Well-being as a Composite Capability in the Smart Building Domain: A Formal and Technical Study

Khalid Halba[1 2 3] , Asmaa Hailane[3 4] , Ahmed Lbath[2] , Edward Griffor[3] , Anton Dahbura[1]
[1] Johns Hopkins University, Baltimore MD, USA, (khalba1, AntonDahbura)@jhu.edu
[2] Grenoble Alpes University, Grenoble, France, (khalid.halba, ahmed.lbath) @univ-grenoble-alpes.fr
[3] NIST, Gaithersburg MD, USA (khalid.halba, edward.griffor, asmaa.hailane) @nist.gov
[4] University of Ottawa, Ottawa, Canada, ahail055@uottawa.ca

***Abstract* — Smart building value-added capabilities are gaining significant attention from various stakeholders, including the general public, researchers, and industry. One such capability is well-being, a composition of multiple atomic capabilities that characterize a smart building. Atomic functions that compose a well-being capability include temperature, noise level, pollution level, and humidity, to name a few. Multiple efforts have addressed this specific capability and its composition requirements and techniques from standardization, technical, and quality of service aspects. One such effort is the IoT and CPS Composition Framework (ICCF), a novel framework for rapid modeling, specifying, verifying, and prototyping IoT and CPS capabilities. ICCF relies on the NIST CPS Framework guidelines to address different stakeholders' concerns; it also leverages composition semantics inspired by the mPlane platform to describe entities and interactions intuitively. In addition, it uses the Temporal Logic of Actions + (TLA+) formal verification techniques to verify the correctness of core functions. This work leverages the ICCF framework to provide the following contributions: i) description of a stakeholder-defined well-being composition capability based on the ICCF framework foundations, ii) an in-depth characterization of the well-being capability, iii) considerations regarding the formal aspects of the well-being capability, including verifying its correctness, deadlock, and state-space, iv) implementation of the composite capability using a lightweight microservices environment, v) discussion of results based on the different domains of interest including residential buildings and factories. Finally, a summary of this effort is provided, and challenges to capabilities composition as well as future plans for improvement are highlighted.**

Keywords* — *Internet of Things(IoT), Cyber-Physical Systems(CPS), Composition, Smart Buildings, Well-being, Formal Verification, Micro-services

I. Introduction

Innovating value-added capabilities in Internet of Things (IoT) or Cyber-Physical Systems (CPS) - concepts that are converging over time [1] - through composition is gaining more and more interest as a wide range of sensors and appliances are generating data that can be exploited to improve various features within the different types of environments including smart buildings. Smart buildings are emerging as complex cyber-physical systems with humans in the loop [2], that should provide a safe and comfortable space for inhabitants. Well-being is a function that represents the quality of living perceived by different entities (residents, maintenance, owners, etc.) in a particular building with specific properties; this means that well-being requirements in a residential building can differ from those of a factory or a hospital. In other words, as buildings vary in terms of shape, characteristics, and purposes, well-being definition varies accordingly to accommodate those particularities. For example, well-being in a residential building has the primary purpose of addressing the occupants' comfort. Consequently, it requires reasonable levels of temperature, humidity, air quality, noise level [3]–[5], and WiFi signal strength as wireless internet access in smart buildings is common practice as opposed to wired communications and represents a crucial feature that appliances for well-being or entertainment rely on [6].

On the other hand, for hospitals, well-being is based on ensuring sanitary conditions, which may require more adjustable levels of temperature, humidity, and air quality compared to those of residential buildings [7]. The same can be said about clean rooms in which semiconductors are built where certain products require temperature, humidity, and air quality (dust, particulate matter) levels that differ from those of residential buildings or hospitals [8].

Factors that influence well-being can be classified into two categories: i) static factors that a stakeholder has little to no ability to change or improve; these include the shape of the building, its location, to mention a few, and ii) dynamic factors, which the stakeholder can alter and improve using multiple techniques or systems and those factors include temperature, noise, humidity, and air quality. This work focuses on dynamic factors that impact well-being in a given type of building.

Modeling, prototyping, and rapidly implementing com-

posite capabilities in IoT and CPS are essential for innovation in smart buildings. Therefore, following streamlined and easy-to-follow guidelines is crucial to ensure adherence to standards [9] and best practices and achieve an acceptable level of correctness and reliability.

In this work, the IoT and CPS Composition Framework (ICCF) [10] -a framework for composing novel IoT and CPS capabilities- is leveraged for guiding, modeling, prototyping, composing, and verifying an IoT composite capability called well-being. This capability addresses concerns in different types of buildings. This effort illustrates an end-to-end composition effort; to the best of our knowledge, the well-being capability has never been formally specified as a full-fledged capability; this effort provides a prototype for this novel feature. The ICCF framework guidelines followed in this work derive from the NIST CPS Framework -a framework that provides a comprehensive analysis of a CPS and captures its generic functionalities, activities, and artifacts needed to support its conceptualization, realization, and assurance- [11]–[13], make use of intuitive and comprehensive mPlane semantics [14] to describe entities and interactions, and leverages robust, straightforward, and easy to use Temporal Logical of Actions (TLA) formal verification techniques and tools [15], [16]. It was demonstrated in [10] that these foundations, guidelines, and tools, form a powerful framework that would help researchers, developers, and engineers better frame and organize their composition and innovation efforts and adhere to procedures and semantics to improve capabilities quality, correctness, and reliability. The contributions of this paper are organized as follows: Section II provides a comprehensive related work about well-being in IoT. Section III explains the introduced definition of well-being and its relation to different stakeholders' concerns and describes entities and interactions contributing to its composition and computation in an ICCF implementation called IoTCaP. In Section IV, mPlane semantics for the well-being function are formally specified using the PlusCal language, model checking and deadlock analysis of the formal specification are done using TLA+. In section V, an implementation of IoTCaP is provided as well as a discussion of the results obtained. Finally, a summary of this effort is provided, in addition to discussing encountered challenges and future work.

II. Related Work

This section provides a comprehensive review of capabilities composition in the smart building domain. In addition, it highlights efforts addressing well-being or comfort as a capability in the IoT or CPS space.

A. Smart Building Applications And Composition

Different research efforts addressed smart buildings applications from a service composition perspective. In [17], *Brick*, a uniform metadata schema for representing smart buildings components -including sensors and subsystems, and describing the different interactions that occur between these components- is proposed. The goal of such schema is to provide APIs that enable the creation of portable energy efficiency applications. In [2], a platform-based methodology for smart building design (PBD) was proposed, which promotes the reuse of hardware and software on shared infrastructures, enables rapid prototyping of applications, and involves extensive exploration of the design space to optimize design performance. In [18], IoT was leveraged to build an Energy Management System (EMS) that takes into account the behavior of individual customers who occupy smart buildings. This idea is extended in this work to allow different stakeholders to customize well-being to address their needs. In [19], full-IP IoT with real-time Web protocols is discussed as a major enabler for efficient and meaningful aggregation of services known as composition and how that would impact domains such as smart buildings. In [20], energy cost, thermal comfort, and social IoT (SIoT) concepts are composed to provide a Smart Heating, Ventilation, and Air Conditioning (HVAC) system for smart buildings. In [21], Fault Maintenance Trees (FMTs) and probabilistic model checking are used to evaluate various dependability metrics and maintenance strategies of Heating, Ventilation, and Air-Conditioning of a smart building. In [22], an adaptive service composition framework that supports dynamic reasoning on numerous smart city IoT services was proposed, and a contExt Aware web Service dEscription Language (wEASEL) abstraction was leveraged to represent services, compositions, and interactions. For evaluating the composition framework, an OWLS-TC4 testbed was proposed to evaluate the accuracy and novelty of simple and composite services. In [23], a model was proposed for simulating the core engineering subsystems of a smart building. The model is based on Matlab Simulink, the Simscape physical modeling library, and the Stateflow library. The goal of the model is to simulate coordination between subsystems and assess power consumption for optimization purposes.

B. Previous Efforts On Studying Well-Being

In this section, previous efforts that tackled comfort or well-being as an IoT capability are discussed, in particular, the IoT and CPS Composition Framework (ICCF) [10]. Research on well-being usually addresses a single atomic capability that contributes to well-being instead of studying well-being as a composite function which this works addresses. In [24] the role of smart urban technologies was demonstrated especially in ensuring sustainable cities and well-being for the citizens; however, the study focused on sustainability and energy efficiency aspects and didn't address well-being concerns. In [25], a summary of IoT technologies enabling smart buildings was provided, and a referenced study [26] found that peo-

ple spend 80 percent of their lifetime inside buildings; the study showed the extent to which comfort is a key component that IoT researchers and engineers must address to improve the well-being of smart building residents. In [27], highlighting the differences between smart buildings and autonomous buildings was performed to clear the confusion associated with these two concepts; they also introduced a measurement to quantitatively assess the building "intelligence". One key criterion contributing to the building intelligence metric is "inhabitants comfort". This component is defined as the ability of "Smart" homes to learn from inhabitants' behavior and maximize their comfort. In [28], the book on Green and Smart buildings discusses the shift buildings are witnessing as they become more and more people-centric rather than engineering, construction, or technology-centric. This is exemplified in how building owners, developers, and facility managers focus on increasing the occupant well-being and comfort by building smarter spaces that engage occupants, understand how they use buildings in new ways, and get them involved in implementing sustainable practices. In [29], a focused study on addressing thermal comfort was done within an occupied building and how that requires energy and, thus, an optimized solution balancing energy use with indoor environmental quality (adequate thermal comfort, lighting, etc.). In [30], a quantitative composite air quality metric was introduced as a contributor to well-being. In [31], balancing comfort requirement with environmental and energy constraints was discussed by leveraging a Context-Aware Framework for Collaborative Learning Applications (CAFCLA) to combine various technologies that simplify the creation of context-awareness and social computing systems that influence user behavior to favor efficient energy resources without compromising comfort in the workplace. In [32], comfort and well-being were addressed from a privacy perspective. By setting and interacting with smart devices, inhabitants risk giving away details about their preferences that compromise elements of their privacy. A framework that forces IoT Assistants -when capturing and managing the privacy preferences of their users- was proposed to communicate privacy-sensitive information to privacy-aware systems. In [33] a discussion around extracting information that enhances inhabitants' comfort from smart buildings was done. Smart buildings generate huge amounts of data, and the integration of Big Data Analytics (IBDA) and IoT to address the large volume and velocity of real-time data in the smart building domain is proposed to enhance well-being. In [34], the health and well-being of smart cities residents is discussed as a social aim of IoT rather than the usual sustainability and ecology aim. A case study of smart health and well-being in Kashiwanoha Smart City in Japan was discussed, and the impact on resident lifestyles was assessed. Findings suggest that smart cities have great potential to be designed and executed to tackle social problems and realize more sustainable, equitable, and livable cities. In [35], comfort, among other qualities, is discussed as a crucial property in educational buildings. Comfort contains thermal, acoustic, visual, and air quality components. The paper presents a case study in Nuevo León, Mexico, where a comparative study was conducted to assess the teaching-learning process in different environments with different health, safety, and comfort criteria. In [36], a new metric was proposed: the Smart Readiness Indicator (SRI). SRI shows whether or not a building respects different criteria defined by the European Union Standards, the criteria contributing to the SRI include energy, flexibility for the grid, self-generation, comfort, convenience, well-being and health, maintenance, and fault prediction, information to occupants. The SRI methodology differentiates between three weights for all functionality levels and impact criteria: equal weights, residential, and non-residential buildings. In [37], smart buildings are discussed as well as several technologies that support their functionalities, including the automated building management system (BMS), which aims to achieve the well-being of occupants, promoting a comfortable environment while ensuring efficient use of building resources. Context-based reasoning as a modeling paradigm for smart buildings is proposed as a supportive mechanism to realize smart building applications.

C. Takeaway

As a summary of the related work above, well-being or comfort was addressed as a qualitative feature, with some efforts addressing single aspects of well-being such as air quality. However, and to the best of our knowledge, none of the research efforts investigated tackled well-being as a dynamic quantitative composite measurement that can be formally specified and verified, which this effort aims to achieve.

III. An ICCF Definition Of Well-Being As A Composite Capability

This section proposes an ICCF-based definition for well-being as a quantitative, dynamic, and composite capability. Next, a discussion sheds light on this capability from different stakeholders' perspectives and how these perspectives can be quantified and incorporated in the well-being metric computation. After that, semantics representing the requirements mentioned above using mPlane notations and algebra are provided. Finally, an illustration is done for the required operations using a sequence diagram and pseudo-code in the context of a platform called the IoT Capabilities Platform (IoTCaP).

A. A Proposed Definition For Well-Being Based On Stakeholders Concerns

A set of metrics that contribute to well-being in smart buildings were identified; some of these metrics are found

in the WELL Building Standard version 2™ [38]. These metrics include temperature, humidity, air quality, noise level, and WiFi signal strength. These capabilities change over time, and sometimes they fall under comfort levels that compromise well-being as a whole. In this work, *weights* and *scores* characterize, quantify, and contribute to the computation of the $well-being$ metric as seen in equations (1) below:

$$\begin{aligned} &1 \leq st, sh, sa, sn, ss \leq 5 \\ &Wt{+}Wh{+}Wa{+}Wn{+}Ws{=}20 \\ &0 \leq Wt, Wh, Wa, Wn, Ws \leq 20 \\ &(st, sh, sa, sn, ss, Wt, Wh, Wa, Wn, Ws) \in \mathbb{N} \\ &WB{=}Wt * st{+}Wh * sh{+}Wa * sa{+}Wn * sn{+}Ws * ss \end{aligned} \tag{1}$$

where WB, a value between 0 and 100, represents well-being (higher is better). st, sh, sa, sn, and ss represent scores assigned to the values of temperature, humidity, air quality, noise, and WiFi signal strength, respectively. These scores must be between 1 and 5, with a higher score representing a range of more comfortable values for the stakeholder in a given domain.

Finally, Wt, Wh, Wa, Wn, and Ws represent individual weights for temperature, humidity, air quality, noise, and WiFi signal strength, respectively. Individual weights represent the importance a stakeholder assigns to a given metric and can be between 0 and 20, but their sum is always equal to 20.

For example, in a residential building, the weights for the different metrics would all have the same importance, which means the weight is equal to 4 for each metric.

If a single metric is crucial and the others are negligible, the value 20 is assigned as the weight for that critical metric, while the other metrics will have a weight equal to zero.

B. IoTCaP: An ICCF-based Platform For Implementing The Composite Capability Entities And Interactions.

This subsection describes entities and interactions within IoTCaP using the mPlane semantics and algebra, describes those interactions using a sequence diagram, and highlights the overall behavior using pseudo code.

1) Algebraic Description Of IoTCaP Entities: *IoTCaP* is the platform built to implement the ICCF-based composite capability. *IoTCaP* implements the following entities: *Actor* : logs into the IoTCaP front-end interface and selects domains or custom weights, *Auth* : authentication mechanism, *IoTCaP* : dashboard for visualizing the composite capability as well as inserting weights either manually or by selecting a specific domain, $well-being$: back-end process that implements the mPlane protocol by posting, *specification* to sensors. The $well-being$ back-end also computes the composition based on the received weights and the composition formula, *Sensors* : receive specifications from the $well-being$ back-end and return *Results* based on the *Capability* schema. Consider $\mathscr{R}$, the space of capabilities that can be composed and decomposed using the ICCF framework. Consider *Ct*, *Ch*, *Ca*,*Cn*, *Cs*, the Capabilities descriptors generated by the temperature, humidity, air quality, noise, and signal strength sensors respectively.
Consider *sCt*, *sCh*, *sCa*,*sCn*, *sCs*, the specification posted to the sensors of temperature, humidity, air quality, noise, and signal strength respectively.
Consider *rCt*, *rCh*, *rCa*,*rCn*, *rCs*, the result obtained from the sensors of temperature, humidity, air quality, noise, and signal strength respectively.

2) Algebraic Description Of IoTCaP Interactions Using mPlane Semantics: The different interactions that occur within IoTCaP can be described as follows:

- Authentication interaction: The *Auth* entity authenticates an *Actor* to the IoTCaP front-end; the *Actor* can be a user or a process. The Authentication function is defined as Authenticate(*Actor*, *Auth*); it compares *Actor* credentials against *Auth* database and allows access to the *IoTCaP* front-end if there is a match.
- Discovery interaction: The $well-being$ back-end discovers entities that contribute to computing $well-being$. The discoverability function is defined as Disc($well-being$, *C*), it takes $well-being$, a composition manager, which also represents the $well-being$ back-end and *C*, a capability as input and returns a binary that shows whether or not that capability is discovered.
- SendSpec(Src,Dst,Specification) interaction: It is a request *sC* used in two instances: i) *sCwb* : a well-being specification (in the mPlane semantics, a specification is a request sent from a particular entity or a service to get data values or results from a sensor or another service) sent from the *IoTCaP* front-end to the $Well-being$ back-end, or ii) *sCt*, *sCh*, *sCa*,*sCn*, *sCs*, sent from the $Well-being$ backed to the different *Sensors* providing atomic capabilities.
- sendResult(Src,Dst,Result) interaction: According to the mPlane semantics, processes or sensors provide a *Result* for the *Specification* they receive based on their *Capability*. Two instances of this operation are witnessed in our implementation: i) results of sensor values returned upon processing specifications sent by the $well-being$ back-end. ii) results of the computed well-being value returned to the IoTCaP front-end.
- Composition function: Consider an operator ψ, which represents a k-ary composition operator. To illustrate composition in this paper, an assumption of k=5, representing the five capabilities contributing to the composite capability of well-being, is made. The composition is an operator on values obtained after sending a specification to all atomic capabilities and receiving results.
- Capability weight and composition computation:

Consider Wt , Wh, Wa, Wn, Ws, the weights of rCt, rCh, rCa,rCn, rCs respectively, the well-being composite result $rCwb$ can be expressed as seen in equation (2):

$$\begin{aligned}&\psi : \mathbb{N}^5 \rightarrow \mathbb{N}\\&(rCt, rCh, rCa, rCn, rCs) \rightarrow rCwb\\&\rightarrow Wt * rCt + Wh * rCh + Wa * rCa + Wn * rCn + Ws * rCs\end{aligned} \tag{2}$$

• Specification decomposition function: since the well-being Cwb represents a composite capability $CapabilityDescriptor$, the $Specification\ sCwb$ will be decomposed to its atomic $Specifications$ (sCt, sCh, sCa,sCn, sCs) by applying the decomposition operator ψ^{-1} as seen in expression (3):

$$\begin{aligned}&\psi^{-1} : \mathscr{R} \rightarrow \mathscr{R}^5\\&(sCwb) \rightarrow (sCt, sCh, sCa, sCn, sCs)\\&\rightarrow \psi^{-1}(sCwb)\end{aligned} \tag{3}$$

3) Illustrating Entities And Interactions Using A Sequence Diagram: The sequence diagram in Fig. 1 summarizes the interactions between IoTCaP entities. The "Actor" authenticates to its profile in IoTCaP then selects weights for different metrics based on their importance. Once weights are submitted, the "Well being" composition engine computes a well-being value and returns it to the Actor as well as values for atomic metrics. In the case where the sum of weights exceeds 20, an error is displayed to the Actor.

4) Illustrating mPlane-described IoTCaP Entities And Interactions Using Pseudo-code:

The pseudo-code in Algorithm 1 summarizes the well-being composition operations. In particular, line 15 refers to the computation of the composite capability of well-being.

IV. Formal Specification And Verification

Formal verification is necessary because it is a reliable way to verify the mPlane-modeled functions. Furthermore, fixing and optimizing the well-being model becomes possible by studying deadlocks and in-variants. This section converts mPlane semantics defined previously to PLUSCAL language, which is translated to TLA specification. Then the model checker yields the state space, and by studying invariants, assessing the correctness of the model, or proposing corrections in the case of errors or deadlocks is possible. The choice of TLA+ was based on the fact that it is a trusted tool for verifying microservices, including in commercial solutions such as AWS [39] where it was leveraged to verify the correctness of properties such as fault tolerance in storage services.

Algorithm 1 IoTCaP platform mPlane interactions.

1: **if** $Ct, Ch, Cs, Ca, Cn \in \mathscr{R}$ and
2: Disc($well - being$,(Ct, Ch, Cs, Ca, Cn))$\leftarrow$ true and Authenticate($Actor$, $Auth$)$\leftarrow$ true **then**
3: sendSpec($IoTCaP$, $well-being$, $sCwb$)
4: ψ^{-1}(sCwb)$\rightarrow$(sCt, sCh, sCa,sCn, sCs)
5: sendSpec($well-being$, Ct, sCt) and
6: sendSpec($well-being$, Ch, sCh) and
7: sendSpec($well-being$, Ca, sCa) and
8: sendSpec($well-being$, Cn, sCn) and
9: sendSpec($well-being$, Cs, sCs)
10: sendResult(Ct, $well-being$, rCt) and
11: sendResult(Ch, $well-being$, rCh) and
12: sendResult(Ca, $well-being$, rCa) and
13: sendResult(Cn, $well-being$, rCn) and
14: sendResult(Cs, $well-being$, rCs)
15: ψ(rCt,rCh,rCa,rCn,rCs)$\rightarrow$($rCwb$)
16: sendResult($well-being$, $IoTCaP$, $rCwb$)

A. Formal Verification Environment

TCL version 1.5.7, TLA+'s model checker, is executed in a four-core CPU-equipped Linux Ubuntu 14.04 VM with 8GB RAM.

B. Adapting The State Space For PLUSCAL's Requirements

Converting mPlane algebra to PLUSCAL language requires some modifications, including adapting the state space of sensor values to include positive values only. Fig. 2 describes a pipeline of operations that yields values compatible with PLUSCAL. The outcome is shown in Table I where sensor values -provided by the temperature and humidity sensor (DHT22), noise sensor (KY-038), air quality sensor (SDS011), and WiFi signal strength sensor (ESP8266 SOC)- were translated into positive values and distributed among ranges representing different well-being areas. These areas are assigned scores that reflect how they contribute to the well-being metric.

For example, the temperature sensor provides temperature values tv between -40 ºC and 125 ºC. To make these values positive, the value 40 ºC is added to both ends of this range to make all values positive (because the minimum temperature value is -40 ºC). The best values of temperature that contribute to well-being are between 20 ºC and 22 ºC. This range RT of values is assigned a score RTs of 5, the equivalent range in PLUSCAL is the positive range RpT with values between 60 ºC and 62 ºC. A score $RpTs$ of 5 is assigned to this positive range as well, and this value is used to compute the overall well-being after subjecting all the sensor values to the same pipeline processes described in Fig. 2. In the case of humidity, sensor values hv won't require adaptation as the values are already positive, as it can be seen in Table I.

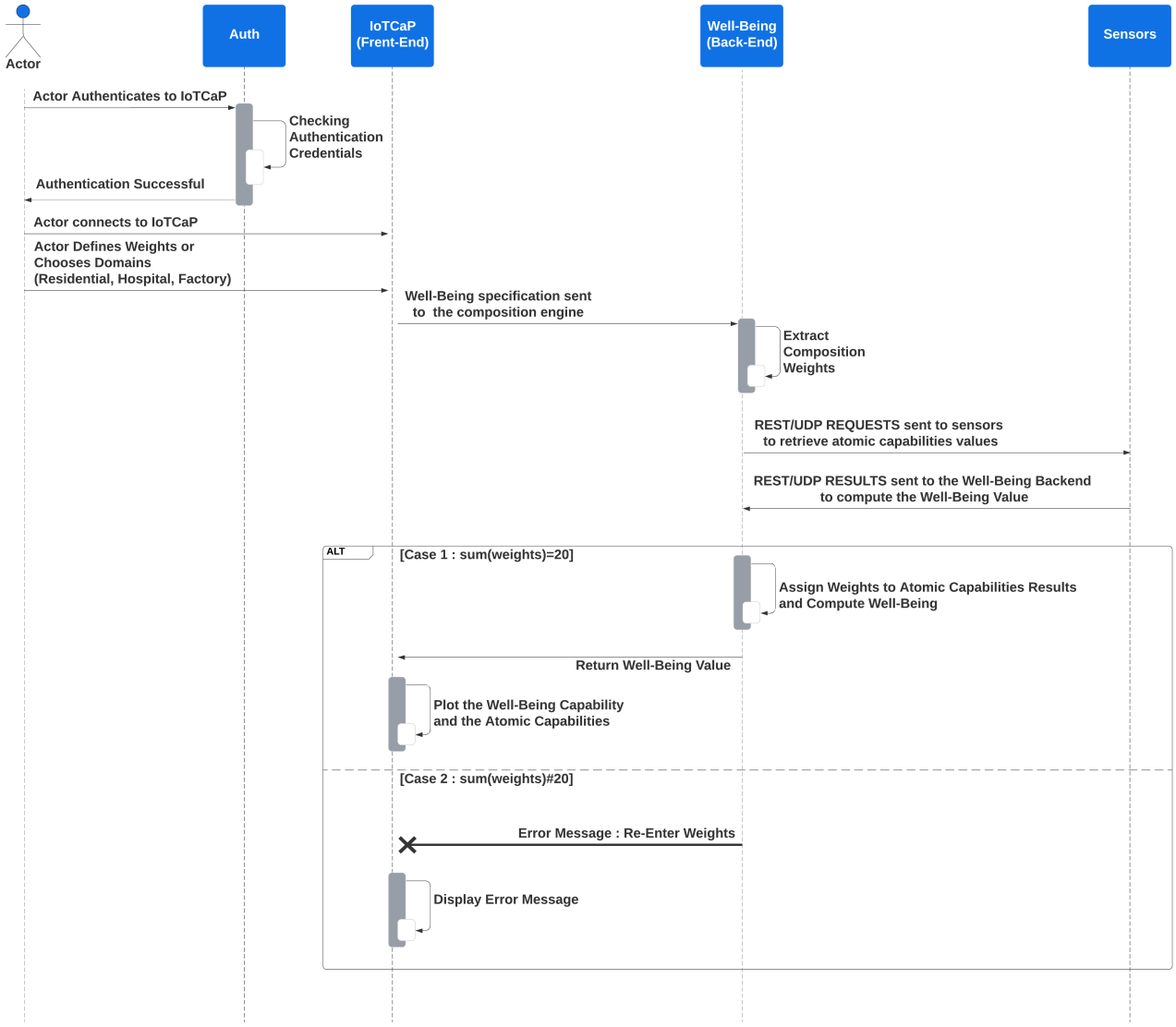

Figure 1. Sequence diagram for the well-being capability expressed using mPlane interactions.

C. Interpreting Core Composition Functions Into PLUSCAL And Translation Into TLA Specification

Two core functions for composing the well-being capability are interesting from a formal verification perspective:

a) The score assignment function: The role of this function is to assign scores from 1 to 5 for sensor values based on how much they contribute to well-being.

b) the Composition computation function: The role of this function is to compute well-being based on both the assigned scores of sensor ranges of values and metric weights.

Fig. 3 shows both of these functions as described in the PLUSCAL language.

After translating the PLUSCAL description, the TLA specification is generated as seen in Fig. 4.

D. Model Checking And State-Space Analysis

1) Running Symbolic Execution And Discussing Results Of The State Space: The symbolic execution of the model results runs combinations of all atomic capabilities scores and weights to assign score values to humidity ranges and determine the well-being state space. In Fig. 5, it took 4 minutes and 31 seconds to perform symbolic execution; TLC can improve its execution time when it leverages multiple EC2 or Azure cloud instances. The number of states generated across all combinations is 60750000, with 35437500 distinct states, which means more optimization can be done on the model. The queue experienced congestion at 31 seconds after execution, but it was emptied over. This simple but efficient method of calculating well-being using TLC's model checker was verified, and the results show the correctness of the core

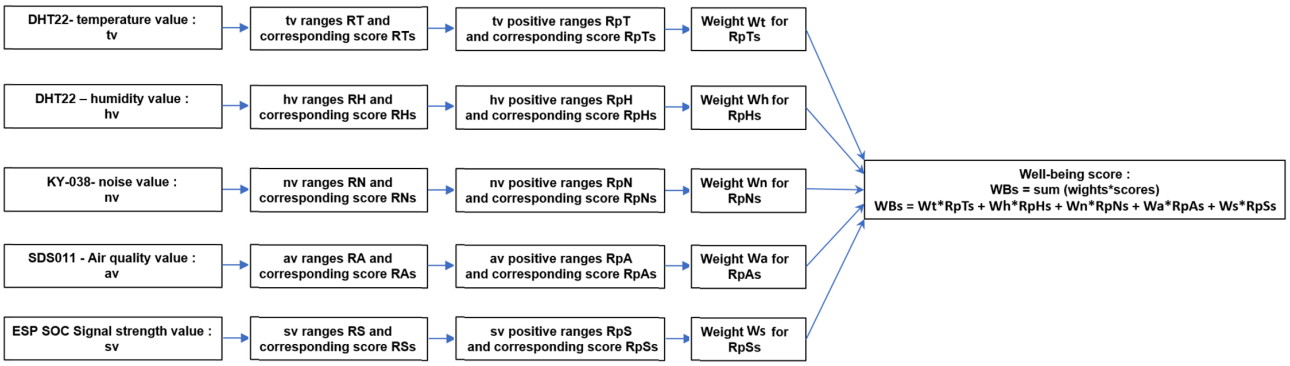

Figure 2. Modeling the state space for the well-being composition values based on corresponding atomic capabilities values, ranges, projections, and scores.

Table I. STATE SPACE OF VALUES, RANGES, AND SCORES RELATED TO WELL-BEING ATOMIC CAPABILITIES IN A RESIDENTIAL BUILDING.

Atomic Capability	Sensors	Units	Low value	High Value	Real sensor values ranges (RT, RH, RN, RA, RS) and score assignment per range (RTs, RHs, RNs, RAs, RSs)					TLA+ compatible sensor values ranges (RpT, RpH, RpN, RpA, RpS) and score assignment per range (RpTs, RpHs, RpNs, RpAs, RpSs				
					1	2	3	4	5	1	2	3	4	5
Temper-ature	DHT22	$^{\circ}C$	tv		RT					RpT				
			-40	125	[-40,-1] or [41,125]	[0,9] or [31,40]	[10,16] or [26,30]	[17,19] or [23,25]	[20,22]	[0,41] or [81,165]	[40,49] or [71,80]	[50,56] or [66,70]	[57,59] or [63,65]	[60,62]
Humidity	DHT22	%	tv		RH					RpH				
			0	100	[0,10] or [80,100]	[11,20] or [61,79]	[21,25] or [55, 60]	[26,29] or [51,54]	[30,50]	[0,10] or [80,100]	[11,20] or [61,79]	[21,25] or [55, 60]	[26,29] or [51,54]	[30,50]
Noise	KY-038	*dB*	tv		RN					RpN				
			23	129	[81,129]	[51,80]	[41,50]	[31,40]	[25,30]	[81,129]	[51,80]	[41,50]	[31,40]	[25,30]
Air Quality	SDS011 2.5UG	$\mu g/m^3$	tv		RA					RpA				
			0	999	[101,999]	[36,100]	[21,35]	[13,20]	[0,12]	[101,999]	[36,100]	[21,35]	[13,20]	[0,12]
WiFi Signal Strength	ESP8266 WiFi SOC	*dBm*	tv		RS					RpS				
			-80	-5	[-80,-61]	[-60,-51]	[-50,-41]	[-40,-31]	[-30,-5]	[0,19]	[20,29]	[30,39]	[40,49]	[50,75]

functions of the composite capability.

2) Preventing State Space Explosion: Score assignment was executed for humidity values alone to reduce the state space and prevent state space explosion. Once humidity is verified, swapping ranges to compute score assignment for the other sensor values is straightforward. This technique saves hours of symbolic execution runtime. TLA+ can also save time by running the symbolic execution on cloud instances such as EC2 or Azure instances.

E. Deadlock Case And Corrective Measures

As software developers, applying formal methods and model checking enables thinking above the code level, validating the understanding of the composite capabilities, and finding critical bugs that are difficult to spot. For example, the invariant study aims to test the capability in case well-being weights aren't equal to 4. i.e., Wt and Ws randomly vary between 0 and 8. However, if well-being is set as an invariant where its value is strictly inferior to 100, and with random weights or without proper controls, the symbolic execution of the well-being model might yield values superior to 100, which is an incorrect outcome that causes TLC to throw deadlock errors. Adding a control instruction that forces TLC to check whether the sum of weights equals 20 would prevent this deadlock case.

```
---------------------------- MODULE wellbeing ----------------------
EXTENDS Integers, TLC
(*--algorithm wellbeing
variables hv \in 1..100, RpHs \in {1,2,3,4,5},
          RpTs \in {1,2,3,4,5}, RpNs \in {1,2,3,4,5},
          RpAs \in {1,2,3,4,5}, RpSs \in {1,2,3,4,5},
          WT \in 0..8, WH=4, WS \in 0..8, WA=4, WN=4
begin
  HumidtyRangeVerification:
    if ((hv<51)/\(hv>29)) then
        RpHs := 5;
        else if (((hv<30)/\(hv>25))\/((hv<55)/\(hv>50))) then
        RpHs := 4;
        else if (((hv<26)/\(hv>20))\/((hv<61)/\(hv>54))) then
        RpHs := 3;
        else if (((hv<21)/\(hv>10))\/((hv<80)/\(hv>60))) then
        RpHs := 2;
        else if (((hv<11)/\(hv>0))\/((hv<101)/\(hv>79))) then
        RpHs := 1;
        end if;
        end if;
        end if;
        end if;
        end if;
  CompositionScoreVerification:
      if (WH+WT+WS+WA+WN<21)then
          print ((WH*RpHs)+(WT*RpTs)+(WA*RpAs)+(WN*RpNs)+(WS*RpSs));
            end if;
end algorithm;*)
```

Figure 3. Screenshot from TLA+ indicating the functions to verify described using the PLUSCAL language.

V. Implementation, Results, and Analysis

This section describes the experiment environment, hardware, and toolkits used to implement the IoTCaP platform. The results obtained are also discussed in the light of other efforts and based on different stakeholders' perspectives.

A. Experiment Environment, Hardware, and Toolkits

An Ubuntu 18.04 desktop machine with an i711700 8 core processor and 32 GB of ram is running eclipse, where Vert.X verticles are collecting data from hardware described in Fig. 6.

The sensors used include a DHT-22 temperature and humidity sensor, a KY-38 noise sensor, an SDS-011 particulate matter sensor for measuring air quality, a GL-Inet Acces point that provides network access to two ESP8266 boards that play two roles: provide signal strength values to the IoTCaP platform and also running a web-server/client that collect all sensor data and sends it to a custom API in the Linux-Machine where vert.X verticles are running.

B. Micro-services Toolkits

ICCF is platform agnostic; platforms are swapped when better ones are available. For this project's microservices toolkit needs, Vert.X is picked as it satisfies latency, performance, and logging requirements needed for IoTCaP as concluded from Table II. In addition, Vert.X is an approachable and efficient toolkit for writing asynchronous and reactive applications on the JVM [40].

```
\* BEGIN TRANSLATION (chksum(pcal) = "b4c94829" /\ chksum(tla) = "f7f24b1")
VARIABLES hv, RpHs, RpTs, RpNs, RpAs, RpSs, WT, WH, WS, WA, WN, pc

vars == << hv, RpHs, RpTs, RpNs, RpAs, RpSs, WT, WH, WS, WA, WN, pc >>

Init == (* Global variables *)
        /\ hv \in 1..100
        /\ RpHs \in {1,2,3,4,5}
        /\ RpTs \in {1,2,3,4,5}
        /\ RpNs \in {1,2,3,4,5}
        /\ RpAs \in {1,2,3,4,5}
        /\ RpSs \in {1,2,3,4,5}
        /\ WT \in 0..8
        /\ WH = 4
        /\ WS \in 0..8
        /\ WA = 4
        /\ WN = 4
        /\ pc = "HumidtyRangeVerification"

HumidtyRangeVerification == /\ pc = "HumidtyRangeVerification"
                            /\ IF ((hv<51)/\(hv>29))
                            THEN /\ RpHs' = 5
                            ELSE /\ IF (((hv<30)/\(hv>25))\/((hv<55)/\(hv>50)))
                            THEN /\ RpHs' = 4
                            ELSE /\ IF (((hv<26)/\(hv>20))\/((hv<61)/\(hv>54)))
                            THEN /\ RpHs' = 3
                            ELSE /\ IF (((hv<21)/\(hv>10))\/((hv<80)/\(hv>60)))
                            THEN /\ RpHs' = 2
                            ELSE /\ IF (((hv<11)/\(hv>0))\/((hv<101)/\(hv>79)))
                            THEN /\ RpHs' = 1
                            ELSE /\ TRUE
                            /\ RpHs' = RpHs
                            /\ pc' = "CompositionScoreVerification"
                            /\ UNCHANGED << hv, RpTs, RpNs, RpAs, RpSs, WT, WH,
                                            WS, WA, WN >>

CompositionScoreVerification == /\ pc = "CompositionScoreVerification"
        /\ IF (WH+WT+WS+WA+WN<21)
        THEN /\ PrintT(((WH*RpHs)+(WT*RpTs)+(WA*RpAs)+(WN*RpNs)+(WS*RpSs)))
        ELSE /\ TRUE
        /\ pc' = "Done"
        /\ UNCHANGED << hv, RpHs, RpTs, RpNs, RpAs,
                        RpSs, WT, WH, WS, WA, WN >>

(* Allow infinite stuttering to prevent deadlock on termination. *)
Terminating == pc = "Done" /\ UNCHANGED vars

Next == HumidtyRangeVerification \/ CompositionScoreVerification
        \/ Terminating

Spec == Init /\ [][Next]_vars

Termination == <>(pc = "Done")

\* END TRANSLATION
=============================================================================
```

Figure 4. Screenshot from TLA+ indicating the composition functions to verify translated to TLA specification.

Vue.JS, a lightweight front-end platform, is used for its compatibility with the libraries needed to transmit data between Vert.X verticles and the front-end interface. Code for both IoTCaP back-end (Vert.X) and IoTCaP front-end (Vue.JS) is available in Github [41].

C. Running IoTCaP And Discussing Stakeholders Domains and Requirements

After powering the sensors, running Vert.X verticles and launching the GUI, the IoTCaP platform is up as seen in Fig. 7. An *Agent* picks domains of preference based on the smart building requirements. For example, choosing a smart building type of factory for semiconductors manufacturing would suggest a higher weight for air quality as dust is intolerable in clean rooms. On the other hand,

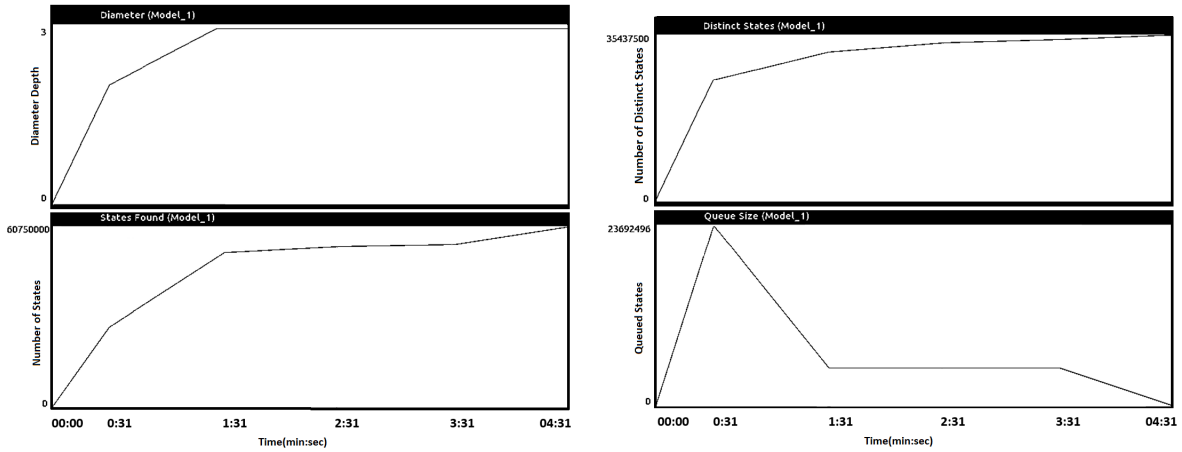

Figure 5. TLC model checking results show historical data of the well-being model's total number of generated states, distinct states, state queue handling, and the states diameter length.

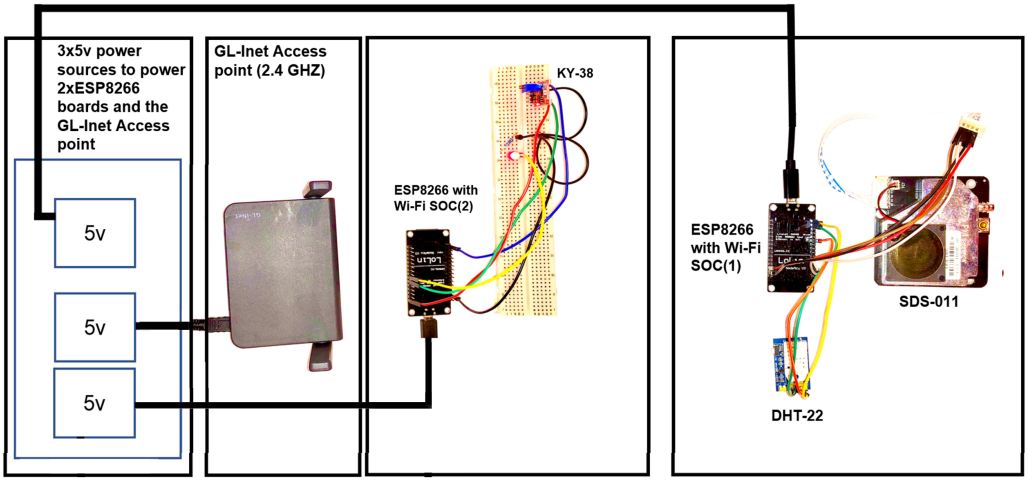

Figure 6. Experiment devices, boards, and atomic sensors: 2 X ESP8266, 1 X DHT22, 1 X SDS011, 1 X GLiNET WiFi Access, 1 X KY038, 3 X 5V DC outlets.

picking a residential building as a domain suggests equal weights for all metrics as they are equally important. The front-end interface also allows users to set thresholds for well-being and get e-mail warnings when the value sinks below specified values.

VI. Conclusion, Challenges, And Future Work

This work introduces a stakeholder-defined well-being composition capability based on the ICCF framework foundations. First, scores and weights to quantitatively compute the metric of the composite well-being capability are introduced. Second, an algebraic specification is developed using the ICCF framework semantics to describe entities and interactions. Third, formal verification is executed on the well-being model. Fourth, the core composition operations and the state-space dimensions are analyzed. Fifth, A Vert.X/Vue.JS implementation for the well-being feature is built and presented, and run-time behavior is discussed based on two stakeholders' perspectives: residential buildings and factories.

Two challenges related to this work were identified: i) when composing atomic capabilities, gaining insight on the composite capability is straightforward, but losing sight of what caused a particular state of the composite capability isn't always easy. A suggested solution is to leverage AI by assigning profiles to atomic capabilities, which would lead to a better understanding of what caused a particular state of a composite capability.

An identical issue was addressed in [47], where researchers recognized appliances that consume the most energy based on their energy profile. ii) The second challenge is when running a model checking for many variables and values; it might take a long time to execute

Table II. COMPARING IMPLEMENTATION TECHNOLOGIES.

Technology	Ref	Performance	Long running Processes	Logging	Latency [ms]	Application Related Pros(+) and Cons(-)
Vert.X	[40]	EventLoop model (less overhead, less CPU usage)	Supported (JVM Support)	Vert.X verticles offer significantly faster messages for error recovery compared to Akka	1.8	(+)More flexibility with callbacks, futures, Java Completion Stage, Kotlin coroutines, RxJava, and fibers support. (-) Many conflicting versions.
Node.js	[42]	reactor pattern (High CPU usage, high memory usage)	Supported	Errors in Node.js are handled through exceptions.	1.2	* Both Node.js and Vert.X give scalable server-side applications, but Vert.X scales better than Node.js.
Akka	[43]	Actor model (less overhead, moderate CPU usage)	Supported (JVM Support)	Akka actors offer messages for error recovery	2.8	* Vert.X is more suited for a wider variety of tasks, while Akka fits more for designing large systems with several subsystems that handle humongous concurrency.
Spring Framework	[44]	(Slow startup time, high load, High Heap memory usage)	Supported (JVM Support)	Default Logback Logging	4.2	(-)Spring is less flexible and slower than vert.X and uses blocking APIs
Quarkus	[45]	Vertx EventLoop (less overhead, high CPU usage)	Supported (JVM Support)	JBoss Log Manager	4.7	(+)Runs exceptionally well in container environments like Kubernetes. Vert.X is used to power the Quarkus networking stack.
Netty	[46]	memory and CPU overhead	Supported (JVM Support)	JdkLoggerFactory	1.3	(+)Provides non-blocking I/O APIs for the JVM. Slightly faster than Vert.X (-)APIs low-level compared to Vert.X

Figure 7. Screenshot of the Vue.JS IoTCaP front-end, an actor can fill in the weights for the capabilities based on his/her preferences and submit a specification to the well-being back-end, a result is returned that carries both atomic capabilities as well as the composite capability of well-being based on inserted weights.

the symbolic execution. A solution to this challenge would involve leveraging cloud instances to run TLC on multiple AWS or Azure instances and assessing the benefits of this approach.

REFERENCES

[1] C. Greer, M. Burns, D. Wollman, and E. Griffor, "Cyber-physical systems and internet of things," 2019.

[2] R. Jia, B. Jin, M. Jin, Y. Zhou, I. C. Konstantakopoulos, H. Zou, J. Kim, D. Li, W. Gu, R. Arghandeh *et al.*, "Design automation for smart building systems," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1680–1699, 2018.

[3] A. Schieweck, E. Uhde, T. Salthammer, L. C. Salthammer, L. Morawska, M. Mazaheri, and P. Kumar, "Smart homes and the control of indoor air quality," *Renewable and Sustainable Energy Reviews*, vol. 94, pp. 705–718, 2018.

[4] P. Wolkoff, "Indoor air humidity, air quality, and health–an overview," *International journal of hygiene and environmental health*, vol. 221, no. 3, pp. 376–390, 2018.

[5] K. Huang, J. Song, G. Feng, Q. Chang, B. Jiang, J. Wang, W. Sun, H. Li, J. Wang, and X. Fang, "Indoor air quality analysis of residential buildings in northeast china based on field measurements and longtime monitoring," *Building and Environment*, vol. 144,

pp. 171–183, 2018.

[6] S. Das, "Technology for smart home," in *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals & Systems and Networking (VCASAN-2013)*. Springer, 2013, pp. 7–12.

[7] M. Leung, A. Chan, and P. Lam, "Control and management of hospital indoor air quality," *Med Sci Monit*, vol. 12, no. 3, p. 23, 2006.

[8] J. Li and Y.-F. Zhou, "Occupational hazards control of hazardous substances in clean room of semiconductor manufacturing plant using cfd analysis," *Toxicology and industrial health*, vol. 31, no. 2, pp. 123–139, 2015.

[9] S. L. McGregor and E. B. Goldsmith, "Expanding our understanding of quality of life, standard of living, and well-being," *Journal of Family and Consumer Sciences*, vol. 90, no. 2, p. 2, 1998.

[10] K. Halba, E. Griffor, A. Lbath, and A. Dahbura, "A framework for the composition of iot and cps capabilities," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2021, pp. 1265–1272.

[11] E. R. Griffor, C. Greer, D. A. Wollman, and M. J. Burns, "Framework for cyber-physical systems: Volume 1, overview, version 1.0," *NIST Special Publication*, pp. 1500–201, 2017.

[12] E. R. Griffor, C. Greer, D. A. Wollman, M. J. Burns *et al.*, "Framework for cyber-physical systems: Volume 2, working group reports," 2017.

[13] D. A. Wollman, M. A. Weiss, Y. Li-Baboud, E. R. Griffor, and M. J. Burns, "Framework for cyber-physical systems: Volume 3, timing annex," 2017.

[14] B. Trammell, M. Mellia, A. Finamore, S. Traverso, T. Szemethy, B. Szabo, D. Rossi, B. Donnet, F. Invernizzi, and D. Papadimitriou, "mplane architecture specification."

[15] L. Lamport, *Introduction to TLA*. Digital Equipment Corporation Systems Research Center [SRC], 1994.

[16] I. Konnov, J. Kukovec, and T.-H. Tran, "Tla+ model checking made symbolic," *Proceedings of the ACM on Programming Languages*, vol. 3, no. OOPSLA, pp. 1–30, 2019.

[17] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal *et al.*, "Brick: Metadata schema for portable smart building applications," *Applied energy*, vol. 226, pp. 1273–1292, 2018.

[18] P. Bellagente, P. Ferrari, A. Flammini, and S. Rinaldi, "Adopting iot framework for energy management of smart building: A real test-case," in *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. IEEE, 2015, pp. 138–143.

[19] S. N. Han, I. Khan, G. M. Lee, N. Crespi, and R. H. Glitho, "Service composition for ip smart object using realtime web protocols: Concept and research challenges," *Computer Standards & Interfaces*, vol. 43, pp. 79–90, 2016.

[20] C. Marche, M. Nitti, and V. Pilloni, "Energy efficiency in smart building: a comfort aware approach based on social internet of things," in *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 2017, pp. 1–6.

[21] N. Cauchi, K. A. Hoque, A. Abate, and M. Stoelinga, "Efficient probabilistic model checking of smart building maintenance using fault maintenance trees," in *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, 2017, pp. 1–10.

[22] A. Urbieta, A. González-Beltrán, S. B. Mokhtar, M. A. Hossain, and L. Capra, "Adaptive and context-aware service composition for iot-based smart cities," *Future Generation Computer Systems*, vol. 76, pp. 262–274, 2017.

[23] O. Y. Maryasin, A. Kolodkina, and A. Ogarkov, "Computer simulation of a smart building," *Automatic Control and Computer Sciences*, vol. 53, no. 7, pp. 787–793, 2019.

[24] M. V. Moreno, M. A. Zamora, and A. F. Skarmeta, "User-centric smart buildings for energy sustainable smart cities," *Transactions on emerging telecommunications technologies*, vol. 25, no. 1, pp. 41–55, 2014.

[25] M. Jia, A. Komeily, Y. Wang, and R. S. Srinivasan, "Adopting internet of things for the development of smart buildings: A review of enabling technologies and applications," *Automation in Construction*, vol. 101, pp. 111–126, 2019.

[26] R. Yang and L. Wang, "Multi-agent based energy and comfort management in a building environment considering behaviors of occupants," in *2012 IEEE Power and Energy Society General Meeting*. IEEE, 2012, pp. 1–7.

[27] E. I. Batov, "The distinctive features of "smart" buildings," *Procedia Engineering*, vol. 111, pp. 103–107, 2015.

[28] N. Y. Jadhav, *Green and smart buildings: advanced technology options*. Springer, 2016.

[29] T. M. Lawrence, M.-C. Boudreau, L. Helsen, G. Henze, J. Mohammadpour, D. Noonan, D. Patteeuw, S. Pless, and R. T. Watson, "Ten questions concerning integrating smart buildings into the smart grid," *Building and Environment*, vol. 108, pp. 273–283, 2016.

[30] Q. P. Ha, S. Metia, and M. D. Phung, "Sensing data fusion for enhanced indoor air quality monitoring," *IEEE Sensors Journal*, vol. 20, no. 8, pp. 4430–4441, 2020.

[31] O. García, P. Chamoso, J. Prieto, S. Rodríguez, and F. de la Prieta, "A serious game to reduce consumption in smart buildings," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer, 2017, pp. 481–493.

[32] P. Pappachan, M. Degeling, R. Yus, A. Das, S. Bhagavatula, W. Melicher, P. E. Naeini, S. Zhang, L. Bauer, A. Kobsa *et al.*, "Towards privacy-aware smart buildings: Capturing, communicating, and enforcing privacy policies and preferences," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2017, pp. 193–198.

[33] M. R. Bashir and A. Q. Gill, "Towards an iot big data analytics framework: smart buildings systems," in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2016, pp. 1325–1332.

[34] G. Trencher and A. Karvonen, "Stretching "smart": Advancing health and well-being through the smart city agenda," *Local Environment*, vol. 24, no. 7, pp. 610–627, 2019.

[35] T. Martinez, M. Duarte, and A. C. Garcia-Luna, "How using smart buildings technology can improve indoor environmental quality in educational buildings," in *SHS Web of Conferences*, vol. 102. EDP Sciences, 2021, p. 03003.

[36] E. Markoska, N. Jakica, S. Lazarova-Molnar, and M. K. Kragh, "Assessment of building intelligence requirements for real time performance testing in smart buildings," in *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE, 2019, pp. 1–6.

[37] P. Fazenda, P. Carreira, and P. Lima, "Context-based reasoning in smart buildings," in *Proceedings of the first international workshop on information technology for energy applications*, vol. 923, 2012, pp. 131–142.

[38] R. Vedvik, "Understanding well v2 certification: As well v2 certification becomes more prevalent, project managers, engineers and designers need to understand which preconditions and optimization features are impacted by the systems they design." *Consulting Specifying Engineer*, vol. 58, no. 6, pp. 24–31, 2021.

[39] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff, "How amazon web services uses formal methods," *Communications of the ACM*, vol. 58, no. 4, pp. 66–73, 2015.

[40] J. Ponge, *Vert. x in Action*. Manning Publications, 2020.

[41] K. HALBA and A. Hailane, "Iotcap : A well-being platform based on the iccf framework using vert.x as a back-end and vue.js as a front-end." https://github.com/usnistgov/ICCF, date accessed 02-02-2022.

[42] M. Cantelon, M. Harter, T. Holowaychuk, and N. Rajlich, *Node. js in Action*. Manning Greenwich, 2014.

[43] R. Roestenburg, R. Williams, and R. Bakker, *Akka in action*. Simon and Schuster, 2016.

[44] C. Walls, *Spring Boot in action*. Simon and Schuster, 2015.

[45] T. Koleoso, "Microservices with quarkus," in *Beginning Quarkus Framework*. Springer, 2020, pp. 51–132.

[46] N. Maurer and M. Wolfthal, *Netty in action*. Simon and Schuster, 2015.

[47] P. Franco, J. M. Martínez, Y.-C. Kim, and M. A. Ahmed, "A framework for iot based appliance recognition in smart homes," *IEEE Access*, vol. 9, pp. 133 940–133 960, 2021.