Software Platforms for Smart Building Ecosystems: Understanding the Key Architectural Capabilities and Trade-offs

Marcelo Cataldo[1], David Garlan[2], James D. Herbsleb[2],
Amber Lynn McConahy[2], Young-Suk Ahn Park[2], Bradley Schmerl[2]

February 2013
CMU-ISR-13-104

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

[1] Robert Bosch LLC, Research and Technology Center
[2] Institute for Software Research, School of Computer Science, Carnegie Mellon University

ABSTRACT

This report examined the technical aspects of a future ecosystem-ready software platform for the smart building domain. We analyzed the type of contributors that may exist in a smart building ecosystem, the quality attributes that those roles are concerned with, and the key functional attributes that the software platforms of future smart buildings should provide in order to make smart building ecosystems successful. Combining analyses of interviews, use-cases, and architectural scenarios, we identified a collection of key functional capabilities and several architectural trade-offs that need to be evaluated and carefully considered.

Keywords: Software Architectures, Software Platforms, Ecosystems, Smart Buildings.

Contents

1. Introduction

Over the decades, Bosch has developed core competencies in technology innovation and transforming those technologies into products for capital-intensive industries that are characterized by relatively predictable patterns of evolution. However, we are in the midst of an important transformation where customers are requesting "*more value*" from a product (e.g. personalization through the provision of a large number of complementary accessories or custom applications). Companies are being forced to respond to market demands faster and to provide more options than ever before (Zhu et al, 2007). In this new environment, the key driver for innovation and value creation is not the technology per se, but rather how technologies are being used and recombined which enables third parties to contribute easily. These "*applications*" represent a business space that is significantly less understood and predictable. Consequently, developing products in this new environment requires different approaches that enable and support "*exploratory development*" and "*agility*" (MacCormack et al, 2001; Smite et al, 2010). Furthermore, exploration implies that failures should not be avoided but must be embraced and managed. An innovative and flexible mindset needs to be part of the development approaches (Brandenburger & Nalebuff, 1998; Iansiti & Levien, 2004; MacCormack et al, 2001; Smite et al, 2010). Platform ecosystems provide a potential solution to the challenges presented by developing products for a less understood and predictable business space. For instance, the iPhone and Android ecosystems are classic examples of supporting extra value to end users through mass personalization. These ecosystems offer thousands of applications. However, most users only have a relatively small number of applications on their mobile devices, and these applications provide a particular "value" that the users require. More importantly, the number of applications offered in these ecosystems is significantly higher than any company on its own could develop or even envision. A failure to understand and meet these new challenges on time can have a dramatic impact on the existing business, as recent prominent examples (such as Nokia and RIM) demonstrate.

For Bosch in particular, the business context must encompass current practices as well as future business areas, such as the Internet of Things and Services. In this context, platform ecosystems provide a number of key benefits:

- **Accelerate innovative capability** by bringing together contributions from a wide range of different types of participants such as platform developers, application developers and, even end users.
- **Enable efficient and flexible product development**, which is becoming increasingly more challenging for any single company as the heterogeneity and complexity of new systems continue to increase.
- **Reallocation of risk** allows for the exploration of failure conditions by transferring the development effort and cost of certain features or applications from the company to other partners in the ecosystem. In this way, failure can be managed more easily while maintaining low costs ("fail fast and cheap").

In this report, we focus on the technical aspects of a future ecosystem-ready platform for those domains. The rest of the document is organized as follows. First, we provide a brief overview of the concept of platform ecosystem including both technical and non-technical considerations. We then describe the methodology used in our analyses, which is followed by the corresponding results. We conclude the document with a preliminary proposal of the architecture for a SW platform as well as recommendations for next steps.

2. The Basics of Platform Ecosystems

Business partnerships and business networks are pervasive in our economy. Furthermore, the Internet and related technologies have introduced major transformations enabling the creation of distributed business networks not possible a few decades ago. In recent years, academics and practitioners started to use the term ecosystem as a biological analogy to traditional business networks. Firms interact with each other in complex ways, and the health and performance of each firm is dependent on the health and performance of the whole. Firms are therefore simultaneously influenced by their internal capabilities and by their complex interactions with the rest of their business context, and the notion of ecosystems captures these complex dynamics quite well.

The business dimension of ecosystems has received the most attention. However, researchers (e.g. Iansiti & Levien, 2004) have argued that the key players in an ecosystem, known as *keystones*, have to recognize the critical impact of the platform and take steps towards realizing this impact by creating real opportunities for the other firms in the ecosystem. In other words, platforms are at the core of an ecosystem and are the fundamental basis upon which a business ecosystem is developed. Hence, platforms create niches where other participants can thrive.

In order to illustrate this, an example offers insight into these concepts. If we consider the iPhone ecosystem, there is the iOS platform, which is developed, controlled and maintained by Apple. Based on the platform and the accompanying toolset, other companies as well as individuals participate in the ecosystem by developing and selling applications that end users acquire through the App Store. Apple has developed a number of mechanisms that govern numerous aspects of the ecosystem, such as how participants in the ecosystem develop their applications, how to ensure quality of these applications, how these are offered to end users, and how end users acquire and use those applications. The combination of platform properties and governance mechanisms allow niche players (e.g. app developers) to extract value while the platform owner (e.g. Apple) gains by appropriating value from the rapid adoption and market dominance of the platform.

Gawer and Cusumano (2002) also argued that ecosystems are critical mechanisms for companies to become platform leaders and drive innovation within their industries. The authors suggested that for companies to have their products as the foundation from which other companies can build their own products, four "*levers*" can pull the control of the ecosystem into a more advantageous position. These four dimensions of control cover both strategy formulation and implementation issues, which are unavoidably intertwined, whereby keystones need to make choices on these dimensions in a coherent fashion. The four dimensions are:

1. **Platform architecture**, which consists of all the decisions made by keystones and wannabes related to the architecture of the product and the broader platform, such as what functionality is included and how other participants can access it.
2. **Relationship among participants**, which focuses on defining how collaborative or competitive the relationship should be between keystones (or wannabes) and niche players.
3. **Incentives and value creation**, which consists of decisions made by the firm in terms of what business arrangements are made with other players, thus allowing them to create value and to provide incentives that encourage others in the ecosystem to participate.
4. **Internal organization**, which focuses on the organizational mechanisms the platform leader uses internally to coordinate its own work in the ecosystem context, to orchestrate the ecosystem as a whole,

and to establish a level of transparency that gives ecosystem partners confidence in its stewardship. This dimension covers issues related to governance, collaborative technology, processes, and culture.

Herbsleb and colleagues (2010) developed a framework based on a similar set of dimensions to analyze platform ecosystems. They studied several successful ecosystems such as Eclipse, Gnome, Apache, and VistA, and their results have an important implication for Bosch: *platform ecosystems are not all created equally.* On one end of the continuum, we have a platform ecosystem where there is a **single platform** and a **single keystone** (e.g. iPhone ecosystem). On the opposite end of the continuum, we have a more complex ecosystem where **multiple platforms** could exist, and there are high levels of **heterogeneity** in terms of hardware and software elements. The home and building automation setting is a good example of this type of ecosystem, where a wide range of devices (e.g. appliances, security systems, HVAC systems, etc) and manufacturer interests must be supported by a platform. Such a heterogeneity continuum demands different attributes from the platforms at different points along the continuum.

Combining the work of Iansiti and Levien (2004), Gawer and Cusumano (2002) and Herbsleb and colleagues (2010), we consider a **platform ecosystem** as *(a) a platform, (b) a complex set of interrelationships among companies and individuals, and (c) a collection of mechanisms (e.g. economic, legal, and organizational) that foster those relationships to create and appropriate value around a product platform*.

2.1. Technical Considerations

Academic researchers suggest that developing and sustaining a platform ecosystem requires an understanding of how to design product platforms suitable for an ecosystem as well as those design decisions related to the other three critical dimensions of ecosystems (Gawer & Cusumano, 2002; Herbsleb et al, 2010; Iansiti & Levien, 2004). A product platform is the core element of a potential future ecosystem for Bosch. Product platforms need to have a particular set of attributes that allow participants in an ecosystem to make use of the shared technical work and, ultimately, extract value from participating and contributing to the ecosystem. These attributes can be separated into two distinct sets. First, we have the critical general characteristics of the platform:

- The **modularity of the platform**, which refers to the ability to complete development work on different modules relatively independently (Baldwin & Clark, 2000). The modularization approaches used in systems developed in-house (like all Bosch development projects) are not typically sufficient for a platform ecosystem setting. Therefore, platform functionality must be exposed in stable ways so that ecosystem participants are able to keep pace with the evolution of the ecosystem. Thus, modularity is a key factor limiting the effects of platform evolution on external interfaces.
- The **openness of the platform** is another critical attribute and refers to the extent that the platform owner allows other ecosystem members to participate in the development and evolution of the platform itself. For instance, in the case of a privately owned platform like what Bosch would be in a position to develop, the degree of openness involves a crucial tradeoff between value creation from broader adoption, fostered by openness, and the ability to profit from the value created, which can be hindered by openness (West, 2003).

Second, platforms in an ecosystem context present several design challenges that differ in important ways from platforms developed to support in-house development, such as traditional product-line approaches used in large development organizations. These differences are realized as specific new requirements for SW platforms, such as:

- **Heterogeneity of Devices and Functionality**: Domains, such as smart homes, smart buildings, assisted living, or eMobility are characterized by significant levels of heterogeneity at the level of devices as well as at the level of functionality that make use of those devices. Consequently, software platforms need to provide mechanisms to adequately support these different forms of heterogeneity.
- **Extensibility of the Platform**: In terms of the evolution of a software system, there are two key inter-related aspects that should be carefully considered. On the one hand, we have the evolution of interfaces (or APIs) that refers to changes in the interfaces that support new functionality while maintaining compatibility with existing versions of the interfaces. On the other hand, we have the evolution of the functionality of the software system, which refers to adding new features or extending existing ones.
- **Liability Mitigation Mechanisms**: A continual challenge in open software ecosystems as well as certain other domains is the uncertainty surrounding the reliability and support of the system. In other words, users may want assurances about important characteristics of the system. For instance, how reliable and available the system is in the field, how it will respond in various failure conditions, and what are the consequences of different types of failures. In addition, in the context of an ecosystem, there are governance and organizational aspects that come into play. For instance, how quickly will the platform owner repair a major defect, and what happens to niche players if they need support, and who is legally liable in case of costly system failures, etc.
- **Scope of the Platform**: One important result from academic research on open source projects is that a platform ecosystem should start with a platform that has a basic set of features that are complete. This has a dual purpose. First, this ensures that the initial product is sufficient for at least some applications, which will begin to create a base of users. Second, it assists new and prospective participants in understanding the ecosystem and the potential products available, hence, creating a basic foundation for future engagement. However, the ecosystem participants involved in the development of the platform need to carefully consider the decisions about that what functionality should be incorporated into the platform and what functionality should be developed by external parties as part of domain specific extensions or applications. As a result, such decisions have important implications on the incentives a particular platform creates for external participants.

2.2. Non-technical Considerations

While a technical platform is usually the basis for an ecosystem, a successful ecosystem requires addressing adequately a number of aspects associated with other key dimensions of an ecosystem, as well as their interrelationship with properties of the technical platform. These key dimensions are:

- **Ecosystem Management and Governance** concerns decision rights, specifically, who has the right to make what kinds of decisions. It is critical to assign roles and responsibilities to individuals and/or entities with the right incentives and competencies to create, grow, maintain, and protect an ecosystem as well as understand the needs of existing and prospective ecosystem members. It is important to point out that too much central control over the platform threatens one of the primary benefits of open platforms – the ability of members to freely and independently innovate. Often, "community managers" play an important role, providing support to resolve issues related to the needs of the ecosystem's participants as well as their development through community events such as training and conferences. For example, ecosystems may be organized around a foundation that plays a key stewardship role, or may have well-organized user groups that help bring new members on board and provide feedback to the platform owner about the needs of niche players and end users.

- **Incentive systems** are critical mechanisms to motivating participation and sustainability of the ecosystem. Incentives may be created by the attributes of the platform (e.g. interfacing mechanisms that do not force participants to disclose IP) as well as non-technical approaches (e.g. a foundation organizing approach that reduces risk for niche participants to invest in products for the ecosystem, since foundations will not compete for business).
- **Development and collaborative tools** that support ecosystem participants in the creation of their contributions. A successful approach involves choosing standard tools and development languages rather than proprietary ones in order to make the learning curve easier. In addition, a valuable collaborative infrastructure should support ecosystem participants so they can easily find others with compatible interests and goals, and share information about needs, opportunities, and changes.

3. Methodology of the Analyses

We approach our investigation in three complementary steps. First and building on past academic research, we initiated an analysis to identify the potential set of roles or participants in a smart building ecosystem. This initial analysis served as the basis for identifying a candidate setting for carrying out interviews across those roles, which was our second step. We identified the Gates Hillman building at Carnegie Mellon University because it is a state-of-the-art building with the latest green and sustainable technologies[1], and its construction was finished very recently. Through a few initial interviews with personnel from the facility management organization of CMU, we identified 9 different individuals and organizations that represented examples of many of the roles. In addition, interviews with those individuals allowed us to identify 3 more individuals and organizations as examples of other roles. In total, we conducted 13 semi-structured interviews. For those interviews, we developed a set of interview scripts tailored to the specifics of each role or type of stakeholder. The interviews were recorded and the content was transcribed. The transcripts of the interviews served as one of the data sources for our analysis. A second data source was a use-case workshop carried out with professors and doctoral students from CMU as well as researchers from Robert Bosch Corporate Research organization.

In the third and last step of our work, we used both data sources, the interviews and the outcome of the workshop, as inputs for analysis of functional and non-functional requirements for a future SW platform for smart buildings. We paid special attention to "pain areas" that exist today and those challenges could be addressed in a platform ecosystem of the future. In order to illustrate the various functional and non-functional requirements, we developed a collection of scenarios, and we used those scenarios to highlight (a) the specific implications in terms of architecting the SW platform and (b) the potential trade-offs that may arise. The results of those analyses are reported in the following section.

4. Results

In this section, we report the results of our analyses. We organize the content of our results as follows. We start with a brief overview of how the business context in the smart building domain has evolved over the years. Second, we present a summary of the insights from all the interviews. Third, we summarize the set of key architectural quality attributes that we identified from the analysis of the interviews. Finally, we present the collection of scenarios that we developed through our analyses and that illustrate the functional and non-functional requirements of future software platforms for smart building domains.

[1] http://www.cmu.edu/corporate/partnerships/gates_center.shtml

4.1. Overview of the Evolution of the Business Ecosystem

We carried out discussions with executives from various firms that supply systems for buildings today. These discussions allowed us to better understand the general business context of systems for buildings as well as how that context has evolved over the years. In general terms, we can identify two "generations" of the business ecosystem as well as a future third generation. The first generation is characterized by traditional independent systems (either with mechanical or electronic control systems). Developed over the past years, trends towards integrating systems within buildings are leading the way to the 2nd generation of the business ecosystem. We see the future of smart buildings based on an integrated platform as the 3rd generation. Figure 5.1 depicts the general structure of the business context across generations, and the following paragraphs describe the main attributes of the various generations.

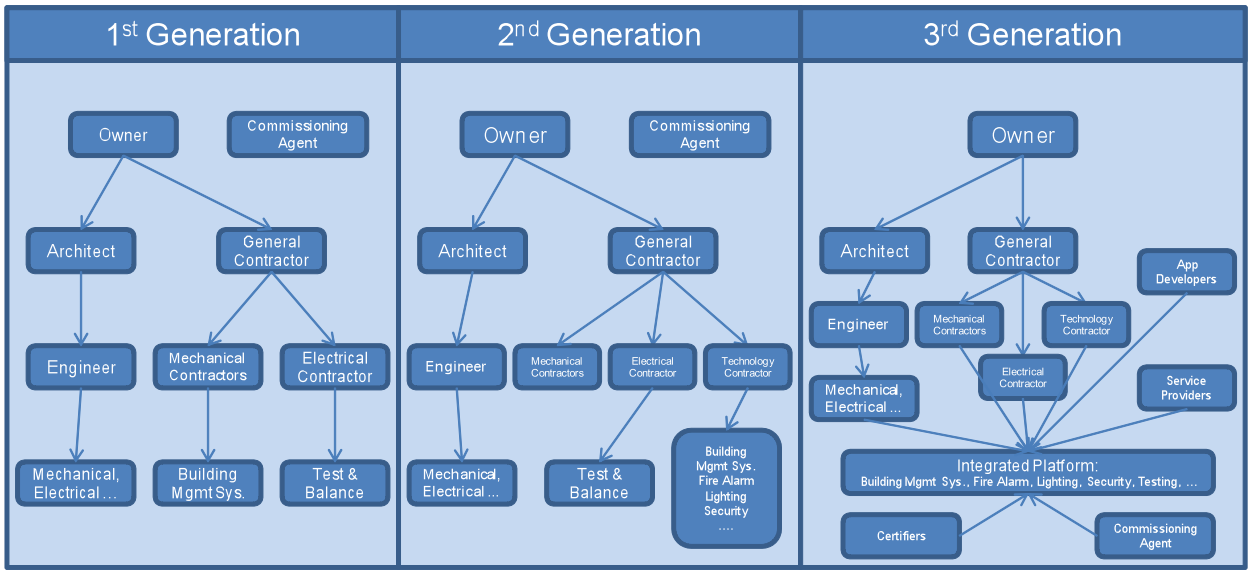

Figure 5.1: Evolution of the Business Ecosystem towards the Smart Building World

- First generation: this generation is characterized by completely independent systems. Consequently, the business context is also organized around silos that focus on particular competencies. In Figure 5.1, we have the example of a particular organization for a building that needs to be developed and constructed. In the 1st generation business context, the general contractor has a collection of specialists for each relevant discipline for each of the systems being installed.

- Second generation: due to the trend towards integration of systems, there is a change in the business context, particularly, in new developments. Certain companies are emerging with a new role, typically, called "technology" contractor, which in basic terms acts as the integrator and provides some basic platform for enabling the integration of two or more systems. In this role, the technology contractor needs to coordinate with the specialists of other disciplines (e.g. electrical, mechanical) on a number of aspects from the sequence of work to be done to the specific details of what type of sensors or actuators can or cannot be used in a particular integrated system.

- Future third generation: we envision that as smart buildings move towards an integrated platform setting, the business context will be significantly different from today's (2nd generation) context. First, the integrated platform would provide integration not at the low level (e.g. interconnectivity) that some of today's platform do but at a higher level of functionality that would enable the activities of different roles

in flexible ways. In this generation, we envision that roles such as inspectors, certifiers, and commissioning agents would be able to interact with the platform to perform their tasks. Today, those roles solely focus on specific parameters of the physical parts of the system and do not have interaction with any type of integration platform element. We also envision that the platform would enable not just the work of traditional specialists (e.g. mechanical, electrical, etc) but also enable a new host of roles such as application developers and service provides who contribute functionality into the platform of a future building.

Given the current state of the business context (2nd generation) and the trends that suggest a future 3rd generation with the characteristics described above, the focus of our analyses have been on identifying what the "pain" areas are today and how a 3rd generation system would be able to address those challenges.

4.2. Analysis of Interviews

We conducted 13 semi-structured interviews with individuals that play different roles in today's setting. The interviews were recorded, and the content was transcribed. In the subsequent pages, we report for each interview a brief summary of the interviewee's responsibilities and key insights from the discussion. Those insights combined with our analysis of current and future trends in relevant technologies allowed us to construct the list of scenarios discussed later in this report.

Interview #1: University Engineer	
Responsibilities	• Important decision maker for all capital expenditures on buildings, allocation of funds, and overseeing replacement, rebuilding, and restoring equipment • Responsible for steam system, electrical system, maintenance and operations, and all IT resources • Mainly works with people who are fixing things
Insights from Interview	• Pushed for building a custom IT system that takes data from existing equipment, uses open source SCADA, relational database, gets data from all over campus, provides simpler user interface for current systems (e.g. Vivarium). • Goal was to make it easier to understand what's happening when you get an alarm • Created and maintains a system called "electric line 1", which makes it easier to monitor electric consumption across campus (e.g. shows current status of feeders). • System makes use of a couple of standards (MODBUS and BACNet). Different "incarnations" of the system have been in existence for 17 years. • Lighting systems are also moving towards an integrated environment, equivalent to BACNet. For instance, it is now possible to automatically shut off lighting for Earth Hour. • Wean and Gates buildings have "limited" HVAC/lighting interface. For instance, heating/AC controls use occupancy sensors. • Sometimes use the ticket system to get information about what needs maintenance. They have no easy way to query or to see from ticket what equipment is referenced (e.g., which air handler feeds office where there is a problem). • Spatial layout of AutomatedLogic HVAC UI is excellent. Would like consistency throughout CMU because it can be hard to get necessary information in order to see what is going on. • Would be better to have open systems. • AutomatedLogic is closed and they are moving to Delta, which is more open, even though they don't have the great spatial layout feature. • They have two programmers who do logic programming of HVAC using a graphical programming language invented by AutomatedLogic.

Interview #2: Lead Engineer	
Responsibilities	• Inspections and problem handling • Supervises three HVAC mechanics who are responsible for subsets of buildings
Insights from Interview	• Inspections involve mainly physical activities to check air handlers, filters, pressure drop across filter, and belts • Typical problem is triggered by a work request submitted by the "customer representative" to the facilities management secretary. It is done via email. If urgent, the customer representative tends to call the mechanic directly. The secretary knows how to match up problems to people. • They diagnose the problem. For instance, if readings are nonsensical, the unit in the ceiling is reset (manually). If they are unable to fix the problem, they need to call LogicAutomation to fix. Although, many times they could see problem on computer (e.g. valve was constantly open). • If they need to look at some place inaccessible, they have to call maintenance personnel for lift. • They have a person responsible to call external vendors when necessary • Frustrating when there are multiple systems in a building. It is impossible to control and see everything in same system. For instance, some buildings, like Doherty, have multiple air handling systems: one for fume hoods and two for regular air handling. Coordination could also be very complex. • The systems they have typically log who does what for manual overrides and provides warnings about what has changed. • It is good practice to have systems organized by location rather than function • HVAC manufacturer provides a book that tells what reading values ought to be. They also provide sequence of operations and information about how everything is supposed to work. • Commissioning agents use these books to check that everything is functioning correctly, and they can override various settings to see if system responds as it should • Water temp is controlled by a different group in facility management (plumbers), but they coordinate with lead engineer on temperatures. • Steam also comes from a different system, and the HVAC systems interact with fitters over steam pressure. • Lead engineer also interacts with electrician, for instance, when sizing and/or ordering a new system. Coordination is about the location of a unit and the availability of power in right voltage. • HVAC system in the Gates building has a user interface that graphically shows all the programs that are running, for instance, for startup of the system, to make settings, to open valves, etc. The HVAC Vendor provides these programs. • These programs are customized for each room in the building if necessary. There are various dependencies among systems.

Interview #3: Senior Project Manager	
Responsibilities	• Handle projects from beginning to end for building construction on campus. • Starts with feasibility study, hiring of architect or planner, and figuring out what is wanted or needed • Goes through design process with architect and possibly a construction manager • Lifecycle cost analysis (although it is not clear who actually does the analysis)
Insights from Interview	• In early stages of the project, he engages with companies to bid on building systems and works out what is needed with university engineering and tenants. • Discusses designs proposed by architects to make sure they meet requirements. Example: what happens when occupant leaves a window open – does HVAC run continuously? Put a sensor on window? Problem: weather tightness, and who would be responsible if there was a leak? Also, each sensor (a "point") would cost $1K to add (programming cost). • When there is a new building project, many entities review and comment on proposed designs. In the case of the Gates building, 20-30 entities were involved, including campus, E&HS, cable, computing services, and university engineer. • Many challenges in these reviews. Most of the people just look at the plans, but not everyone can understand them. These reviews also need to be setup at different points in the design process because people have different perspectives. • The stages of a project are roughly: feasibility, planning, programming (predesign), schematic design, design development (take sketches and make them real, decide on systems, material, etc.), and construction documents (exactly how to be built). • At design development, usually 3-4 previously vetted vendors are invited to bid on each of the systems. Ideally, they would like to have some competition among control systems vendors (e.g. usually two). The Gates building only has one. Engineers prefer to have just one vendor. • Vendors present user interface (UI) and automation features as part of their bid. The UI capabilities were a big factor in the vendor selection for Gates. • The general view is that not many dependencies exist among bidding packages. However, they recognize that the biggest issue is coordination. For instance, integrating classroom lighting and classroom control systems – how do they interoperate? What platform should be used? These coordination efforts are extremely hard. • In fact, it is often not clear where to do integration. A typical situation is "vendors tell architects that they need to own it". If communication and coordination plan not decided on early in the project, the execution tends to be very difficult • CMU is responsible for testing, and it is all done at the end. There is no incremental testing. Nobody gets things done until the end. • Testing involves manually going in and testing every office as if it was occupied. Also, there are specific tests for major equipment. • Compliance with standards is always specified but rarely verified. • Serviceability is important, and they pay attention to designing for it, mainly because of cost implications. For instance, to retrofit one sensor for a window in an office, the cost maybe $5K-$10K. Costs stem from the complex process: (a) contract HVAC vendor, (b) who would write specialized program for the room, (c) subcontracts electrician to install wire and sensor and tie into one control, (d) then test and integrate

	• The cost of programming changes in existing programs is very high because they are locked into single vendor. In addition, problems (if any with new software) are very hard to diagnose o Example: in the Gates building, a mistake in programming introduced a defect that was difficult to diagnose (e.g. tried to retrofit, reboot) and resulted in modifying the control for each individual room. Lots of rooms missed.

Interview #4: Researcher/Faculty	
Responsibilities	• Research in assistive living systems • Builds, deploys, and gather lessons and feedback from deployment of experimental systems of sensors and actuators for particular tasks
Insights from Interview	• In a past research project, he set up the Carnegie Science Center boat, Voyager, with sensors throughout and internet connectivity for the sensor). Idea was to create an "energy dashboard." Set up sensors on everything that used or created electricity. The approach involved: o Talking to people to understand systems on the boat, how they connect. o Connecting sensors to a single display, via wired network, on a laptop. o Installing receiver for Internet connection on Mount Washington (a hill in the city of Pittsburgh). However, ran into problems such as too much moisture at original location at Station Square, signal attenuated. o Internet connection was to allow them to stream data to students' classroom and accumulate a log. o Created an informal narrative of use cases for this. Conceptual design was to look over boat and observe activities. o Installed cameras on the microscopes so others could see plankton and hand held cameras for birds, plankton, and turbidity instrument on deck. o Wiring things up caused them to make holes in bulkhead, which was a Coast Guard inspection problem. o Worked out ways to manually record and timestamp some data, e.g, turbidity readings. • In another project, he worked with Blueroof (the instrumented houses in the suburbs of Pittsburgh where Bosch was also a partner in the project) to find ways to infer activities from non-video sensors. For instance, identify when people were in bed, when the front door is open, when water is left running on the faucet, when the stove is on, or when the pillbox is open. o Selected minimal set of sensors: telephone, coffee maker, and pillbox. Focus was: what can you infer from these sensors? They collected 3 months of historical data in order to explore how to automatically interpret and extract information and patterns from the data. o An obvious key challenge was collecting ground truth data. Observation or self-report were the two methods used. Another alternative was to use a script where you have people run through a number of activities and develop the test data. o A key message: "You'd always like to get ground truth marked set of data. Usually over-sense at first, see what is best discriminator." o Sometimes can use a "collaboration of sensors" to recognize activities and develop training sets. o For some applications, there is a need for an interface where a professional (e.g., doctor or physical therapist) can tailor program to a user. o Another key challenge is how do you retrain algorithms when something changes (e.g. new sensor, new data types). It is typically computationally expensive to do on devices. The notion of cloudlets could be of help here.

Interview #5: Customer Representative	
Responsibilities	• Monitors HVAC daily and identify potential problems • Investigates further if necessary • Call facility management services (FMS) to report
Insights from Interview	• Numerous IT systems are involved in the management of the facilities: o A paradox relational database to track phones, keys, room assignments, square footage, and configuration of furniture. This database includes the data about the current state of 12 buildings. ▪ This database is merged with CS Gold, university system for keeping track of people and updating accordingly. ▪ Police and emergency can get access. Floor marshals for fire are in DB o Trouble ticket system is tied into financial system (internal billing). o Oracle calendar is used for space allocation. • Numerous activities are involved in terms of managing facilities and interacting with those IT systems. For instance, look up occupants, give access to labs, education labs, robotics labs, change data when moving offices, phone problems (look up phones), identify all graduate student or faculty offices, and change schedules for doors. • Many of these activities are related to policies which sometimes are formal and sometimes are just reasonable person.

Interview #6: System Programmer	
Responsibilities	• Does programming for lighting and HVAC • Uses "blackbird" to set times for lights to go on and off • Rescheduling for efficiency
Insights from Interview	• The lighting system in the Gates building is challenging to program. For instance, room numbers are not visible in the interface, therefore must map between IDs and room numbers. The vendor provides lifetime programming free. • Has done some programming remotely from iPhone using Team Viewer tool. • Extended existing from pre-programmed displays to provide "color play" on buildings, such as Hunt library building. • Troubleshooting typically involves reviewing logs to see when problem started (e.g., humidity out of range, temperature out of range). Sometimes needs to compare logs of two different systems to troubleshoot. The correlation between logs needs to be done manually. • Rebooting the systems is quite complicated, and it looks like the processes tend to combine manual and automated sequences. The vendor for the HVAC system (Johnson Control) does all sequence changes. • Historical data are kept for several years (e.g. 3 years), which are used for trending. • The Mango tool is used to do power management on "hot" days. • Also, it is required to turn off fire alarm sensors in particular areas when construction is going on. Keeps track on note pad what sensors are turned off when/where. • Incompatibility among protocols is a significant problem. For instance, Etan and MODBUS, and converter hardware are very expensive.

Interview #7: System Programmer	
Responsibilities	• Develops features for assistive living systems • Develops ways to install sensors
Insights from Interview	• There are some interesting challenges in the context of assistive living systems, such as privacy concerns and installation. For instance, end-users many times cannot self-install, and installation may have very high failure rate. • In one project, he installed sensors on water pipes in basement of a house to detect type of activity using machine learning models. Some examples include: (a) is water running, then (b) classify activities, and (c) identify actual device. • As is the case in many of this type of endeavors, there are some important challenges, such as the need for time-stamped data for training, obtaining ground truth data (and in large amounts), and labeling data based on activities (e.g. you want the data to be based on activities – e.g. washing, grooming, cooking – not device based). o Great to get users to label somehow, also ability to query periodically to ask about activities. Crowdsourcing labeling would be huge win, however it poses privacy issues, • Another challenge occurs when there are changes in sensor because a change requires the classifier to be retrained. Migrating models is a hard problem and having a mechanism (e.g. a platform) that supports that process would be very valuable.

Interview #8: Project Manager	
Responsibilities	• Same as senior project manager but for smaller-scale projects • Support senior project manager • Important responsibility: keep track of schedule
Insights from Interview	• The first step is the feasibility study, in particular, the science and the needs. Then, the next step is working out the requirements: o Run through scenarios. For instance, what is the best you could do? What is actually possible? What are the tradeoffs? Where it could fit on campus? o Concerned with things like pressure rations, temperatures, humidity. o Look at the code-related requirements (e.g. building codes, safety codes, professional society codes). o Identify known requirements for types of space, for instance, office space, chemistry labs (more complex and expensive), and bio labs (even more expensive). o Campus guidelines dictate some things, e.g., noise, temperature. • The next step is design, which involves a wide range of activities (from the PM perspective). The process is very fluid until the request for bids is submitted and final specifications are worked out. Examples of activities in this step include: o Face-to-face meetings for reviews and planning with stakeholders o Coordinate with A&E to get preliminary design • The evaluation of bids includes scoping checklist to make sure all submitted bids are proposing the same thing. Once bids are received, the process of looking for brands of equipment starts. Typically, one brand is selected, but they "keep their minds open." Equipment suppliers bid to mechanical contractor and general contractor, as well as to CMU. • Vendors do not want their system to control others. They are typically agreeable to receive data, and if there is a problem, then raise an alarm. However, concerns about liability are very high in the case of controlling some particular devices or subsystems. • Close engagement with commissioning agent, who goes through all documentation and performs tests. When done and approved, the building is ready to occupy. • Some of the difficult problems that are identified prior to occupying the building come from "assumptions" vendors make about their context. For example: system from Sweden wouldn't operate properly in positive pressure room, as a result it produced bad smell. It took 6 months to identify the source of the problem. • During the design and construction process, a large number of change requests are generated due to design and field conditions. In the Gates building, there were about 100 requests that involved about $1,000,000. • Integrated system would be great. For instance, see everything about a room, then if there is a problem drill down into the data of each of the systems. However, it is very hard to get systems to interoperate. • Software updates are another big problem. When the systems are first installed, you know what the algorithm is. Once they are in operation, the parameters and configuration change since programs (or scripts) are added, so when updating the SW, you no longer know what it is doing. The only way to tell is through logged data. Ideally, you would like to have a hardware simulator to see if

	software works. • Accountability when integration doesn't work is another huge problem. It typically causes lots of aggravation and delay. For instance, you don't pay the mechanical contractor until they demonstrate that the graphics and programming work. Would like to get to point where what you are buying is measurable, but none of the vendors are there yet. • "Interconnectivity, reliability, and simplicity are the key."

Interview #9: System Tester	
Responsibilities	• Designs test cases for large-scale HVAC systems
Insights from Interview	• Testing requires learning about the systems (the Bosch products), especially the physical properties and requirements, as well as the possible configurations and potential ways of installing the systems. • The test cases focus on normal operational ranges, borderline test cases, and in some cases, outside operational range testing. • Coordination with the development organization is required when a defect or an unexpected result is found. • Marketing defines the standard configurations and they are tested as part of a product development process • Testing of configurations that marketing will not directly offer is also done. For example, sometimes customers (typically the installer companies) may use the boiler and the control module from one vendor and other parts of the systems (e.g. solar panel for water heating) from a different manufacturer. So testing of those configurations is needed. • Over years, developed experience about some particular combinations that can be more complicated so tend to focus on those. • ”Complicated configurations” could occur, and one example is the case where the customers install all Bosch products except for the boiler. Another example is when customers use the Bosch system for the main system and then use different manufactures for each unit in the building. These systems use common protocol but a lot of times there are problems. • Example: there was one case where the user would complain that the commands took a long time to respond and initiate the operations (e.g. starting the boiler). After a lot of investigation, they found that the message exchange was ok but the timeout configuration was not correct for the setup. The main controller would take too long to poll the components so the console would expire the message and resend it, so the tests that they do try to focus on things like this. • Requires numerous types of devices from different vendors to test “special” configurations.

Interview #10: Architect	
Responsibilities	• Responsible for overall concept design and detail design
Insights from Interview	• Architects are involved in some aspects of the technical aspects of the various subsystems. In particular, the involvement is focused on trying to make sure that everything fits the general design and the customers' requirements. o Example: sometimes the selection of a particular heating system may be the best choice for that department, but it may not fit the expectations of the design or the expectations of other system like energy management. • Negotiations are required with engineers regarding design choices and constraints of the site or the equipment. o Example: the temperature sensors were placed on the outer doors. The thermodynamics engineer said that the temperature on the windows side would be higher than in the far wall area because of the dimensions of the space. The heating engineer placed the sensor there to save wiring and comply with building regulation. This issue would result in the windows being activated too late and potentially getting the space too warm before the air ventilation systems starts working. • Lots of effort goes into the review process.

Interview #11: Application Developer	
Responsibilities	• Development of iOS application • Interaction design • Client consulting
Insights from Interview	• Experience in a wide range of apps from manufacturing oriented to consumer-based. • In one project, factory app, the goal was to implement a mobile interface to an existing production control system. The mobile app required an interface to a Windows server via a .NET based protocol, which is not readily available in existing mobile platforms (e.g. iOS or Android). o The app was targeted to support decision-making, primarily for people running factories. Example: foam prices vary throughout the day, so app allows you to check the cost of raw materials and figure out if the day made effective use of materials. o The mobile platform did not control production machines, it just dealt with the "data" manipulation features. It did have a "shutdown production" option. o Development process: ▪ Client has idea and comes to him with "We want to do this…" ▪ Sit down with client and brainstorms with stickies. ▪ Write all features of existing web interface on stickies and a few extra features that could be useful. ▪ Remove stickies that were not necessary for mobile app. ▪ Run sample UI by the client for feedback, which is a Photoshop prototype with basic UI elements. Asks if anything is missing or if there is a reason it needs to be changed. ▪ Create user interface with basic functionality to test basic features, such as server connection to get base communication established. ▪ Use simulator and iterative process in implementing UI and features. ▪ Use extensive unit testing and writes test cases before implementing features. Testing done on test server with a copy of the actual data. Beta version thoroughly tested a local testing for 1.5 months running on actual servers in parallel with normal control processes before given to clients. When things pass, app is gradually rolled out to a few clients, who run it for 6-7 months before the final rollout. • Feedback on the pros and cons of SDKs for mobile platforms: o Apple does not give you control over a lot of low level stuff that other platforms give control of. o Simulators have some valuable capabilities for testing but are typically not sufficient to fully test apps. In addition, there are constraints in terms of the capabilities of each device in the simulator that render them useless when the app uses those capabilities. o In platforms such as Android, the diversity of devices creates serious challenges in terms of testing across all types of supported devices – typical approach is to limit the "supported" variants. o Platforms, such as Android, give the developer a lot of flexibility and control on the access to resources (e.g. photos, files, etc). Conversely, iOS is very restrictive. Developer indicates what it needs, and the platform takes care and returns results. o SDK has some built in classes to help with test cases, which helps validate input and provides

	methods for assertion testing. • Apple uses the concept of sandboxes around apps to protect the platform from the apps. This notion could be extended with some sort of dynamic boundaries to allow inter-app communication or access to shared data. Then, “dynamic sandboxing” as a technique should be explored.

Interview #12: Lead System Specialist	
Responsibilities	• Supports sales by evaluating a bid and proposing solution • Designs a system once a contract is awarded
Insights from Interview	• Proposed solutions may involve one or more systems (e.g. HVAC, hot water, lighting, etc.). • The use of software tools to design proposed systems is common but experience plays an important role in terms of anticipating issues (e.g. anticipating changes which are quite common) • Organization hierarchy is: owner with a resident engineer hires general contractor, who hire sub-contractors (e.g. electrical, mechanical, plumbing, etc). The sub-contractor comes to vendor. There are systems (e.g. temperature control) that are "above" all this so vendor also interacts directly with the general contractor. The vendor may further interact directly with owner because the customer is concerned about aspects such as comfort. • Riser diagrams are typically considered as the "architecture" of the system. The riser diagrams depict electrical wiring, piping, and other physical elements and their interconnections within a building. Riser diagrams typically depict the floors and the controllers for each floor. • There are also decisions about the communication protocol (e.g., BACnet, N2, LON). The decision is part of the specifications, and typically you want just one protocol. BACNet protocol is "standard" within the industry, but each company interprets the standard differently. In his opinion, LON is more standardized and interoperability is more likely than with BACNet. • Fiber is typically used for connections among buildings, so you have fiber going to each controller/panel on each floor. From there, there is support for Cat5, Ethernet, the network automation engine that are connected to routers, which are where you connect the devices (freeze alarm, smoke, temp, isolation dampers, etc). Servers are used for consolidating all the information about the building. The servers are responsible for generating the alarms, the graphics, etc., and they can be accessed remotely (e.g. via an internet connection). • Adding new equipment or devices is full of special cases that require custom programming. Engineers write the new sequence of operations, and they test using simulators. About 95% of defects can be detected before deployment. • The systems provide capabilities to drill down on the relevant information to find all devices connected to some system or part of the system. They use software for generating simple case diagrams for the particular unit, then the engineer customizes for a particular case with the default sequence of operations. Sequences of operations seem to be grouped in modes, such as unit enable, occupied mode, and unoccupied mode • The use of "checkout sheets" is quite common, particularly, in the context of pre-functional check outs. • The window for making sure everything works is quite short. Inspection involves all "eyes and ears" - walking around to make sure everything looks and sounds right.

Interview #13: System Designer	
Responsibilities	• Defines all the pieces that go into a particular system that an installer can set up at a customer site • His involvement starts after the sales process is completed
Insights from Interview	• Different steps are involved depending whether a project is a retrofit or a new installation. On a retrofit job, site visits are always necessary to figure out what is there. In addition, the drawings may be out of date or not available at all. • The basic system diagram has inputs on left, outputs on right, and logic block in between. Each logic block is a program, which contains the operation sequences and are articulated via a graphical language (similar to SimuLink). • In one project, he worked on trying to identify opportunities to do energy savings. The context was a casino building with multiple rooftop HVAC units. The owner has the big picture idea, and the challenge for the project was to find out what extra parts/elements are required for the new functionality. o A good commissioning agent can help here because they tend to have the overall view. • Energy management is behind the Panoptix platform provided by Johnson Control. The main idea is to take advantage of the wealth of data available, put it in a common format, and store it in the cloud. • Panoptix evolved from Grid Logic, which was a company bought by Johnson control. • Panoptic does data normalization, which refers to normalizing timestamps to sync data from different devices and different proprietary databases. • Eventually Panoptix will integrate all data under the technical contractor. • Backward compatibility is an important quality of this context.

4.3. Ecosystem Roles

The individuals we interviewed were representative of a collection of roles that may be an integral part of a future smart building ecosystem. In addition, the content collected in the interviews suggests additional potential contributors in the ecosystem. This section summarizes all the various stakeholders we consider will be an integral part of a future smart building ecosystem, and we describe their main responsibilities. It is worth pointing out that we have constructed generalized categories of stakeholders based on the information from the interviews, such that we can abstract the differences in naming conventions of roles used across companies and organizations.

Role	***Description***
End-user	The resident (e.g. students, faculty) or a visitor to the smart building that will be the main user of the facilities.
Facility Management	The organization in charge of all operational and maintenance aspects of the smart building. Typically, this organization consists of many individuals with different responsibilities, such as operations and maintenance (e.g. of the various systems – HVAC, access control, etc – as well as diagnose problems that main arise), customer support (e.g. receive requests for service from residents) and technical support for construction projects (e.g. new buildings or extension of existing ones)
Building Project Manager	The person (or persons) in charge of managing the construction project of a new smart building from the "owner's" side. This role typically has several responsibilities including the decision to select constructors, system providers, collect requirements from the future residents, and manage the evolution of the project.
Installer	The person (or persons) that installs a new device, piece of software, sub-system, or an entire system in a smart building. A member of a device manufacturer company, an integrator or a sub-contracted company, could represent this role.
Commissioning Agent	This role is responsible for certifying construction and/or operation compliance with certain standards (e.g. LEED – Leadership in Energy and Environmental Design).
Certifier	This person (or persons) provides a wide range of certification services. In essence, this role is a generalization of the commissioning agent. However, the certification services covered by this role extend well beyond compliance with standards and include certifications of new devices, sub-systems, systems, SW functionality, and even new applications.
System Architect Consultant	This role is responsible for understanding how all the various systems in a smart building "fit" together, and, typically, it would have decision-making rights to select components and/or systems, such that the architectural requirements of the overall system are satisfied.
Integrator	This role carries out all the necessary activities to ensure that all the sub-systems are integrated as required. Then, this role differs from the "system architect consultant" role in that this role is responsible for "putting" the pieces together in a smart building system (e.g. design/implementation oriented work) while the "system architect consultant" ensures that all the pieces "can be put together" (e.g. architectural specification oriented work).
Device Manufacturer	These are the producers of devices, appliances, and systems that could be part of a smart building

	(e.g. temperature sensors, movement sensors, cameras, refrigerators, HVAC systems, access control systems, etc)
Device Programmer	This role is responsible for developing new control functionality associated with a new or existing device. Personnel from the device manufacturers, as well as from other organizations, such as service providers or even personnel from facility management, could represent this role.
Application Developer	This role is responsible for developing new software functionality with capabilities that go beyond the sphere of control of sensors and actuators. Example applications could be data visualization, data analytics (e.g. pattern recognition features), and energy management.
Platform Provider	This role refers to the organization that provides a software platform that enables at least (a) some or all of the devices and sub-systems in a smart build to interconnect and (b) the control and state of those devices and sub-system can be managed. Examples of this role include companies like Johnson Control (e.g. the Panoptix platform) and Bosch (e.g. INST's AIP platform).

In the rest of section 5, we will use these role names as part of our analyses.

4.4. Summary of Use-Case Workshop

We also conducted a workshop to develop end-user use cases. The workshop participants included researchers from Robert Bosch Corporate Research, one professor from Carnegie Mellon University, and several undergraduate and graduate-level students. The procedure used for the workshop was the following. The workshop consisted of 3 hours organized in 2 main parts. During the first part of the workshop, the participants generated candidates of use-cases individually. During the second part of the workshop, the candidate use-cases were clustered based on commonalities. Then, the participant were organized into 4 groups, and those groups further developed the clusters of use-cases, refining their definition and classifying them in terms across two dimensions: availability of the necessary technology and effort associated with realizing such the use-cases. The following table summarizes the 6 main use-cases that came out of the workshop and were considered in our analyses for defining functional and non-functional requirements on the software platform for smart building

Use-Case	***Description***
#1	User arrives at smart building/home, and his/her smart-phone is updated with the software necessary to "access" the smart environment functionality.
#2	User requests directions to a location within a particular building, and the system guides him/her through the indoor spaces (this is indoor navigation with augmented reality).
#3	User lost an item (e.g. a bag pack). The building recognizes the object and triggers notification to the Lost & Found service. The users is notified and guided through the building back to the object or to the Lost & Found office.
#4	A user is able to interact seamlessly with an Audio & Entertainment system via a smart-phone (or similar device) to display photos, documents, etc.
#5	Users are in a large room or auditorium, and they are able to adjust temperature, sound and other environmental parameters to fit their needs based on their location throughout the auditorium.
#6	Users need to use a service (e.g. a restroom or buy food), and the building guides him/her to the "right" place depending on particular criteria (e.g. shortest path, shortest wait times, etc).

4.5. Key Quality Attributes of the SW Platforms

The analysis of business and other considerations define qualities of the software system that must be accommodated by its architecture (Bass et al, 2003). These qualities are attributes of the architecture that exist over and above the required functionality of a software system. Bass and colleagues (2003) argued that despite the fact that functionality and other qualities are closely related, development organizations tend to focus primarily on functionality relegating the other qualities. The consequences of such tendencies are most often redesigns of the software architecture that do not involve changes in the functionality of software systems but rather involve modifications to improve maintainability, portability, scalability or performance of the systems. In other words, most redesigns have to do with addressing quality attributes that were neglected in the original designs.

Before going any further, it is important to state some definitions based on Bass and colleagues' work:

- Functionality: "…it is the ability of the system to do the work for which it was intended…functional requirements define a particular set of behaviors that the system must have…"
- Quality Attribute: "…it specifies criteria that can be used to judge the operation of a system, rather than specific behaviors…"

Functionality and quality attributes should be considered as orthogonal (Bass et al, 2003). The same piece of functionality may be achieved by using a wide range of architectural structures. Then, arguably, functionality is largely independent of structure. The choice of a particular software architecture constrains the allocation of a piece of functionality to a structure when other quality attributes are also relevant. However, achieving quality attributes is not just a design-time consideration, but it should also be part of the implementation and deployment. No quality attribute is entirely dependent on design, nor is it entirely dependent on implementation or deployment (Bass et al, 2003). Some of the key take-away points made by Bass and colleagues (2003) in regards to quality attributes are:

- The software architecture of a system is critical to the realization of the qualities attributes of a system.
- Those qualities should be designed into the system and can be evaluated at the architectural level.
- The software architecture simply provides the foundation for achieving quality attributes.
- In complex software systems, quality attributes are rarely achieved in isolation.
- Therefore, architectural trade-offs will exist and must be understood and managed.

The interviews combined with the use-case workshop and analyses of technology trends have provided us with sufficient context to identify a collection of basic quality attributes that future software platforms for the smart building domain should support. The following table describes the six key quality attributes that we identified. For each quality attribute, we provide pointers to the source of evidence. For instance, several interviews point to the need to add new devices to the system, as well as, the significant costs associated with doing so. Therefore, extensibility becomes an important quality for future systems.

Quality Attribute	*Description*	*Evidence*
Extensibility	The ability to add new capabilities (e.g. new software function, new devices) with minimal impact to the system	Interviews 3, 6, 8, 11 and 12
Heterogeneity	The ability to support a wide range of types of devices in the system, as well as, the ability to support a wide range of roles that may be involved in the ecosystem	Interviews 1, 3, 5, 7, 8, 9, 10, 12 and 13 Use-cases 2, 4 and 5
Security/Privacy	The ability of the system to restrict access to functionality and data by unauthorized persons (security) and to operate and manage data in a privacy-preserving manner	Interviews 4, 5, and 7 Use-cases 1 and 3
Scalability	The ability of the system to manage growth in the amount of data generated and made available to the system, as well as, growth in the number of types of data (e.g. time series, streaming, unstructured, etc)	Interviews 4, 6, 7 and 13 Use-cases 2, 3 and 6
Certifiability	The ability of the system to support the certification, test, validation, and verification of new capabilities (e.g. new app, new functionality in the platform, upgrade of existing functionality, new devices, etc.)	Interviews 3, 6, 8, 9, 11 and 12
Diagnosability	The ability of the system to support the diagnosing of problems within the system (e.g. unusual temperature readings) as well as traceability and auditability.	Interviews 1, 2, 6, 8, 9 and 11

It is worth pointing out that the quality attributes listed in the table above are not the only ones that matter. In fact, traditional quality attributes such as availability, performance, and usability are also quite important. However, the goal of this section is to bring attention to specific quality attributes that particularly impact a SW platform for smart building and smart campus domains.

4.6. Analysis of Scenarios

In order to ensure appropriate consideration of quality attributes in the architecture of future software platforms for smart building and smart campus domains, we developed a collection of scenarios that illustrate one or more quality attributes. We structured the information of each quality attribute scenario based on the work of Bass and colleagues (2003). They defined quality attribute scenario as "… a quality-attribute-specific requirement...". The authors suggested that six pieces of information constitute a quality attribute scenario:

- *Source of stimulus* is some entity that generates a stimulus.
- *Stimulus* is a condition that needs to be considered when it arrives at a system.
- *Environment* is the set of conditions within which a stimulus occur.
- *Artifact* is the part of the system that is stimulated. This may also be the whole system.
- *Response* is the activity undertaken after the arrival of the stimulus.
- *Response measure* represents the "measurable" properties that occur after a response and allows the requirement to be tested.

Traditionally, the analysis of quality attribute scenarios has focused on articulating the six pieces of information that constitute the scenario. However and when considering the ecosystem context and the particular domains of interest (smart buildings/smart campus), we thought it was necessary to make some minor modifications in order to reflect the fact that we are focused on future systems that have almost no resemblance to current systems. Therefore, we replace the "*environment*" data element with "*situation today*" to capture the key challenge in today's systems that should be addressed in systems of the future. Furthermore, we added to the list of elements in each scenario in order to capture other important pieces of information. Specifically, we added three additional elements:

- *Roles that care about the scenario* refer to the set of roles in the ecosystem that may be impacted by the quality attribute or may be involved in the scenario.
- *Architectural implications* refer to particular considerations (e.g. functionality, structure, styles) that may be relevant to the quality attribute.
- *Architectural trade-offs* refer to the set of options that may exist and articulate the quality attribute in the software architecture of the platform.

Combining these additional elements with the traditional ones provides us with a broader understanding of the scenarios, the quality attributes, and the architectural considerations that should be carefully taken into account.

The rest of the section describes the 16 scenarios that were developed as part of our analyses.

4.6.1. Scenario 1

Description	**Add a New Device**
Illustrates Quality Attribute	Extensibility, Heterogeneity
Source of Stimulus	Desire to add a window sensor in every room that has a window that can be open.
Stimulus	Stakeholder's need
Situation Today	At design time, adding a device has an approximate cost of $1000
Target Artifact	All the control systems that will interact (e.g. temperature control)
Response	Sensor is added and integrated into the system (e.g. new behavior: if the window is open, heating will not take place).
Response Measure	Cost of addition $100
Role that cares	Facility management, building project manager, installer, integrator, device manufacturer, device programmer, system architect consultant
Architectural Implications	The SW platform needs to provide mechanisms for: • …integrating the control functionality of the new device seamlessly • …integrating the collection, storage, processing and access to the data generated by the new device • …controlling access to the new data in a way that is consistent with existing policies and flexible enough to allow the data to be used in novel new ways • …enabling HW device simulation for testing, validating and certifying the control functionality prior to the deployment of the physical device
Architectural Trade-off	Pushing the responsibility of supporting a standardized protocol onto the device manufacturers (a la Control4 or Android@Home) ***VERSUS*** Providing support for multiple protocols in a "gateway" type of device Note: this trade-off would apply to some types of devices however for others (e.g. window sensor) it is not feasible or practical at all to "force" the manufacturer to support some particular protocol and or APIs

4.6.2. Scenario 2

Description	**Add a new visualization for historical data for managing energy**
Illustrates Quality Attribute	Extensibility, Usability
Source of Stimulus	Desire to add a new type of visualization of the energy consumption historical data from the all the building's systems.
Stimulus	Stakeholders' need.
Situation Today	Assuming the availability of the data, they need to be collected manually from each system based on current "ownership" rules. Then the data needs to be normalized and stored. Access control mechanisms need to be put in place. Finally, the new visualization can be developed and deployed. The effort associated in these various steps makes the addition of this new type of functionality quite costly.
Target Artifact	All the systems in the building that "generate" energy consumption data.
Response	New visualization is added and integrated into the system.
Response Measure	The effort of development and integration is less than 40 hours.
Role that cares	Facility management, device programmer, app developer, system architect consultant
Architectural Implications	The SW platform needs to provide mechanisms for: • …integrating the new SW functionality (in this case the visualization) seamlessly • ...integrating the collection, storage, processing and access to the data generated by the devices/systems • ...controlling access to the data in a way that is consistent with existing policies and flexible enough to allow the data to be used in novel new ways • ...enabling testing, validating and certifying of the new SW functionality prior to its deployment
Architectural Trade-off	Providing a general capability for storing, accessing, and processing data that allows potential users to abstract the specifics and/or low level aspects of the data. ***VERSUS*** Providing specialized capabilities for each "type" of data. Note: where "type" could be characterized in terms of traditional data type (time series, stream data, etc.), in terms of the type of system, or in terms of the information it refers to (energy consumption, temperature, movement, geographical, etc.)

4.6.3. Scenario 3

Description	**Add the capability of analyzing historical data from all energy consuming devices in a secured and privacy-preserving manner**
Illustrates Quality Attribute	Extensibility, Heterogeneity, Security/Privacy
Source of Stimulus	Desire to add an analytical capabilities that uses energy consumption historical data such that the privacy concerns of the end-users are protected.
Stimulus	Stakeholders' need
Situation Today	Similarly to scenario #2 and assuming the availability of the data, they need to be collected manually from each system based on current "ownership" rules. Then, the data needs to be normalized and stored. Access control mechanisms need to be put in place. Finally, the new visualization can be developed and deployed. The effort associated in these various steps makes the addition of this new type of functionality quite costly.
Target Artifact	All the systems in the building that "generate" energy consumption data.
Response	New analytical capability is added and integrated into the system.
Response Measure	The effort of integration is less than 40 hr. It is worth pointing out that the effort associated with the analysis may differ greatly depending its type, so the measure is focused on the effort associated with integrating the functionality on the existing system.
Role that cares	End-user, facility management, device programmer, app developer
Architectural Implications	The SW platform needs to provide mechanisms for: • ...integrating the new SW functionality (in this case the visualization) seamlessly • ...integrating the collection, storage, processing and access to the data generated by the devices/systems • ...controlling access to the data in a way that is consistent with existing policies and flexible enough to allow the data to be used in novel new ways • ...enabling testing, validating and certifying of the new SW functionality prior to its deployment
Architectural Trade-off	Trade-off #1: Providing a uniform access control mechanism ***VERSUS*** Providing specialized or tailored access control capabilities for (a) each "type" of data (lots of nuances associated with the type of data e.g. heating info versus location) or (b) that can be tailored/defined by the data "provider"/"owner" or (c) tailored/defined by the application developer Trade-off #2: Owner retains control of the data and decides to share them or not ***VERSUS*** Owner defines usage/access policies and is able to certify them

4.6.4. Scenario 4

Description	**Identification of the location of an individual is only possible to "administrative" personnel**
Illustrates Quality Attribute	Security/Privacy, Usability, Certifiability
Source of Stimulus	Desired to minimize (ideally eliminate) the misuse of location data generated by systems like surveillance cameras, GPS-enabled devices or in-door localization mechanisms in order to protect the privacy of occupants (permanent or temporary) of smart building.
Stimulus	Stakeholder's need and legislative requirement (potentially in some regions).
Situation Today	Policies are defined on case-by-case basis and, typically, manually enforced via processes.
Target Artifact	All subsystems that generate location-disclosing data.
Response	Automatically enforce access control on "location"-disclosing data.
Response Measure	Given a piece of "location-disclosing" data, it is possible to determine which roles (and individuals) have access to it.
Role that cares	End-user, facility management, building project manager, commissioning agent
Architectural Implications	The SW platform needs to provide mechanisms for: • ...providing the basic framework for role-based access control to the data • ...describing policies that define the context under which the privacy constraints may apply (e.g. private versus public spaces, classes of devices) • ...tracing/logging access and usage of the relevant classes of data
Architectural Trade-off	Providing a uniform access control mechanism ***VERSUS*** Providing specialized or tailored access control capabilities for (a) each "type" of data (lots of nuances associated with the type of data e.g. heating info versus location) or (b) that can be tailored/defined by the data "provider"/"owner" or (c) tailored/defined by the application developer

4.6.5. Scenario 5

Description	**Add a classifier that uses existing data and generate new data that is used in data-triggered control**
Illustrates Quality Attribute	Extensibility
Source of Stimulus	Need to add a new classifier algorithm that analyzes existing data (historical and/or currently generated) and generates new (summarized) data that is used to control a particular system.
Stimulus	A vision of the future / a potential stakeholders' need
Situation Today	Systems do not provide this type of capabilities, however, research in the area of assistive leaving and home automation have demonstrated their significant value.
Target Artifact	All the systems in the building that could use "data-triggered" control mechanisms.
Response	The algorithm is added and integrated into the system.
Response Measure	The deployment of the new algorithm does not impact the operation of the system.
Role that cares	Facility management, installer, integrator, device manufacturer, app developer
Architectural Implications	The SW platform needs to provide mechanisms for: • ...integrating the new SW functionality seamlessly • ...integrating the generated data by the new SW functionality seamlessly into the control functionality of the relevant system • ...controlling access to the data in a way that is consistent with existing policies and flexible enough to allow the data to be used in novel new ways • ...enabling testing, validating and certifying of the new SW functionality prior to its deployment
Architectural Trade-off	Providing a uniform extension mechanism (e.g. a la Eclipse plug-in) ***VERSUS*** Providing a specialized approach tailored to specific types of functionality or types of data accessed/generated

4.6.6. Scenario 6

Description	**The building timely notifies of some event X**
Illustrates Quality Attribute	Usability, Certifiability, Diagnosability
Source of Stimulus	Desire to automatically get notifications of certain events on a timely manner.
Stimulus	Stakeholders' need
Situation Today	Notifications are handled separately by different systems depending on their type/nature. Emergency-related notifications (e.g. fire alarms) are handled by a separate system that is highly regulated and audited by government agents. Other types of notifications (e.g. out-of-range temperatures) are managed by their corresponding systems and they do not tend to be effectively handled.
Target Artifact	All the subsystems that generate notifications.
Response	Events are notified to the relevant target audience.
Response Measure	Notifications are delivered within X seconds of the relevant event occurring. Note: the time to deliver the notifications might be specified by some legal requirements (e.g. for fire alarms that may required defined value by the regulating body) or by some policy.
Role that cares	End-user, facility management, commissioning agent, certifier
Architectural Implications	The SW platform needs to provide mechanisms for: • ...registering and specifying events and notifications (e.g. something along the lines of a service-level agreement with particular constraints depending on the type of event) • ...managing the potential interactions among events or notifications and other subsystems (e.g. a hierarchical scheme of alerts as NASA uses for the space state to manage complexity and "volume")
Architectural Trade-off	Providing a uniform mechanism for defining events and triggering the notifications (e.g. something that can be standardized as DB triggers) ***VERSUS*** Providing a more specialized approach that is tailored to the specific nature of the subsystems

4.6.7. Scenario 7

Description	**Use motion detectors to verify that an evacuation has been taken place**
Illustrates Quality Attribute	Extensibility, Certifiability, Security/Privacy
Source of Stimulus	Need to confirm that a building has been evacuated after a relevant notification (e.g. fire alarm) has been issued.
Stimulus	A vision of the future / a potential stakeholders' need
Situation Today	Confirming that a building has been evacuated involves visiting all the rooms in person which is time consuming and, in some situations, it could even be infeasible.
Target Artifact	All subsystems involving motion detection (e.g. sensors or cameras) and the notification subsystems.
Response	An "evacuation completed" notification is issued.
Response Measure	Evacuation is confirmed within the time specified in the evacuation procedures and the evacuation confirmation can be certified by the corresponding government agents (as they would normally do for current systems).
Role that cares	Facility management, commissioning agent, certifier
Architectural Implications	The SW platform needs to provide mechanisms for: • ...registering and specifying events and notifications (e.g. something along the lines of a service-level agreement with particular constraints depending on the type of event) • ...enabling testing, validating and certifying of (a) the detection of living entities in the building's public and private spaces, (b) the delivery of the notification upon (c) completion • ...tracing/logging access and usage of the relevant classes of data • ...controlling access to the data (e.g. generated by motion detection/cameras) in a way that is consistent with current policies, it is privacy preserving and flexible enough to allow the data to be used effectively
Architectural Trade-off	N/A

4.6.8. Scenario 8

Description	**Add 500 security cameras that generate video stream and metadata without degradation of the system**
Illustrates Quality Attribute	Scalability (of data)
Source of Stimulus	Desire to add a large collection of devices (e.g. 500 cameras) that generate a significant amount of new data to an existing smart building system without degrading the operation of the system.
Stimulus	Stakeholders' need
Situation Today	The typical situation in today's world would require several steps: (a) acquire the necessary new storage capacity, (b) make that new capacity available, (c) defined and implement the storage mechanisms (e.g. DB definition, etc), (d) defined and implement the processes (e.g. scripts) to collect the streams generated by the cameras and store them into the database.
Target Artifact	All subsystems interacting with new devices or using data generated by new devices
Response	The 500 cameras are installed and become operational.
Response Measure	The degradation in operational performance of any related SW functionality (e.g. pattern identification algorithm) cannot exceed X% of pre-installation performance. Note: the percentage of degradation can be based on a wide range of factors.
Role that cares	Facility management, integrator, installer, device manufacturer, device programmer, app developer, system architect consultant
Architectural Implications	The SW platform needs to provide mechanisms for: • ...integrating the new SW functionality (in this case the visualization) seamlessly • ...integrating the collection, storage, processing and access to the data generated by the devices/systems • ...controlling access to the data in a way that is consistent with existing policies and flexible enough to allow the data to be used in novel new ways • ...enabling testing, validating and certifying of the new SW functionality prior to its deployment
Architectural Trade-off	Trade-off #1: Providing a general capability for data storage, accessing and processing that allows potential users to abstract the specifics and/or low-level aspects of the data. ***VERSUS*** Providing specialized capabilities for each "type" of data. Trade-off #2: Providing an extensible storage environment (e.g. cloud-based or single-naming-space) for all the raw data ***VERSUS*** Providing an approach that pre-processes the raw data, stores the resulting data and offload the raw data to a secondary storage Note: where "type" could be characterized in terms of traditional data type (time series, stream data, etc.), in terms of the type of system, or in terms of information it refers to (energy consumption, temperature, movement, geographical, etc.)

4.6.9. Scenario 9

Description	**A new application that can assist on moving a person from one office to another is installed**
Illustrates Quality Attribute	Heterogeneity, Security/Privacy
Source of Stimulus	The desire to make the process of moving a person's office within or between buildings more efficient.
Stimulus	Stakeholders' need
Situation Today	The process of moving a person's office consists of a collection of checklists that are manually considered by facility management personnel. The process is further complicated by the multitude of departments that are involved (e.g. responsible people for telephone, computer connectivity, moving furniture, moving and reinstalling computers, keys, moving preference in heating systems, etc). These activities are highly error prone.
Target Artifact	All relevant subsystems including the IT related system that keep track of, furniture inventory, telephone assignments, keys within the same building or between buildings, access control, etc.
Response	Application is deployed.
Response Measure	Application is able to assist in a move of person's office across buildings satisfying all current checklists and the move occurs without errors and delays.
Role that cares	End-user, facility management, system architecture consultant
Architectural Implications	The SW platform needs to provide mechanisms for: • ...interfacing (e.g. share/request information) from another building's platform • ...interfacing and coordinating with existing IT-related systems (e.g. furniture inventory, access control, key-tracking, telephone/communications, etc)
Architectural Trade-off	N/A

4.6.10. Scenario 10

Description	**The HVAC manufacturer develops a new version of the software and wants to upgrades the version running in the building**
Illustrates Quality Attribute	Extensibility, Diagnosability
Source of Stimulus	The desire to upgrade the software platform of the HVAC system.
Stimulus	Stakeholders' need
Situation Today	Typically, the manufacturer performs the upgrade which typically involves shutting down (partially or totally) the system, and replacing the HW that contains the image of the software (e.g. in some EPROM/ROM device). The rollback procedures involve putting the old HW back.
Target Artifact	The HVAC system (or more generally, the target SW piece to be upgraded).
Response	The software upgrade is deployed.
Response Measure	The upgrade is successful and no interdependent SW and control functionality is non operational as a result of the upgrade.
Role that cares	End-user, facility management, device manufacturer, installer, app developer
Architectural Implications	The SW platform needs to provide mechanisms for: • ...integrating the new SW functionality (in this case the upgrade) seamlessly • ...integrating the collection, storage, processing and access to potential new data generated by the upgraded software • ...controlling access to the data in a way that is consistent with existing policies and flexible enough to allow the data to be used in novel new ways • ...enabling testing, validating and certifying of the upgrade prior to its deployment
Architectural Trade-off	N/A

4.6.11. Scenario 11

Description	**A temperature sensor becomes faulty and the building is able to setup a repair activity within 24 hours**
Illustrates Quality Attribute	Diagnosability
Source of Stimulus	The need to efficiently diagnose a problem, identify the source of the problem and take the necessary steps to resolve the problem
Stimulus	Stakeholders' need
Situation Today	Diagnosing and identifying the source of a problem in any of the systems (e.g. HVAC) is a major pain point for facility management and service/maintenance personnel (as highlighted by several interviews). Some of the key sources of the "pain" are the lack of relevant data, the difficulty of correlating relevant data because how disperse those data may be across the various systems, and the lack of interoperability between systems.
Target Artifact	All subsystems
Response	The faulty temperature sensor is identified and a repair is scheduled with the corresponding provider.
Response Measure	The faulty temperature sensor is identified and a repair is scheduled with the corresponding provider all within 24hrs.
Role that cares	Facility management, installer, integrator, device manufacturer
Architectural Implications	The SW platform needs to provide mechanisms for: • ...representing the "state" of a sensor/device as an explicit part of the state model (difference between model of the sensor and sensed values) • ...diagnosing sensors/devices (e.g. redundancy, voting, look for anomalous patterns in the data generated by the sensor/device, etc) • ...integrating the new SW functionality (in this case the upgrade) seamless • …representing and mapping about physical entities of the system to geographical information (e.g. where a device is located)
Architectural Trade-off	Providing fail-safe operational mode ***VERSUS*** Providing a "failure" operational mode

4.6.12. Scenario 12

Description	**A tester or certifying person (e.g. commissioning agent) is able to run through the "checkout sheet"**
Illustrates Quality Attribute	Certifiability
Source of Stimulus	The desire to support the testing/certification process (and make it more efficient).
Stimulus	Stakeholders' need
Situation Today	The usage of "checkout sheets" to aid in the pre-functional checkups or certification checkups is quite common. A specialist, a tester or certifier utilizes those sheets as reminders of steps to follow or things to verify and/or inspect.
Target Artifact	All subsystems.
Response	The system supported the verification/certification process specified in a particular "checkout sheet".
Response Measure	The verification/certification process takes X% less effort. Notes: the reduction in effort is a parameter that can be defined based on a wide range of factors.
Role that cares	Facility management, building project management, commissioning agent, certifier, app developer
Architectural Implications	The SW platform needs to provide mechanisms for: • ...representing workflows (e.g. "checkout sheets") of parts of a system to verify and their associated states • ...enabling HW device simulation for testing, validating and certifying the control functionality of the various devices and/or subsystems • ...tracing/logging relevant classes of data
Architectural Trade-off	Providing test-safe operational mode ***VERSUS*** Providing testing capabilities on live or shutdown system

4.6.13. Scenario 13

Description	**A resident of a smart building should be able to know what data the building has stored about the resident and which applications use the resident's data**
Illustrates Quality Attribute	Security/Privacy
Source of Stimulus	The desire to have awareness and understanding of who, how and when particular pieces of data generated by the "devices" in a smart building are used.
Stimulus	Stakeholders' need
Situation Today	Typically, systems have very coarse-grain mechanisms for controlling access and/or ownership of the data generated by the devices and systems. For instance, HVAC systems use an authentication mechanism to control access to temperature readings across a building. Once the user gains access, all the data is available without any control from the end-users that may be generating some of those data.
Target Artifact	All sub-systems.
Response	Upon interaction with the system, the resident gets a list of all the types of data being stored and which SW functionality uses those data.
Response Measure	The residents are able to certify that the response in properly executed.
Role that cares	End-user, facility management, system architect consultant
Architectural Implications	The SW platform needs to provide mechanisms for: • ...controlling access to the data generated by the various devices/subsystems in a way that is consistent with current policies, it is privacy preserving and flexible enough to allow the data to be used effectively • ...providing the basic framework for role-based access control to the data • ...describing policies that define the context under which the privacy constraints may apply (e.g. private versus public spaces, classes of devices) • ...tracing/logging access and usage of the relevant classes of data
Architectural Trade-off	Providing a uniform access control mechanism ***VERSUS*** Providing specialized or tailored access control capabilities for (a) each "type" of data (lots of nuances associated with the type of data e.g. heating info versus location) or (b) that can be tailored/defined by the data "provider"/"owner" or (c) tailored/defined by the application developer

4.6.14. Scenario 14

Description	**The location of a person conditions the applicability of rules/policies**
Illustrates Quality Attribute	Heterogeneity, Security/Privacy
Source of Stimulus	The need to apply privacy rules/policies contingent on the geographical location of a building resident.
Stimulus	A vision of the future / a potential stakeholders' need
Situation Today	Systems do not provide these capabilities. They are typically articulated as some sort of policy for devices such as cameras. We envision a smart building that is able to determine the location of a particular resident/visitor within any of the spaces (e.g. private or public) and apply the privacy rules/policies on any type of data generated by the system contingent on person's location.
Target Artifact	All subsystems that generate data with privacy implications.
Response	Identified the location of a particular resident/visitor, the generated data is treated in a way consistent with the relevant privacy rules/policies.
Response Measure	The correctness of the functionality is certifiable.
Role that cares	End-user, facility management, system architect consultant
Architectural Implications	The SW platform needs to provide mechanisms for: • ...controlling access to the data generated by the various devices/subsystems in a way that is consistent with current policies, it is privacy preserving and flexible enough to allow the data to be used effectively • ...providing the basic framework for role-based access control to the data • ...describing policies that define the context under which the privacy constraints may apply (e.g. private versus public spaces, classes of devices) • ...tracing/logging access and usage of the relevant classes of data • …providing geographical information services
Architectural Trade-off	Providing a uniform access control mechanism ***VERSUS*** Providing specialized or tailored access control capabilities for (a) each "type" of data (lots of nuances associated with the type of data e.g. heating info versus location) or (b) that can be tailored/defined by the data "provider"/"owner" or (c) tailored/defined by the application developer

4.6.15. Scenario 15

Description	**A company guarantees that their system will generate X amount of savings and the facility owners are able to verify that is the case**
Illustrates Quality Attribute	Certifiability
Source of Stimulus	The need to verify claims made by system manufacturers in terms of the potential benefits of their solutions.
Stimulus	A vision of the future / a potential stakeholders' need.
Situation Today	The decision to acquire/upgrade certain equipment/systems is among many things driven by the potential benefits of the new system (typically articulated in terms of reduction in operational costs). Therefore, there is always a need to verify the realization of those benefits.
Target Artifact	All subsystems
Response	The amount of savings is computable.
Response Measure	The computed amount of savings can be certified.
Role that cares	Facility management, building project manager, device manufacturer, system architect consultant
Architectural Implications	The SW platform needs to provide mechanisms for: • ...tracing execution of the various SW functionality in the system (e.g. a la platform instrumentation) • ...tracing/logging access and usage of the relevant classes of data • ...modeling the "factors" that could influence the "savings" (e.g. in terms of energy management: weather, traffic of people, open doors/windows, etc)
Architectural Trade-off	N/A

4.6.16. Scenario 16

Description	**Add the capability to integrate visitors into the smart building**
Illustrates Quality Attribute	Extensibility, Heterogeneity
Source of Stimulus	The need to make visitors to the smart building an integral part of the environment
Stimulus	A vision of the future / a potential stakeholders' need.
Situation Today	Systems do not provide this type of capabilities. We envision that visitors are able to access many of the building's capabilities via different modalities in a similar ways as residents.
Target Artifact	All sub-systems.
Response	The visitor is registered with the smart building and gains access to the collection of capabilities available to those individuals.
Response Measure	The integration of a visitor takes a few minutes.
Role that cares	End-user, facility management
Architectural Implications	The SW platform needs to provide mechanisms for: • …integrating the new user seamlessly • ...controlling access to data and SW functionality in a way that is consistent with existing policies and flexible enough to allow the visitor to have similar experience as residents • ...enabling delivery of "interaction model" to the visitor (e.g. via an app store model")
Architectural Trade-off	N/A

4.7. Coverage of Quality Attributes and Roles

The architectural scenarios described in the previous section are not an exhaustive list of all possible scenarios and use-cases. However, we believe they provide an adequate representation of the quality attributes required by future SW platforms, an indication of the relevance of those quality attributes to the ecosystem's roles, and also an example of the key functional capabilities that these future SW platforms would have to provide in order for the ecosystem to have the possibility of success. In this section, we summarize the mapping of the quality attributes to roles in the ecosystem via the various architectural scenarios. The following table depicts how the various scenarios (the values in the cells) are representative of a particular quality attribute that is of interest to particular role.

Role	Heterogeneity	Extensibility	Security/Privacy	Scalability	Certifiability	Diagnosability	Usability
end-user	S14,	S10,	S4,S9,S13,S14,				S6,
facility management	S2,S3,S14	S1,S2,S3,S5,S7,	S4,S7,S9,S13,S14,	S8	S7,S12,S15,	S6,S10,S11,	S2,S4,S6
installer		S1,S5		S8		S10,S11,	
certifier					S6,S7	S6,	
commissioning agent					S4,S6,S7,S12		S4,
integrator	S3,	S1,S3,S5	S3,	S8	S12,	S11,	
device manufacturer		S1,S5		S8	S12,S15,	S10,S11,	
device programmer	S2,	S2,		S8			S2,
app developer	S3,	S3,S5,	S3,	S8	S12,	S10,	
building project manager		S1,			S4,S12,S15		S4,
system architect consultan	S1, S9	S1,S2	S9, S13, S14	S8	S15		

It is also worth pointing out that the platform provider role is not included in the table because all scenarios have architectural implications for the platform, consequently, they all impact the platform provider in one of way or another.

5. Key Capabilities for the SW Platform

The scenarios described in section 4.6 provide valuable insight into particular capabilities that the future SW platforms for smart buildings should provide in order to satisfy a particular scenario. Furthermore, a review of the "architectural implications" element of the scenarios shows that many of these capabilities are required across several scenarios, therefore, they represent key capabilities for the SW platform. In this section, we discuss those capabilities in more detail.

5.6. Mechanisms for Integrating New Software Functionality

One of the most basic capabilities that future software platforms need to provide consists of mechanisms for seamlessly integrating new software capabilities. In the smart building context, new devices, new data sources, new ways of combining or utilizing those data, as well as new services will create a constant need for new software functionalities. Therefore, the software platform needs to be able to handle those new additions in a way that is simple and as flexible as possible while minimizing (or ideally eliminating) risks on the operation of the overall system. Broadly speaking, these new software functionalities could be of two forms. First, we have control functionality associated with an existing or new device. This new control software functionality will, at a minimum, deal with the data generated by the device and take actions (e.g. trigger an actuator) based on that data and, potentially, other elements, such as configuration settings, rules, or policies. The second group of new software functionality corresponds to all the non-control-system type of software features that the smart building may need or offer (e.g. data visualization, data analytics, in-door navigation services, information services).

There are several key technical aspects that relate to adding new software functionality in a smart building system that need to be carefully considered when designing the future software platforms:

1. Enabling the new functionality: the platform needs to be able to integrate and make the new functionality available while the overall system is operational. Therefore, the platform needs to support mechanisms similar to plug-ins or loadable kernel modules that can bring software functionality into operation while the system is live. Capabilities, such as sandboxing, may also be required in order to protect the new functionality as well as the rest of the running system.
2. Interfacing with the platform's capabilities: new functionality needs to be able to access and interface with other existing functionality in the platform. Certainly, this could easily be achieved using standard APIs as operating systems and platforms do today. However, mechanisms for discovering the available APIs and how to use them could be very valuable for providing flexibility and extensibility.
3. Using existing data and creating new data: new software functionality (particularly non-control functionality) should be able to access and use the existing data generated by the devices or stored in the platform. In fact, functionalities, such as energy management, could be entirely about using the historical data captured by the platform. Consequently, the platform needs to provide mechanisms for new software functionality to discover existing data sources and understand their properties such that they can be used. Section 6.2 discusses this topic further.

5.7. Mechanisms for Integrating New Data Sources

In a smart building context, taking advantage of the data generated by all the devices as well as those data generated by software functionality (e.g. control SW, pattern identification, summarization, etc) will be a key driver for success. Therefore, the platform needs to provide flexible and extensible mechanisms for collecting,

storing, accessing, and processing the data. There are several key technical aspects related to adding a new data source to a smart building system that need to be carefully considered when designing the future software platforms:

1. Device-generated data collection and storage: the platform needs to be able to adequately support the heterogeneity inherent in the smart building context. Different types of devices will generate quite distinct types of data that need to be handled by the platform (e.g. time series, streaming video, etc). In addition to the heterogeneity, the platform needs to support the scalability of the data. For instance, a smart building owner may decide to add hundreds of cameras or window sensors and integrating those data source without noticeable degradation of system performance should be supported.
2. Data processing: the platform needs to provide mechanisms for the various software functionalities (e.g. control software, apps, etc.) to access and be able to manipulate and process the data. We discuss the access control and privacy aspects in section 6.9. In terms of being able to manipulate and process the data, the platform needs to provide at least two complementary basic functionalities. First, mechanisms to allow the users of the data to discover the type of data that is available (e.g. time series, streaming audio, streaming video, etc.) and the properties of those data (e.g. type of source device, location of source device, time scale, length of the data history, reliability, etc.). Second, mechanisms to efficiently process the data are necessary. Today, there is a wide-range of technologies for large-scale data storage and processing that should be considered and evaluated for applicability in this context (e.g. Hadoop, Cassandra, MongoDB, Google's BigTable, etc). However, it is highly likely that the smart building context would require a combination of SQL database technologies and non-SQL technologies.
3. Software-generated data collection and storage: software functionality, such as control software or applications will likely generate new data. Typically, these new data sources will be the outcome of some sort of processing done on the raw data generated by a device or a collection of devices. For instance, a machine learning algorithm could use motion detection data from a sensor to estimate occupancy in a building. Then, the time series with occupancy data would become a new data source that was extracted from the motion sensor's data. The platform needs to be able to add these types of new data sources and make these new data sources available to other software functionality in similar ways as it would happen for a new device.

5.8. Mechanisms for Geographical Information

In addition to the general capability of data collection, storage and processing, the platform also needs to provide capabilities specific to geographical information. We discuss this data-related capability separate because it has unique implications for the platform. The interviews and use-case workshop suggested, at least, two very valuable applications of geographical information in the context of a smart building. First, providing mechanisms for representing and mapping physical entities and assets (e.g. sensors, computers, desks, chairs, etc.) would be key enablers for developing a wide range of applications that would support roles, such as facility management, installers, and repair personnel and provide significant help in the overall operation of the smart building. The second set of applications is related to the end-users. Providing mechanisms to enable in-door navigation and localization services would be of great value for residents and visitors of a smart building. Then the platform needs to provide the foundation for those geographical information-based services or applications.

5.9. Mechanisms for Delivery of Software Functionality and Data

Roles in smart building ecosystems, such as end-users (residents and visitors) and facility management, will need access to different and new functionality over time (e.g. new apps, new data visualization tools, etc). Then the platform needs to support the delivery of these new functionalities in a simple, flexible, and user-friendly way. The application store model used in mobile platforms as well as in other contexts such as in smart homes (e.g. Control4 app store) represents a potentially fruitful approach. However, the nature of the applications that would exist in this future smart building app store could be quite different from the traditional apps that have been popularized by mobile platforms. For instance, today's mobile apps do not interact among themselves. On the other hand, apps in a smart building context are likely to use many different sources of data, potentially create new sources of data that other apps could use, and potentially interact in real-time with other apps that provide certain types of services (e.g. localization services). Therefore, the capabilities of the infrastructure behind the app store model need to be carefully considered.

5.10. Mechanisms for Supporting Testing, Validation and Certification

The smart building context imposes several unique requirements on the platform in terms of testing, validation, and certification capabilities. The heterogeneity of the devices and roles demands that new software functionalities be tested under a wide range of situations and conditions. The potential for significant detrimental impact on the operation of the smart building and even on the privacy and overall well being of its residents demands that new software functionalities are validated and, to some, extent certified. Then the platform needs to provide mechanisms to test, validate, and certify new software capabilities as well as new devices. There are some techniques that could be of great value in this context and should be carefully considered. More specifically:

1. Simulation of devices and configurations: smart building systems have highly heterogeneous collections of devices. Therefore, testing and validating new software functionality could be highly costly. Simulating the HW devices and allowing developers to create system configurations with those simulated devices can provide an effective and efficient mechanism for reducing the effort and costs associated with testing new software functionality.
2. Tracing and logging of data access and usage: this functionality represents a very important capability that can not only help testing and validation but also can be key mechanisms for certifying functionality that has security and privacy implications as well.
3. Platform instrumentation: is, in essence, a generalization of the previous point. Enabling the collection of different types of data via the platform has the potential to significantly improve verification, validation, and certification efforts. For instance, by collecting data about execution patterns and applying data mining techniques to identify good and bad execution patterns, we can create new forms of validation and certification processes.

5.11. Mechanisms for Event and Notification Management

Event and notification management mechanisms are at the core of many complex systems, particularly, those with large numbers of entities (e.g. devices) that need to be integrated and managed. The smart building context imposes, at least, three unique requirements for the platform. First, the platform should support registering and specifying events and notifications in different ways such that the specific needs of a particular device, subsystem, or context are satisfied. For instance, fire alarm related events and notifications might have more stringent

requirements (e.g. in terms of timing) than those related to temperature control. One possible approach could be to use mechanisms along the lines of service-level agreements with particular constraints depending on the type of event and notification. The second important requirement on the platform relates to managing the potential interactions among events or notifications and other subsystems that may emerge in a complex system, such as smart buildings. For instance, the platform could provide a hierarchical scheme for event and notification management (e.g. as NASA uses for the international space station) to manage the complexity and volume of interactions, events, and notifications. Finally, and related to the last point of amount of notifications, the design of the platform needs to carefully consider the usability aspects of the event and notification management system. A poorly designed system that "swamps" end-users or facility management personnel with notifications and alerts will result in limited adoption of the system and lead to the disregard of the notification and alerts and, consequently, of the potential benefits they may create.

5.12. Mechanisms for Fault Detection and Diagnostics

Diagnosing a failure or malfunction in one or more of the many subsystems that are part of a building is a labor-intensive activity. The high cost of diagnosing and determining the source of the failures stem from several sources. Some of the sources of high cost are related to the lack of interconnectivity or interrelationship of the data from two or more systems that could support the diagnosis process. Other sources of high cost are related to the limited capabilities that today's platforms (or systems) provide in terms of supporting fault detection. Therefore, future platforms should provide these mechanisms. Examples include voting schemes for determining correct values, state or behavior, redundancy mechanisms, and approaches to detect anomalous patterns from the data generated by the various devices, sub-systems or software functions.

5.13. Mechanisms for Articulating Models and Processes

The interviews highlighted a number of activities that involve different roles of the smart building ecosystem and where a platform could provide valuable support in order to make those activities more efficient and less error prone. One architectural implication relates to the ability of the platform to support the definition of models that (a) allow us to represent the state of devices and/or (b) allow us to represent how different pieces of information are interrelated. For instance, being able to define a state model of a sensor would allow us to more effectively and efficiently diagnose a sensor and detect faults in it. A second example relates to point (b). In this case, we envision the possibility of representing, for instance, a model that describes the factors that influence energy savings. Then we could link devices or data sources to the model and the platform would be in a position to support very complex applications for assessing and predicting energy usage and energy efficiency in a smart building.

A second and related architectural implication is the possibility of representing or describing processes such that the platform could provide support for certain activities performed by the ecosystem roles. For instance, "checkout sheets" are representations of workflows that installers or commissioning agents need to follow in order to complete certain tasks. If those workflows can be described within the smart building system and the platform allows linking the various steps of the workflow with certain pieces of information (e.g. data from devices, state of devices, energy usage data, motion/location data, etc.), then these activities could be made more efficient.

5.14. Mechanisms for Access Control and Privacy Management

It is very clear that security and privacy concerns are quite important in a context like smart buildings. However, the specific properties of smart building systems suggest that traditional mechanisms, such as role-based access control to data may not be sufficient to provide the security and privacy protection that would be necessary to make these systems successful (e.g. in terms of adoption, usability and trustworthiness). Therefore, we think the platform should provide additional capabilities that enable more flexible ways to:

1. Access to data and software functionality: certainly a basic access control mechanism (e.g. a traditional role-based) would be necessary. However, such mechanisms would not be sufficient, particularly, in terms of protecting effectively certain pieces of information while simultaneously allowing useful usage of the data. Hence, a more fine-grain mechanism is required for the smart building context. For instance, certain roles under certain conditions (e.g. location, operational mode, etc.) should be able to access certain pieces of data that would normally they would not be able to access. One potential approach to investigate is the notion of capability-based access control. The concept was originally introduced in the 1970s, but it has gained popularity recently.
2. Privacy of data: the same limitations of traditional approaches (e.g. role-based) of access control apply in the context of managing privacy. A case that exemplifies the need for a fine-grained approach is protecting privacy, for instance, of the data generated by a resident depending on whether he/she is in a public space versus a private space. One potential technique to investigate is allowing owners of the data to define access/usage policies for the data and combine such a mechanism with approaches for enabling data operations of the encrypted domain. An approach like this would require rules or policies to be an integral part of a system (see next point).
3. Integration of rules and policies: rules and policies could be numerous, complex, and difficult to track and enforce in complex socio-technical systems, such as smart buildings. If the platform is able to provide mechanisms to describe those rules and policies and integrated them with the security and privacy management systems, we expect that those rules and policies would be more a useful instrument. Usability is a key aspect of this functional capability. Devising user-friendly approaches to the definition and specification of rules and policies would be central for adoption and overall success of such an approach.

5.15. Mechanisms for Interfacing with External Systems

The analyses of the interviews combined with our interpretation of current trends indicate that the platforms of smart buildings will rarely operate in isolation. On the contrary, the software platforms of future smart building will in fact need to interface with a number of other systems. In the context of a campus or a building complex, it is very likely that the software platform in one smart building would need to share and/or request information from another building's platform. Hence, these platforms should provide the capabilities to enable those exchanges of information. Furthermore, the space of potential exchange is so large that it is highly unlikely to anticipate all possible types and forms that those exchanges of information may take. Consequently, the mechanisms provided by the platform should be flexible and extensible. A second type interaction that a smart building platform may have is with IT-related systems (e.g. furniture inventory, access control, key-tracking, telephone/communications, etc.). In this case, the interfacing will be focused on information exchange and coordination for particular activities, such as moving offices, tracking assets or providing services to the residents/visitors of a smart building.

6. SW Architectural Trade-off

The scenarios described in section 5.6 provide also valuable information about certain architectural level decisions that would have significant impact of the future structure and capabilities of those future SW platforms. The impact of those design decisions involves important trade-offs. In this section, we discuss those trade-offs.

6.6. Trade-off #1: Interconnectivity

An important design decision and its associated trade-off relates to where the integration of the devices takes place in the platform. On the one hand, we have the approach taken by platform providers such as Control4 or Android@Home that transfers the responsibility of supporting a particular standardized protocol or collection of APIs to the device manufacturers. On the other hand, we have the approach that utilizes a "gateway" type of device for integrating devices that utilize different protocols. Then, it is the gateway provider's responsibility to provide the integration and the support of all the different protocols.

6.7. Trade-off #2: Extension of Software Functionality

As discussed in sections 6.1 and 6.2, the platform needs to support ways to extend the software functionality and integrate new data sources. In this context, a trade-off exists between a uniform extension mechanism (e.g. a la Eclipse plug-in) and a collection of mechanisms that are tailored to specific types of functionality (e.g. control software, apps, etc) or types of data. A uniform extension mechanism provides a common and standard approach for integrating any type of functionality. This approach has been quite successful in some platform ecosystems, such as Eclipse. However, the simplistic and somewhat minimalistic nature of that approach could be problematic in a heterogeneous context such as a smart building platform. For instance, it may require additional capabilities (e.g. an intermediate layer of plug-ins) to adequately support such heterogeneity. The approaches tailored to specific functionalities or types of data have the advantage of potentially being more efficient. However, they add another layer of heterogeneity that needs to be supported not just within the systems but also within the software development infrastructure. Furthermore, such an approach could lead to limits on the potential contributions from the ecosystem participants stemming from the barriers imposed by the differences in approaches.

6.8. Trade-off #3: Data Management

Design decisions in the context of data management will have a significant impact not just on the overall capabilities of the smart building platform but also on the ability of these platforms to build a valuable and successful foundation for the ecosystem. Despite the broad implications of data management related design decisions, there are two key decisions that have a widespread impact on the structure of the platforms as well as the kind of capabilities that could be offered in terms of data access and processing.

The first design decision involves the storage location of the raw data generated by the devices. On the one hand, we could have an extensible storage environment (e.g. cloud-based storage or single-naming-space network storage) for all the raw data. On the other hand, we could have an approach that pre-processes the raw data, stores the resulting data in the system, and offloads the raw data to a secondary storage. The main trade-off in the context of this design decision is the allocation of cost. With first approach, the cost is allocated on the storage equipment, while in the second approach, the cost is transferred to the data pre-processing.

The subsequent second design decision refers to the traditional tension between generality and specialization (as discussed in 6.1). On the one hand, the platform could provide a general capability for storing, accessing and processing data that allows potential users to abstract the low-level aspects of the data. On the other hand, the

platform could provide a collection of mechanisms that are tailored in terms of each "type" of data – time series, stream data, etc. – or in terms of the type of system or information it refers to – energy consumption, temperature, movement and geographical. This tension between a general approach and a collection of tailored approaches needs to be carefully considered and evaluated.

6.9. Trade-off #4: Access Control

A key trade-off in terms of access control is the generalizability of the mechanism. On the one hand, we can have a uniform access control mechanism that manages access to all software functionalities and data. On the other hand, we have a collection of mechanisms that are can be tailored by "type" of data or type of software functionality or by type of the data owner". In this trade-off, we have the same tension between generality and specialization discussed in 6.1 and 6.3 that needs to be carefully considered and evaluated.

6.10. Trade-off #5: Privacy Management

A key trade-off in terms of privacy management hinges on the decision of who retains control of the data and how that control is enforced or articulated. On the one hand, the control of the data could remain with the owner (e.g. end-user or entity responsible for the device). In this case, the data collected and stored by the system could be highly sparse and, consequently, its value may be limited particularly in terms of analyses to impact control activities. However, we are able to provide a very strong support for privacy protection. On the other hand, the data is always provided to the platform, but the owner is able to specify who can access it and how. In this case, all the data would be available, so we don't have the limitation of the other case. However, the platform needs to provide mechanisms to certify that the policies established by the user are enforced. This means that the platform needs to provide mechanisms for detailed data access and usage tracing and logging.

6.11. Trade-off #6: Event and Notification Management

Event and notification management approaches have received significant attention in the academic and practitioners' literatures over the past few decades. Despite that large body of work that have proposed numerous architectural and design patterns, we think that for the smart building context there is one particular design decision that has very broad implications for the structure of the overall system. On the one hand, we could have the software platform provide a uniform mechanism for defining events and triggering notifications. In essence, we could envision a standardized mechanism throughout the entire smart building system along the lines of what triggers represent in the database domain. On the other hand, we could take the opposite approach and provide a collection of specialized mechanisms that are tailored to the specific nature of the subsystems (e.g. HVAC, fire, lighting, etc) or to the nature of the users or usages. Again and as discussed in 6.1, 6.3 and 6.4, we face a tension between generality and specialization that needs to be carefully considered and evaluated.

6.12. Trade-off #7: Model for System Operational States

In terms of system operation states, we have a key design decision that would push the design on the platform in two very distinct paths. On the one hand, we can define operational states as discrete states where functionalities are either operational or not. In this case, we could envision three operational states: a *live* mode where all functions are operational, a *shutdown* mode where all functions are not operational, and a *failure* mode where a subset of functions is operational and the other set is not. On the other hand, we could have operational states where functions are not in "on" or "off" state, but other states where certain considerations or assumptions are

considered. Examples of these additional states include “fail-safe” or “test-safe” operational states. This second approach provides greater flexibility and potentially increased availability of a system. However, it introduces a significant toll on the complexity of the system because supporting these additional states could be quite involved. The benefits and limitations of this second approach are reversed in the case of the first approach.

7. Concluding Remarks

The work reported in this document focused on the analyses of the type of contributors that may exist in a smart home ecosystem, the quality attributes that those roles are concerned with and the key functional attributes that the software platforms of future smart buildings should provide in order to make smart building ecosystems successful. The analyses of interviews, use-cases and architectural scenarios highlight the need for future smart building software platforms to support the following collection of key functional capabilities:

1. Mechanisms for Integrating New Software Functionality
2. Mechanisms for Integrating New Data Sources
3. Mechanisms for Geographical Information
4. Mechanisms for Delivery of Software Functionality and Data
5. Mechanisms for Supporting Testing, Validation and Certification
6. Mechanisms for Event and Notification Management
7. Mechanisms for Fault Detection and Diagnostics
8. Mechanisms for Articulating Models and Processes
9. Mechanisms for Access Control and Privacy Management
10. Mechanisms for Interfacing with External Systems

In addition to those key functional capabilities, our analyses identify several architectural trade-offs that need to be evaluated and carefully considered:

1. Interconnectivity
2. Extension of Software Functionality
3. Data Management
4. Access Control
5. Privacy Management
6. Event and Notification Management
7. Model for System Operational States

In the second year of this academic collaboration, we will concentrate on two important and complementary activities. First, we will further articulate the functional capabilities required in the platform as well as the specifics of the architectural trade-offs. Second and building on the work reported in this document and on further development of the architectural trade-off and implication, we will build a platform prototype as a test bed for the various design principles identified in this work.

8. Acknowledgements

The authors would like to thank Dr. Elizabeth Latronico (Robert Bosch Corporate Research) for organizing and leading the use-case workshop. In addition, the authors would like to thank Dr. Jorge Guajardo Merchan (Robert Bosch Corporate Research) for his valuable input on the security and privacy related topics discussed in sections 5 and 6. Finally, we gratefully acknowledge the financial support from Robert Bosch Corporate Research.

9. References

Brandenburger, A. and Nalebuff, B. (1998). *Co-opetition*. Crown Business.

Gawer, A. and Cusumano, M. A. (2002). *Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation*: Harvard Business Press.

Herbsleb, J.D. et al. (2010). The VistA Ecosystem: Current Status and Future Directions. Technical Report, Institute for Software Research, Carnegie Mellon University.

Iansiti, M. and Levien, R. (2004). *The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability*: Harvard Business Press.

MacCormack A. et al (2001). Developing Products on "Internet Time": The Anatomy of a Flexible Development Process. *Management Science,* vol. 47, pp. 133-150.

Smite, D. et al. (2010). *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*: Springer-Verlag New York Inc.

Zhu, F. et al. (2007). *Dynamics of platform competition: Exploring the role of installed base, platform quality and consumer expectations*: Division of Research, Harvard Business School.