out of the PROMISE EU FP6 project[2] in which real-life industrial applications required the collection and management of product instance-level information for many domains involving heavy and personal vehicles, household equipment, phone switches, *etc*. Information such as sensor readings, alarms, assembly, disassembly, shipping event, and other information related to the entire product lifecycle needed to be exchanged between several organizations. The project consortium set out to find suitable standards for exchanging such information. PROMISE created two main specifications that fulfilled the necessary requirements: the PROMISE Messaging Interface (PMI) and the PROMISE System Object Model (SOM). At the end of the PROMISE project, the work on these standards proposals was moved to the QLM workgroup of The Open Group[3]. The QLM messaging protocol is the continuity of PMI and consists of two standards proposals [20], namely the QLM Messaging Interface (QLM-MI) that defines what kinds of interactions between entities are possible, and the QLM Messaging Format (QLM-MF) that defines the structure of the information included in a QLM message.

[2]http://promise-innovation.com
[3]http://www.opengroup.org/qlm/

## III. Messaging comparison framework

In our study, several key properties are considered to compare the five messaging protocols previously introduced. These properties mostly come from the framework defined in [21], which are divided into three main categories: *i)* message delivery model, *ii)* message processing model, *iii)* message failure model. Sections III-A to III-C respectively introduce the set of criteria defined for each of these categories; Section III-D then compare the five standards based on these criteria.

### A. Message delivery model

This category defines the properties related to the delivery of the message to the intended recipient. The majority of the criteria composing our framework comes from this category:

1) *Representation*: it indicates the message representation adopted by the messaging protocol. It can be represented either as a *i) Data element* – the message consists in a messaging header and body that respectively contains information about the interface and the data to be conveyed (e.g. an XML message); or as an *ii) Object* – the message is specified as an object, which is nothing more than a programming method where its arguments are the set of information that compose the message. The *Object* representation assumes that this object/method is held by both the sender and the recipient of the message, while the *Data element* representation does not,
2) *Messaging API:* the protocol might be application-specific or -independent,
3) *Initiation:* it defines how the message is transmit between two nodes, which can be either server initiated (i.e. *push* mechanism) or client initiated (i.e. *pull* mechanism). In the *pull* mechanism, a client queries the server whenever it needs the data. The messaging protocol might support one or both mechanisms,
4) *Intermediation:* it specifies whether the messaging protocol offers the possibility to rely on an intermediate party to complete its transaction,
5) *Persistence:* it specifies whether the messaging protocol has a provision for data persistency. Two modes are defined: *i) Persistent* – the data is stored in the system until and unless it is retrieved, *2) Transient* – the data is stored in the system as long as the Time-to-Live (TTL) of the message does not expire,
6) *Subscription:* it specifies whether the messaging protocol proposes subscription mechanisms. Two mechanisms are considered: *i) interval-based* and *ii) event-based*,
7) *Self-Contained:* it specifies whether the message contains all the necessary information to enable the recipient to appropriately handle the message. In more concrete terms, the message contains all the relevant information such as the actions to be performed (read, write, subscription…), the message validity period (TTL), the communication mode (asynchronous/synchronous), *etc.*,

8) *Protocol agnostic:* it specifies whether the messaging protocol supports multiple underlying protocols, making it possible to transport the message using most "lower-level" protocols such as HTTP, SOAP, SMTP, FTP or similar protocols. It might also be possible to transport this message using files on USB sticks or other memory devices.
9) *Synchronicity*: it defines whether the messaging protocol supports *synchronous* and *asynchronous* communications. A communication is referred to as *synchronous* if all data is exchanged within a single active connection, while asynchronous can handle a request in separate connections,
10) *Delivery-Guarantee*: it defines whether the messaging protocol has provision for the delivery guarantee (e.g. by returning responses),
11) *Piggy backing*: it defines whether the messaging protocol allows piggy backing a new request with a response (i.e. without dissociating the new request of the response). This is a crucial property both for real-time communications and to enable communications with nodes located behind a firewall,
12) *Multiple payloads*: it states whether the messaging protocol supports multiple payload formats.

### B. Message processing model

This category deals with the message reception, i.e. how messages are processed by the recipient node and how responses are returned to the requester. Two criteria are defined:

13) *Processing result:* it defines how the requested information is returned to the requestor node. Three modes of response are defined: *i) single return value* – for every request, a single response is generated, *ii) single integrated return value* – the response contains the integrated value for the requested data, *iii) set of individual return value* – multiple response at different intervals are generated,
14) *Communication:* it defines how two nodes communicate once the request message is received. Two levels of communication are defined: *i)* Separate message, *ii)* Callback address.

### C. Message failure model

The third category describes what provisions the messaging protocol provides in case of failures and disruptions. Only one criterion is defined:

15) *Failure notification:* it defines whether the messaging protocol has a provision for failure notifications. Three main approaches are usually implemented: *Timeout of Acknowledgement*, *Reply with error message* or *Exception*.

### D. Synthesis

The five protocols previously introduced are potential candidates to support BLM since they all provide portable and standardized application-level interfaces for data acquisition and control of a wide range of devices. These five protocols are compared in TABLE I based on the comparison framework. Looking at this table, it can be observed that JMS and QLM cover the vast majority of the properties and sub-properties (i.e., they often offer the possibility to choose between different options where possible). In this paper, we claim that the messaging protocol supporting BLM has to meet three important general requirements, namely:

1) to support the complete building lifecycle and related activities, from beginning of life (BoL) including design, analysis, production, and construction of the building, through Middle of Life (MoL) including its use and maintenance, up to End of Life (EoL) including its recycling and disposal;
2) to be able to communicate with any types of intelligent items/systems involved in the building lifecycle (sensors, actuators, RFID, databases…). A key element of the communication lies in the fact that the Messaging Interface standard should be protocol and network agnostic to support all types of technologies and infrastructures, should enable to convey building lifecycle information (related to the building itself or activities that surround it) to relevant data systems and users either synchronously or asynchronously, and should be able to support a wide range of notification mechanisms;
3) to support multiple payload formats and structures to handle the many changes in data media and formats throughout the building lifetime.

TABLE I
MESSAGING PROTOCOL COMPARISON BASED ON 16 CRITERIA

| n° | Property | Sub-property | oBix | ebXML | XMPP | JMS | QLM |
|---|---|---|---|---|---|---|---|
| 1 | Representation | Object | | | | ✓ | |
| | | Data Element | ✓ | ✓ | ✓ | | ✓ |
| 2 | Messaging API | Application-specific | ✓ | | ✓ | | |
| | | Application-indep. | | ✓ | | ✓ | ✓ |
| 3 | Initiation | Push | | ✓ | ✓ | ✓ | ✓ |
| | | Pull | ✓ | ✓ | | ✓ | ✓ |
| 4 | Intermediation | | ✓ | ✓ | ✓ | ✓ | ✓[4] |
| 5 | Persistence | Transient | | | | ✓ | |
| | | Persistent | ✓ | | | ✓ | ✓ |
| 6 | Subscription | Interval-based | ✓ | | | ✓ | ✓ |
| | | Event-based | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | Self-contained | | ✓ | ✓ | | ✓ | ✓ |
| 8 | Protocol agnostic | | | ✓ | | ✓ | ✓ |
| 9 | Synchronicity | Synchronous | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | Asynchronous | | ✓ | | ✓ | ✓ |
| 10 | Delivery-guarantee | | ✓ | | | ✓ | ✓ |
| 11 | Piggy backing | | | | | | ✓ |
| 12 | Multiple payloads | | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | Processing Result | Single return value | | ✓ | ✓ | ✓ | ✓ |
| | | Single integrated… | ✓ | | | | ✓ |
| | | Set of individual… | ✓ | | | | ✓ |
| 14 | Communication | Separate message | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | Callback | | | | | ✓ |
| 15 | Failure Notification | Timeout of Ack. | | | | ✓ | ✓ |
| | | Error message | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | Exception | | | | ✓ | |

[4]Unlike the four other messaging protocols, intermediation is not mandatory in QLM. However, QLM offers the possibility to use such functionality.

Given these requirements, an aspect of prime importance considering complex product lifecycles, as a building lifecycle, is the support of *Multiple payloads* for being embedded in a message (i.e., to meet in part requirement 1). The non-support of this functionality is particularly critical when many actors, corporations and systems are involved in the building lifecycle. Indeed, the higher the product complexity, the higher the number of formats used to store information. In this regard, all protocols support this functionality. Considering now requirement 2, some protocols are limited in some respects: first, some of them do not support synchronous and asynchronous communications (see property 9 in TABLE I), which is a hurdle for supporting a wide range of activities in the building lifecycle (e.g., when communicating with devices that are frequently disconnected from the network); second, some of these protocols are not protocol agnostic (see property 8, e.g. oBIX is only specified for HTTP and WSDL/SOAP protocols) that could prevent building stakeholders from communicating using specific technologies; third, none of these protocols, except QLM, have support for piggy-backing (see property 11) that is of utmost importance to deal properly with products and systems that are mobile or located behind firewall systems [22]; fourth, only QLM provide the possibility of making subscription *with* callback address (see property 14) that is an important feature for real-time notifications. The lack of one of these properties has a significant negative impact on meeting requirement 2. Given these observations, QLM seems to better meet the major BLM requirements. Another determining factor in our decision of using QLM for building lifecycle management is that QLM can use different payload formats including, for instance, oBIX objects. Thus, oBIX can be used as the payload format when communicating with building-related information systems that already have existing support for handling oBIX object structures. Our choice is therefore to use the QLM standards. These standards are introduced in greater detail in section IV.

## IV. QLM MESSAGING STANDARDS

QLM messaging specifications consist of two standards proposals [20]: the *QLM Messaging Interface* (QLM-MI) that defines what types of interactions between "things" are possible and the *QLM Data Format* (QLM-DF) that defines the structure of the information included in QLM messages. Whereas the Internet uses the HTTP protocol for transmitting HTML-coded information mainly intended for human users, QLM is used for conveying lifecycle-related information mainly intended for automated processing by information systems. In the same way that HTTP can be used for transporting payloads in formats other than HTML, QLM can be used for transporting payloads in nearly any format. XML might currently be the most common text-based payload format but others as JSON and CSV can also be used. The accompanying standard QLM-DF partly fulfills the same role in the IoT as HTML does for the Internet, meaning that QLM-DF is a generic content description model for things in the IoT.

### A. QLM Data Format (QLM-DF)

QLM-DF is defined as a simple ontology, specified using XML Schema, that is generic enough for representing "any" object and information that is needed for information exchange in the IoT. It is intentionally defined in a similar manner as data structures in object-oriented programming. QLM-DF is structured as a hierarchy with an "Object" element as its top element. The "Object" element can contain any number of "Object" sub-elements, which can have any number of properties, referred to as InfoItems. The resulting Object tree can contain any number of levels. Every Object has a compulsory sub-element called "id" that identifies the Object. The "id" should preferably be globally unique or at least unique for the specific application, domain, or network.

The high flexibility of the Objects tree makes it possible to respect a precise structure or data model defined in an application or by a standard. QLM-DF defines an extension mechanism that makes it possible to use class inheritance in similar ways as in object-oriented programming. This extension mechanism enables the creation of domain-specific extensions of QLM-DF, while preserving a basic compatibility between different domain extensions. To date, the QLM workgroup has created one such extension, called the *Physical Product Extension*, which provides specifications for managing product lifecycle-related information [23]. Similar extensions respecting IFC specifications is planned to be created in our research.

### B. QLM messaging interface (QLM-MI)

A defining characteristic of QLM-MI is that QLM nodes may act both as "servers" and as "clients" and therefore communicate directly with each other or with back-end servers in a P2P manner. Typical examples of exchanged data are sensor readings, lifecycle events, requests for historical data, notifications, *etc.* One of the fundamental properties of QLM-MI is that QLM messages are "protocol agnostic" so they can be exchanged using HTTP, SOAP, SMTP, or similar protocols. Three operations are possible through QLM:

1) *Write:* used to send information updates to a node;
2) *Read:* used for **immediate retrieval** of information and for placing **subscriptions** for deferred retrieval of information from a node;
3) *Cancel:* used to cancel a subscription.

The subscription mechanism is a cornerstone of that standard. Two types of subscriptions can be performed:

1) *subscription **with** callback address:* the subscribed data are sent to the callback address at the requested interval (could be *interval-based* or *event-based*);
2) *subscription **without** callback address:* the data are memorized on the subscribed QLM node as long as the subscription is valid. The historical data can be polled by issuing a new QLM read query.

## V. BUILDING MANAGEMENT ON THE CAMPUS FIBRE

The goal of this research is to propose portable, interoperable and independent information management tools in the

building industry and related areas. QLM messaging standards are used to fill that need. These standards are implemented in different rooms on the *Campus Fibre*[5], namely a technical hall, two computer science lab rooms and one office (equipped with desktop workstations). This platform will be used for various experimental purposes:

- to equip the site with an energy and indoor quality air monitoring system, whose generated data will be available online. This open data repository will be accessible to any partners of the project or future collaborators;
- to enable proper modeling based on the acquired data (e.g., behavioral models of the targeted buildings);
- to enable testing virtual environments and/or energy management strategies for improving the energy efficiency of the targeted buildings;
- to develop ambient/intelligent systems to pervasively support the different actors in their respective missions.

The main purpose in choosing three different types of rooms is to study the site from different perspectives. For instance, from an energy consumption perspective, the technical hall is probably the most interesting room to be considered due to its strong impact on the campus energy consumption. From a user comfort perspective, it is easier to develop solutions in an office than in a technical hall. Fig. 2 provides a view of both a traditional building lifecycle (from BoL to EoL) and one of the two lab rooms (denoted by lab room 1) that has been instrumented with different types of sensors (contact and temperature sensors) in order to monitor each element involved in the building energy and the indoor air quality (windows, doors, radiators, air-conditioning systems).

Section V-A presents the QLM communication infrastructure, as well as the set of QLM interfaces used in lab room 1.

### A. Supervision of a lab room using QLM messaging

The different sensors disseminated over the lab room are connected to the QLM "cloud" using three distinct nodes (Raspberry PIs to be exact), as depicted in Fig. 2. The QLM "cloud" corresponds, in our study, to the set of communications interconnecting each phase and organization/actor involved in the building lifecycle, as highlighted in Fig. 2. It is therefore possible for any organization implementing QLM to access and convey any lifecycle information related to the building. Let us consider the example of a service provider for help with verifying the two air conditioning appliances located in lab room 1 (see Fig. 2). This service provider, whose server is hosted on another geographic site, is now able to send a subscription request to Raspberry PI 2 to subscribe to data collected by temperature sensors TS1 and TS2 (see Fig. 2). This actor will thus continually receive the subscribed values depending on the type of subscription he has performed (interval-based, event-based, with or without callback address…). In this platform, values collected from the different sensors are mainly used to enrich the building information management (BIM) database with "real-time" data. Two types of subscription are defined between the BIM database and the different sensors (CS1 to CS6 and TS1 to TS6):

- *interval-based subscriptions:* all temperature sensor values are subscribed every 15 min by giving the BIM database address as callback address. Accordingly, the database is updated at the requested interval with current temperature values, considering each part and heat source of lab room 1;
- *event-based subscriptions:* all contact sensor values are subscribed "event-based", by giving the BIM database address as callback address. In our case, this means that a new value will be generated in the database each time a door or a window is closed/opened.

Fig. 3 provides the sequence diagram when subscribing to contact sensor CS3. In this example, the site administrator initiates a subscription from his phone by selecting the information to be subscribed on Raspberry PI 3 (CS3 is selected). The administrator also specifies the type of subscription to be performed (*with* callback), the callback address (the BIM database), the duration of the subscription (set to "-1", which means as long as the subscription is valid), and lastly the interval value (set to "-1" that means "event-based"). The resulting QLM subscription request/message containing all these parameters (built on the basis of the Object tree described in section IV-A) is shown in Fig. 3. This request is then sent to Raspberry PI 3, which creates the subscription and, subsequently, returns a QLM response to the requestor (i.e., the administrator) with the ID of the subscription that can be useful to cancel the subscription before it expires. Therefore, each time window 1 is opened or closed, a notification is automatically pushed to the BIM database using a QLM response, as illustrated in Fig. 3.

Figures 4(a) to 4(c) and Figures 4(d) to 4(e) respectively show the behavior of the two types of subscriptions itemized above, considering a few doors and windows. In the first series of figures (temperature sensors), values are updated in the BIM database every 15 min (time window: 5 a.m. to 12 p.m.), while in the second series (contact sensors) values change in an impromptu manner (time window: 3 p.m. to 7 p.m.). Based on such "real-time" information, it is further possible to design techniques and models to learn and control different parts of the building and the campus. For instance, a first observation is that values generated by TS1 and TS2 have a similar evolution, while values generated by TS3 are continuously higher than these two sensors. This can be explained because TS1 and TS2 respectively monitor the temperature of both air conditioning systems (*cf.* Fig. 2), while TS3 monitors a radiator. One may nevertheless wonder whether it is normal that TS2 is less stable than TS1 (*is there any disruption of the air conditioning system monitored by TS2, or of TS2 itself?*). Another observation, when focusing on CS3 (see Fig. 4(e)), is that window 1 stays open from 3 p.m. to 7 p.m. It could eventually be possible, based on this observation, to trigger an alarm on the concierge's phone.

Although the collected information is, at the current stage, only used to enrich the BIM database, the flexibility of

[5]http://www.enstib.univ-lorraine.fr/en/home/enstib/presentation/

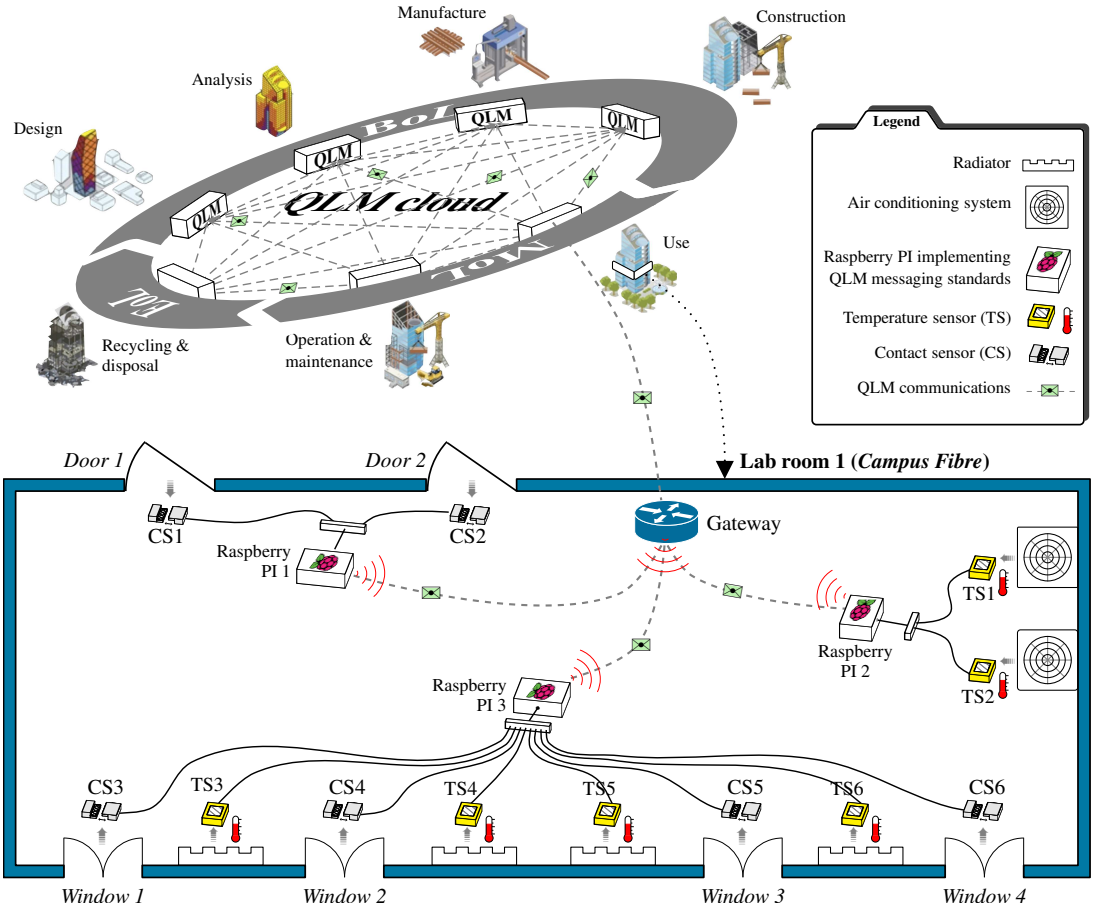



Fig. 2. Instrumentation of a classroom on the *Campus Fibre* using QLM

QLM makes it possible to further extend this platform to other purposes/phases of the building lifecycle. The collected information could, for instance, be transmitted in immediate or deferred ways to repair and design companies, which could lead to improved maintenance scheduling, product design and manufacturing procedures. More generally, more advanced services and interactions among building stakeholders are now possible using QLM. In situations where organizations would use different data formats and structures, would implement firewalls, would like to access information asynchronously or synchronously, the necessary interfaces would be available.

## VI. Conclusion

Building Lifecycle Management (BLM) is becoming an important aspect of the building industry. However, such an initiative is challenging due to the wide scope that needs to be covered by BLM, namely the building's entire lifecycle. The project may address buildings, data, applications, processes, organizational issues, people, *etc.* Most organizations do not have communication infrastructures that operate across such a wide area. Infrastructures and applications are often domain-specific or lifecycle phase-specific, which somehow hinders their openness. Today, such a limitation is mainly due to the lack of sufficiently generic and standardized application-level interfaces that are of major importance to provide the right building service, whenever it is needed, wherever it is needed, by whoever needs it. Without an appropriate messaging protocol as foundation of a BLM project, the organization will not be able to evolved to meet the new business needs, especially when they drastically change.

In this regard, five potential messaging protocols are compared in this paper in order to identify the solution that better meets the major BLM requirements. This study shows that JMS and QLM are two good candidates, but our choice turned to QLM because QLM messaging format is highly flexible and it has fundamental properties such as the *piggy backing*, the possibility of *callback address*, that contribute to increase the scope of possible applications throughout the building lifecycle. A platform set up on a university campus proves that deploying QLM messaging in building environments is technically feasible based on cheap and reliable wireless technologies. Based on this architecture, any partner implementing QLM is now able to access specific building information, e.g. to receive real-time data or to retrieve historical values.

## References

[1] G. Meyer, K. Främling, and J. Holmström, "Intelligent products: A survey," *Computers in Industry*, vol. 60, no. 3, pp. 137–148, 2009.

[2] M. Richards, *Building information management: A standard framework and guide to BS 1192*. BSI Standards, 2010.

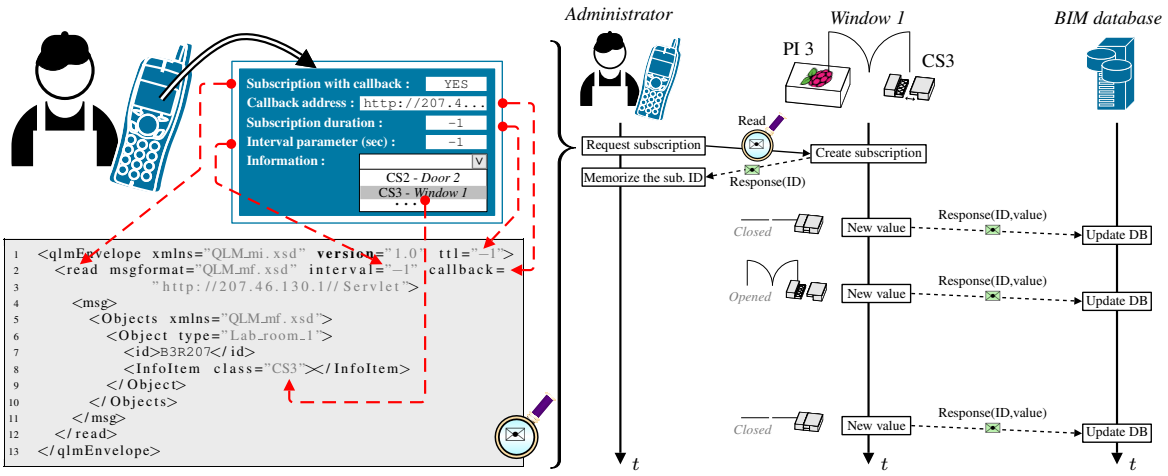
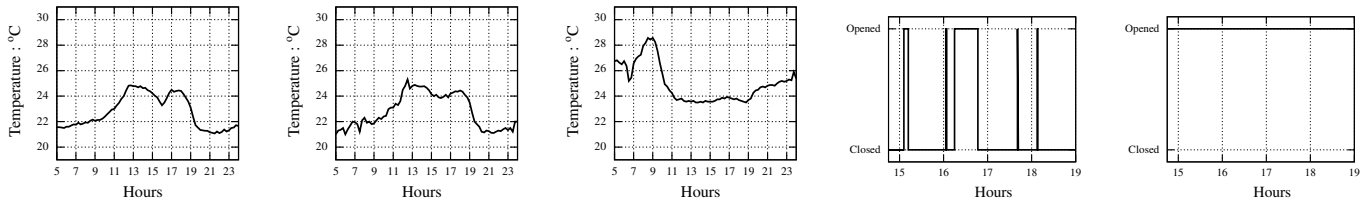



Fig. 3. Creation of an "event-based" subscription regarding Contact sensor CS3




(a) Temperature sensor TS1 (b) Temperature sensor TS2 (c) Temperature sensor TS3 (d) Contact sensor CS1 (e) Contact sensor CS3

Fig. 4. Real-time sensor values collected using the two types of subscription defined in QLM


[3] J. Stark, *Product lifecycle management: 21st century paradigm for product realisation*. Springer, 2011.

[4] K. Främling, J. Holmström, J. Loukkola, J. Nyman, and A. Kaustell, "Sustainable PLM through intelligent products," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 2, pp. 789–799, 2013.

[5] G. Lee, R. Sacks, and C. M. Eastman, "Specifying parametric building object behavior (BOB) for a building information modeling system," *Automation in construction*, vol. 15, no. 6, pp. 758–776, 2006.

[6] S. Bonandrini, C. Cruz, and C. Nicolle, "Building lifecycle management," in *Proceedings of the International Conference on Product Lifecycle Management*, vol. 5, 2005, pp. 461–471.

[7] R. Vanlande, C. Nicolle, and C. Cruz, "IFC and building lifecycle management," *Automation in Construction*, vol. 18, no. 1, pp. 70–78, 2008.

[8] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. Wiley.com, 2011.

[9] K. U. Gökçe and H. U. Gökçe, "eeBIM for energy efficient building operations," *14th international conference on computing in civil and building engineering*, 2012.

[10] V. Bazjanac, "Building energy performance simulation as part of interoperable software environments," *Building and Environment*, vol. 39, no. 8, pp. 879–883, 2004.

[11] A.-J. Berre, A. Hahn, D. Akehurst, J. Bezivin, A. Tsalgatidou, F. Vermaut, L. Kutvonen, and P. F. Linington, "State-of-the art for interoperability architecture approaches," *InterOP Network of Excellence-Contract no.: IST-508*, vol. 11, 2004.

[12] A. Koronios, D. Nastasie, V. Chanana, and A. Haider, "Integration Through Standards–An Overview Of International Standards For Engineering Asset Management," in *4th International Conference on Condition Monitoring*, 2007.

[13] D. Chen, G. Doumeingts, and F. Vernadat, "Architectures for enterprise integration and interoperability: Past, present and future," *Computers in Industry*, vol. 59, no. 7, pp. 647–659, 2008.

[14] N. Ishikawa, T. Kato, H. Sumino, S. Murakami, and J. Hjelm, "Pucc architecture, protocols and applications," in *4th IEEE Consumer Communications and Networking Conference*, 2007, pp. 788–792.

[15] J. Allard, V. Chinta, S. Gundala, and G. G. Richard III, "Jini meets UPnP: an architecture for jini/UPnP interoperability," in *Symposium on Applications and the Internet*, 2003, pp. 268–275.

[16] M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. Stout, "Java message service," Sun Microsystems Inc., Santa Clara, CA, Tech. Rep., 2002.

[17] M. Neugschwandtner, G. Neugschwandtner, and W. Kastner, "Web services in building automation: Mapping KNX to oBIX," in *5th IEEE International Conference on Industrial Informatics*, 2007.

[18] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core," Core; IETF: Fremont, CA, USA, Tech. Rep., 2004.

[19] B. K. Gibb, and S. Damodaran, *ebXML: Concepts and application*. John Wiley & Sons, Inc., 2002.

[20] K. Främling and M. Maharjan, "Standardized communication between intelligent products for the IoT," in *11th IFAC Workshop on Intelligent Manufacturing Systems*, vol. 11, 2013, pp. 157–162.

[21] S. Tai, and I. Rouvellou, "Strategies for integrating messaging and distributed object transactions," in *IFIP/ACM International Conference on Distributed systems platforms*, 2000.

[22] S. Kubler, M. Madhikermi, A. Buda, and K. Främling, "Two-way communications through firewalls using QLM messaging interface," in *11th IFAC Workshop on Intelligent Manufacturing Systems*, vol. 11, 2013, pp. 157–162.

[23] S. Parrotta, J. Cassina, S. Terzi, M. Taisch, D. Potter, K. Främling, "Proposal of an Interoperability Standard Supporting PLM and Knowledge Sharing ," in *10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Service*, 2013, pp. 286–293.