A Framework for the Composition of IoT and CPS Capabilities

Khalid Halba[1,2,3], Edward Griffor[2], Ahmed Lbath[1] and Anton Dahbura[3]

[1]Grenoble Alpes University, Grenoble, France
{khalid.halba , ahmed.lbath}@univ-grenoble-alpes.fr
[2]National Institute of Standards and Technology, Gaithersburg, Maryland, USA
{khalid.halba , edward.griffor}@nist.gov
[3]Johns Hopkins University, Baltimore, Maryland, USA
{khalba1 , adahbura1}@jhu.edu

***Abstract*—By 2030, over a half trillion devices will be connected to the internet. With so many devices providing a wide range of features, there is a need for a framework for innovation and reuse of Internet of Things (IoT) and Cyber-Physical Systems (CPS) capabilities. Such framework should facilitate the composition of capabilities and provide stakeholders means to reliably model and verify compositions. An IoT and CPS Composition Framework (ICCF) is proposed to achieve this goal. ICCF is based on the NIST CPS framework composition guidelines, intuitive composition semantics inspired from the mPlane protocol, and strong formal verification capabilities of the Temporal Logic of Actions (TLA) formal descriptors and tools. This paper demonstrates why such framework, semantics, and formal specification and verification components form a powerful and intuitive composition framework that satisfies different stakeholders concerns. To achieve this purpose, semantics and formal specification of the composition algebra were provided, a well-being composite capability within a smart building was specified, its prototype model in a formal verification tool was run, an analysis of the results of symbolic execution quantitatively and qualitatively was performed, and assessment of the trustworthiness of the composition was done. Lastly, implementation details were provided and proposed extensions to other domains such as smart transportation and smart health were discussed.**

***Index Terms*—Framework, IoT, CPS, ICCF, Capability, Algebra, Composition, Trustworthiness.**

I. Introduction and paper plan

IoT or CPS capabilities composition is the process of generating a value-added capability based on atomic measurements or services. Throughout this paper, IoT and CPS can be used interchangeably [1]. A framework for capabilities composition that addresses different stakeholders concerns would serve as a foundation for open innovation and re-purposing of IoT and CPS capabilities of an expected half-trillion IoT and CPS devices by 2030 [2]. Examples of target domains include smart buildings (well-being), transportation (safety of autonomous vehicles), and healthcare (autonomous ventilators output). Verifying such novel compositions and making sure their deployment won't cause errors is crucial for a trustworthy implementation. For this reason, there is a need for a framework for composing IoT or CPS systems capabilities regardless of their complexity. This work proposes an **I**oT and **C**PS **C**omposition **F**ramework (**ICCF**) that addresses these goals. Such a framework should lay the groundwork for composition and trustworthiness assessments, use straightforward semantics to help developers prototype novel capabilities, and describe tools for the formal verification of such novel composite capabilities. Capabilities composition taxonomy involves formal, technical, and Quality of Service (QoS) components. Together these components provide foundations for representing, processing, and generating sought-after capabilities by diverse stakeholders while preserving properties of interest to those stakeholders. The focus of this paper is on formal components of the proposed framework, technical and QoS aspects remain for future work. Formal components of ICCF include i) foundations for composition ii) semantics for capabilities, interactions, and compositions iii) formal specification languages and formal verification tools used to translate the semantics of composition into specifications for performing model checking and trustworthiness assessment via assertions or deadlock analysis. The contributions of this paper are organized as follows: in section II, related work on previous efforts is provided. In section III: i) criteria based on which the selection of the building blocks of the proposed composition framework is defined, ii) existing frameworks, semantics, and formal verification techniques are compared to select formal components that best

satisfy ICCF requirements. Section IV provides ICCF semantics used to describe interactions and capabilities compositions. In section V an example of composition in the IoT space is presented, its composite function is formally specified and its state space is studied following its model checking. This is followed by quantitative and qualitative analysis and an assessment of the trustworthiness of the results obtained, as well as an outlook on implementation efforts of the studied example and potential extensions to other domains. In the conclusion, a summary of contributions is presented as well as an outlook to future work.

II. Related work

Examples of IoT and CPS frameworks, environments, or standards for capabilities composition include OneM2M environment [3]. It leverages the NIST CPS framework [4] [5] [6], a comprehensive framework that provides capabilities composition guidelines including time synchronization between atomic capabilities. The OneM2M environment can use the M2M semantics provided by the industry segment that uses corresponding data. This makes its semantics domain-specific [7], a higher abstraction layer might be needed to simplify rapid prototyping of composition for different IoT and CPS domains. Fiware [8] was coupled with the IoT-A framework which supports IoT capabilities composition and semantic specification using Business Process Model and Notation (BPMN) 2.0, it also supports powerful features such as synchronous and asynchronous compositions [9]. The BMPN semantics, however, make it challenging to formally verify compositions as that involves converting the BPMN notation to the Generic Property Specification Language (GPSL formal) specification language which can improve expressiveness but might add complexity, impact performance, or limit expressiveness when converting BMPN to a graphically verifiable model such as Property Sequence Chart (PSC) [10]. For the CIM (Context Information Management) environment [11], the foundations for composition are provided by the CIM NGSI evolution framework, it uses RDF (Resource Description Framework) to semantically describe the capabilities of a system. RDF is a graph-based descriptive language, it can be converted to a formally verifiable specification such as ShEx 2.0 (Shape expression schemas v2.0) [12]. ShEx expressions can be used both to describe RDF and to automatically check the conformance of RDF data. However, ShEx checks whether RDF data respects the schema requirements as it is data-oriented not composition function-oriented. This can make it challenging to model check the system features. VITAL is another project that supports IoT-A framework, W3C SSN semantics, but recommendation on formal specification and verification languages and tools to use are not the focus of the framework [13]. The same case for FogFlow [14], an environment that leverages NGSI framework for IoT capabilities composition foundations and YAML as a capability descriptor. AWS [15] is a commercial environment for cloud services, it leveraged PlusCal semantics [16] and TLA [17] formal specification techniques to verify the correctness of properties such as fault tolerance in their storage services, but this didn't extend to cover the microservices and IoT composition solutions such as GreenGrass [18]. To address the limitations of these frameworks and to provide a strong framework for the composition of IoT and CPS capabilities, an IoT and CPS Composition Framework (ICCF) is proposed. This framework leverages the NIST CPS framework composition and trustworthiness recommendations, uses strong semantics inspired from the mPlane protocol [19], and relies on the intuitive PlusCAL/TLA/TLA+ package to prototype, formally specify, and model check capabilities and assess their trustworthiness.

III. Formal criteria and ICCF foundations

A. Formal criteria

Satisfying the formal criteria of the ICCF framework means relying on a framework that provides capabilities composition foundations. This also requires the leverage of lightweight and expressive semantics to describe compositions and being able to translate their semantics easily to a formal specification language for building complex functions and verifying their trustworthiness. Below is a comparison of frameworks, semantics, and formal verification techniques that aims to select components providing the formal components necessary to satisfy the requirements of ICCF.

B. Comparing formal components

1) Frameworks: A framework enables composition when it takes into consideration i) concerns related to the ability of the IoT/CPS to achieve an intended purpose in the face of changing external conditions such as the need to upgrade or otherwise reconfigure an IoT/CPS to meet new conditions, needs, or objectives (adaptability), ii) concerns related to our understanding of the behavior of IoT/CPS due to the richness and heterogeneity of interactions among its components, such as the existence of legacy components and the variety of interfaces (complexity), iii) concerns related to the ability to combine IoT/CPS modular components (hardware, software, and data) to satisfy user requirements (constructivity), vi) concerns related to the ease and reliability with which an IoT/CPS component can

be observed and understood (for purposes of leveraging the component's functionality) by an entity (human, machines), and v) concerns related to the ease and reliability with which an IoT/CPS component's functions can be ascertained (for purposes of leveraging that functionality) by an entity (discoverability). IoT-A is a reference model and architecture (RMA) designed to allow the generation of different IoT architectures tailored to specific scenarios. Using IoT-A with Fiware in [8] enabled the creation of architectures with different functional groups each serving a specific purpose and enabling interoperability, the composition of functions is intrinsic to Fiware using the service organization FG (functionality group) but stakeholders concerns when composing IoT capabilities aren't explicitly addressed. In [20], an OWL-based ontological framework for the opportunistic composition of IoT systems was introduced. The framework leverages holons, which are programming entities used to model distributed systems. Designing holons uses CoAMOS and A3ME ontologies. The resulting ontologies can then be converted to UML or domain-specific languages for further exploitation or composition in the IoT domain. While the capabilities discoverability or composition complexity are addressed in this framework, the adaptability of the composition isn't addressed. In [21], ISCO, (Internet of Smart City Objects), a distributed framework for service discovery and composition was introduced with three major enablers: a semantic functional description of city objects, representing physical devices or abstract services, a distributed service directory that embodies available city services for service lookup and discovery, and planning tools for selecting and chaining basic services to compose new complex services, this effort provides rich implementation aspects in the smart city context but the trustworthiness of composed capabilities isn't addressed. The NIST CPS framework [4] defines criteria that contribute to CPS composition trustworthiness taking into consideration functional, human, trustworthiness, timing, data, and composition concerns. The composition concern addresses adaptability, complexity, constructivity, and discoverability of CPS capabilities, hence, the NIST CPS framework composition foundations are leveraged to guide stakeholders concerns for composing capabilities.

2) Semantics: Semantics for capabilities composition suggest that the semantics are lightweight and expressive enough to represent capabilities, interactions, compositions, and workflows necessary to compose value-added features in IoT and CPS. The W3C Web Ontology Language (OWL) [22] is a semantic web language designed to represent complex and rich IoT capabilities, groups of things, and relations between things. However, it is more geared toward web services and it is not a lightweight approach to composing services. In [23], CyPhyML, a CPS capability description language was discussed with formal specification capabilities supported. However, the language was more geared towards a formal description of CPS systems for model checking purposes and not IoT services. mPlane semantics [19] allow the representation of value-added capabilities using a set of operations designed to facilitate service composition. These compositions can be applied to measurement environments, IoT services, and CPS. mPlane semantics are simple, expressive, and lightweight compared to other description languages investigated, as a result, it satisfies the human aspect of the NIST CPS framework.

3) Formal specification and verification: This aspect is related to the semantics discussed earlier. Building correct compositions and verifying their properties should not be a daunting experience for engineers and developers. These stakeholders should be able to easily use semantics and service descriptors to formally specify and prototype composite services. In [24] authors introduced linear logic LL based on pi-Calculus to describe and formally verify non-functional attributes such as the credibility/trustworthiness of the service composition. Linear temporal logic (LTL) was introduced in [25]. Real-Time Maude formal verification tool that is based on LTL was used to formally verify properties of interest. In [26], Directed acyclic graphs were used to formally model dynamic service discovery, invocation, and composition in opportunistic networks. Petri Nets [27] were used as an algebra to formally model services and processes where the main goal was to formally verify compliance of compositions with the ever-changing regulations on IoT. Temporal Logic of Actions formal specification was used to formally specify and verify critical properties on services within the AWS echo-system [15]. The common aspect between these formal specification languages is how difficult it is to move from description semantics to formal specification of composite services. Except for TLA, which has strong software support using PlusCAL, a high-level language that is comparable to pseudocode and which enables the fast translation of mPlane composition semantics to TLA formal specification. TLA+ is the formal tool that uses notation which is very similar to natural mathematic operations. CoQ [28] or Isabelle [29] use relatively daunting notations that are challenging for stakeholders which might impact the developer's ability to write verifiable compositions and as a result might limit innovation. Based on this comparative study of frameworks, service description semantics, and formal specification and verification

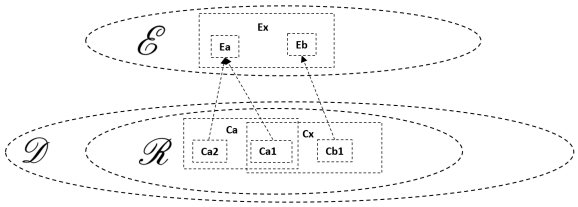

Fig. 1. Space of Capabilities and Entities

techniques ICCF is proposed: a framework for the composition of IoT and CPS capabilities that is based on strong composition foundations provided by the NIST CPS framework, easy and lightweight semantics inspired from the mPlane protocol and leverages the ability to quickly translate service semantics into formal specification thanks to PlusCAL's translation capabilities of composition semantics to TLA specification.

IV. Introducing ICCF composition algebra semantics

This section defines the algebra for describing capabilities, interactions, and compositions.

A. The space of capabilities and descriptors

Figure 1 shows the space of entities and capabilities. $\mathscr{E}$ represents the space of IoT entities, while $\mathscr{D}$ represents the space of capability descriptors. There is a space $\mathscr{R}$ in $\mathscr{D}$ that meets $ICCF$ requirements. $\mathscr{R}$ elements can be composed and decomposed using the ICCF framework specification algebra. There is a $surjective$ relationship between $\mathscr{D}$ and $\mathscr{E}$: one or more $CapabilityDescriptors$ are provided by a single entity ($Ca1$ and $Ca2$ provided by Ea). In the implementation, using microservices, an exception to this rule are those microservices that provide a single and unique $CapabilityDescriptor$ ($Cb1 \rightarrow Eb$ represents this case). If $\mathscr{E}$ is composed of such microservices then the relation between $\mathscr{R}$ and $\mathscr{E}$ is $injective$. Capabilities in $\mathscr{R}$ are either indecomposable or composite.

B. Composition operators and descriptors

Let's define an operator ψ which represents a k-ary composition operator. To illustrate composition in this paper, an assumption of k=2 is considered (which renders ψ a binary composition), Ca1, Ca2, and Ca are represented as JSON objects (with simple key-value pairs representing the $CapabilityDescriptor$ parameters), the composition is an operator on values obtained after sending a specification to all atomic capabilities and receiving results. Let's consider $Ca1$ and $Ca2$ from Figure 1 two indecomposable capabilities and Ca a composite capability obtained as follows:

The composition operator ψ has outcomes in $\mathscr{R}$:

ψ: $\mathscr{R}^2 \rightarrow \mathscr{R}$.

(Ca1,Ca2) $\rightarrow$ Ca

$\rightarrow \psi$ (Ca1,Ca2)

This composition generates the $CapabilityDescriptor$ of the composite capability Ca described below:

```
{ "ID": "Ca_ID",
  "Organization": "Ca_O",
  "NAME": "Ca_N",
  "TIMESTAMP": "Ca_TS",
  "LOCATION": "Ca_L",
  "REFRESH_RATE": "Ca_RR",
  "UNIT": "Ca_U",
  "VALUE": "Ca_V",
  "SIGNATURE": "Ca_S"}
```

Ca_ID: represents the ID of the composite capability. It is an increment of the last ID registered in the CMr registry. Ca_N and Ca_O are the Name and the Organization of the new composite capability respectively, a new name and organization are attributed to the composite parameters when the indecomposable capabilities have different ones. Ca_TS: time of arrival of the composite capability. Ca_L represents the physical location (geographical in terms of latitude and longitude) or logical location (IP address). In the case of a geographical address the composite location is the location that comprises indecomposable capabilities' locations. For logical locations, If the sensors reside in the same IP Subnet then the subnet that comprises their IP address becomes their composite location. Ca_RR represents the frequency at which a measurement is received. A composite value for this parameter should be the longest refresh rate: $Ca\_RR \longleftarrow$ MAX{$Ca1\_RR$,$Ca2\_RR$}. Ca_U reflects the nature and unit of the composite capability. The simple example of power consumption as a composite capability of both current and voltage takes the "Watt" as the composite Unit of "Amperes" and "Volts". In other cases such as well-being in a smart building, indecomposable capabilities such as temperature, humidity, and air quality have different units and the composition algorithm depicts the composite unit. Ca_V: represents the value of the composite capability. IoT providers have the flexibility to define and introduce parameters customized to their composition needs. One such customization is the introduction of weights and multipliers. For example, $Ca\_V \leftarrow \alpha Ca1\_V + \beta Ca2\_V$ where α and β are two doubles that represent the weight of C_a1V and C_a2V, respectively, and (+) a composition operator. Composition rules and parameters are nested in the programmable extension of the indecomposable capabilities descriptors. This addresses the constructivity concern of the NIST CPS framework as the ability to compose capabilities of different units and sources in a modular way would allow more innovation.

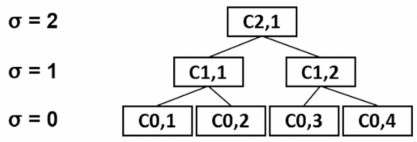

Fig. 2. Composition Hierarchy

C. Capability Hierarchy and Level

Composite capabilities can be further composed into more complex capabilities. $C_{1,1}$ and $C_{1,2}$ in Figure 2 are an example of this case. The hierarchy level σ ranks the capabilities complexity. Every capability can be expressed as follows: $C\sigma, y$, where σ is the hierarchy level and y is the id of the capability at that level. If $\sigma(C)$ =0, the capability is indecomposable. The composite capability descriptor enables tracking the ancestry of the capabilities and verification of their source without directly sending a request to the producing entities. This paradigm addresses the composition complexity concern of the NIST CPS framework.

D. Specifications, results, and interactions

1) Specifications and results: A $Specification$ (sC) for a capability with a descriptor (C) is a request sent from an entity to a resource to get information. The $Specification$ contains information about the capability that helps intermediate entities (including proxies and capability managers) to locate the requested capability. A $Result$ is a $Specification$ for which all the parameters are known. The $Result$ (rC) can be represented as the solution for a system of equations with all the parameters resolved. The space of solutions can contain a unique element, multiple elements, or no element. Below is an example of a $Specification$ represented as a system of equations where the only unknown parameter is C_V: the value of the capability. The other parameters are known as depicted in equation 1. The $Result$ is a unique solution to the $Specification$ as depicted in equation 2.

2) Discovery interaction: The capability manager discovers entities that verify the following rule: $CapabilityDescriptor \in \mathscr{R}$. the discoverability function is defined as Disc(CM, E), it takes CM, a capability manager, and E, an entity as input and returns a binary based on whether or not a capability is discovered. This addresses discoverability, one of the NIST CPS framework composition concerns.

3) Registration interaction: $C \in \mathscr{R} \implies C$ can be registered in CMr. The above implies that all k-ary compositions ψ can be applied to C. The Reg(CM,C) function is a binary function that takes CM (a capability manager) and C (the entity's $CapabilityDescriptor$) as inputs and returns True or false depending on whether or not the descriptor is stored in the CMr and the composition algorithms nested in its programmable extension are stored in the CMt.

$$sC \leftrightarrow \begin{cases} C_{ID} = ID_{value} \\ C_O = O_{value} \\ C_N = N_{value} \\ C_{TS} = TS_{value} \\ C_L = L_{value} \\ C_{RR} = RR_{value} \\ C_U = U_{value} \\ C_V = ? \\ C_S = S_{value} \end{cases} \quad (1)$$

$$rC \leftrightarrow \begin{cases} C_{ID} = ID_{value} \\ C_O = O_{value} \\ C_N = N_{value} \\ C_{TS} = TS_{value} \\ C_L = L_{value} \\ C_{RR} = RR_{value} \\ C_U = U_{value} \\ C_V = V_{value} \\ C_S = S_{value} \end{cases} \quad (2)$$

E. SendSpec(Src,Dst,Specification) interaction

It is a request sC sent to a resource, a proxy, or a capability manager. If C represents a composite capability $CapabilityDescriptor$, the $Specification$ sC will be decomposed to its indecomposable $Specifications$ $(sC1,sC2,...,sCk)$ first by applying the decomposition operator ψ^{-1} as follows:

$$\begin{aligned} \psi^{-1}\text{:} \quad & \mathscr{R} \rightarrow \mathscr{R}^k. \\ & (sC) \rightarrow (sC1, sC2, \ldots, sCk) \\ & \rightarrow \psi^{-1}(sC) \end{aligned}$$

F. sendResult(Src,Dst,Result) interaction

Entities directly provide a $Result$ for the $Specification$ if it doesn't require further composition or if it is available in the cache CMc. This explains how the adaptability concern of the CPS framework is addressed. Composite $Results$ require composition.

$$\begin{aligned} \psi\text{:} \quad & \mathscr{R}^k \rightarrow \mathscr{R}. \\ & (rC1, rC2, \ldots, rCk) \rightarrow rC \\ & \rightarrow \psi(rC1, rC2, \ldots, rCk) \end{aligned}$$

G. $ICCF$ semantics/algebra example

$ICCF$ algebra helps in expressing abstract interactions (discovery, registration, composition, decomposition, specification, results) and enables formal verification, symbolic execution, and making sure the outputs of a system fall within trustworthy values. Pseudo-code in Algorithm 1 summarizes all these operations in a use case explained in the section. The space of capabilities is described above via an example depicted in Figure 1. Let's consider CM, a capability manager, Ex an entity that provides a composite $CapabilityDescriptor$ Cx from indecomposable capabilities $Ca1$ and $Cb1$. These indecomposable capabilities are provided by entities Ea and Eb. Ex requests a composite capability Cx from the nearest CM. CM checks its CMc as to whether a copy of the $Result$ rCx is available. If this is the case, CM returns the data to to Ex. Otherwise, CM sends

requests to entities Ea and Eb based on information in the CMr. These latter respond by sending their $Results$ back to CM. The capability manager performs composition of the $Result\ rCx$ based on algorithms in the CMt and sends it back to Ex.

Algorithm 1 ICCF Protocol

if $Ca1 \in \mathscr{R}$ and $Cb1 \in \mathscr{R}$ **then**
    Disc(CM, (Ea,Eb))← true and
    Reg(CM,($Ca1$,$Cb1$))← true
    sendSpec(Ex, CM, sCx)
if $rCx \in CMc$ **then**
    sendResult(CM, Ex, rCx)
else
    ψ^{-1}(sCx)→($sCa1$, $sCb1$)
    sendSpec(CM, Ea, sCa1) and
    sendSpec(CM, Eb, $sCb1$)
    sendResult(Ea, CM, $rCa1$) and
    sendResult(Eb, CM, $rCb1$)
    ψ($rCa1$,$rCb1$)→(rCx)
    sendResult(CM, Ex, rCx)

So far, ICCF composition interactions and operations were described using semantics inspired from the intuitive mPlane platform and following the capabilities composition guidelines of the NIST CPS framework.

V. Example: Formal specification and assessment of a composite capability: well-being in a smart building

A. Experiment Description

In this section a composition case is studied and its trustworthiness is assessed. It involves composing multiple metrics to get a value-added feature. The example under study is well-being in a smart building: this feature depends on multiple sensor inputs from multiple entities including temperature, humidity, pollution level, and safety sensors in the smart building under study. The well-being composite capability $C5$ is represented as follows:

ψ**:**

$$(rC0\,, rC1\,, rC2\,, rC3\,, rC4) \ \rightarrow\ rC5$$

ψ is the composition operator which represents in this case arithmetic and logical operations on the atomic features $C0$, $C1$, $C2$, $C3$, $C4$ that generate the composite capability $C5$. The goal is to prototype a composition with an assurance, which means the composite feature's values must fall in a trustworthy range. To simplify the specification, the focus is shifted towards the composition of the capabilities data, assuming discovery, registration, and other mPlane protocol interactions are already performed. The composite feature's value, in this case, is a range from 1 to 4 stars representing the level of well-being achieved, with 3 or 4 stars are the trustworthy levels. If security level $rC3$ is not satisfied well-being's value is 0 stars as it is a mandatory aspect. This simple description satisfies the human concern of the NIST CPS framework which guides this implementation. The timestamp is synced across all capabilities values which are discrete. Figure 3 shows the well-being model in PlusCal and its translation to TLA+. Figure 4 shows the range of values generated by each capability and the trustworthy boundaries. Figure 5 shows results after running symbolic execution. The model was run on a Windows Server VM equipped with 4 i9 CPU cores and 24 GB of RAM. TLA+ offers allows connection to remote AWS performant resources to analyze complex and demanding specifications. Figure 6 shows an instance of checking a deadlock state that yields a non-trustworthy outcome.

B. Qualitative and Quantitative Analysis

The symbolic execution of the model results in running combinations of all atomic capabilities values to determine the well-being state space. In Figure 4 it took 18 seconds to perform symbolic execution, APALACHE Model Checker can replace TLC to improve execution time [30]. From Figure 5, the number of states generated across all combinations is 14382900, with 5821200 duplicate states, which means more optimization is required. The queue suffered from congestion instantly after execution but it was emptied over. Through this experiment we demonstrated how to prototype a composition based on the framework and semantics proposed, run symbolic execution, analyze trustworthy results, and reveal errors using a deadlock invariant. Other examples that could benefit from this framework include preventing oxygen toxicity in autonomous ventilators (smart health applications) or studying braking time as a composite feature in an autonomous vehicle to evaluate the braking amount required to prevent collisions (smart transportation applications). Minimizing execution time, queue congestion, and the effect of state-space explosion can be done by optimizing the model, space reduction, or leverage of better hardware (local/cloud) which TLA+ allows.

C. Implementation efforts

ICCF composition framework is agnostic from the implementation perspective and its principles can be extended to multiple environments. Vert.X, a reactive and event-driven programming tool is used to implement the well-being composite feature based on ICCF

```
---------------------- MODULE Wellbeing ----------------------------
EXTENDS Naturals, TLC
CONSTANT TL, TH, HL, HH, PL, PH, WBDV, WBH, WBL
(*
--algorithm WellBeing {
  variables tm\in 1 .. 60;
            s \in 0 .. 3;
            t \in 60 .. 80;
            ts = 0;
            h \in 30 .. 50;
            p \in 0 .. 10;
            WB \in 1 .. 5;
  {
  tm:= tm + 1;
  WB := 0;
  if(s=3) {
    WB := WB + 1;
    if((t<TH)/\(t>TL)){ WB := WB + 1;};
    if((h<HH)/\(h>HL)){ WB := WB + 1;};
    if((p<PH)/\(p>PL)){ WB := WB + 1;};
          };
  else
    {WB := 0;}}
}
*)
\* BEGIN TRANSLATION (chksum(pcal) = "f4b3a373" /\ chksum(tla) = "821e8820")
VARIABLES tm, s, t, ts, h, p, WB, pc
vars == << tm, s, t, ts, h, p, WB, pc >>
Init == (* Global variables *)
        /\ tm \in 1 .. 60
        /\ s \in 0 .. 3
        /\ t \in 60 .. 80
        /\ ts = 0
        /\ h \in 30 .. 50
        /\ p \in 0 .. 10
        /\ WB \in 1 .. 5
        /\ pc = "Lbl_1"
Lbl_1 == /\ pc = "Lbl_1"
         /\ tm' = tm + 1
         /\ WB' = 0
         /\ IF s=3
               THEN /\ pc' = "Lbl_2"
               ELSE /\ pc' = "Lbl_6"
         /\ UNCHANGED << s, t, ts, h, p >>
Lbl_2 == /\ pc = "Lbl_2"
         /\ WB' = WB + 1
         /\ IF (t<TH)/\(t>TL)
               THEN /\ pc' = "Lbl_3"
               ELSE /\ pc' = "Lbl_4"
         /\ UNCHANGED << tm, s, t, ts, h, p >>?
Lbl_3 == /\ pc = "Lbl_3"
         /\ WB' = WB + 1
         /\ pc' = "Lbl_4"
         /\ UNCHANGED << tm, s, t, ts, h, p >>
Lbl_4 == /\ pc = "Lbl_4"
         /\ IF (h<HH)/\(h>HL)
               THEN /\ WB' = WB + 1
               ELSE /\ TRUE
                    /\ WB' = WB
         /\ IF (p<PH)/\(p>PL)
               THEN /\ pc' = "Lbl_5"
               ELSE /\ pc' = "Done"
         /\ UNCHANGED << tm, s, t, ts, h, p >>
Lbl_5 == /\ pc = "Lbl_5"
         /\ WB' = WB + 1
         /\ pc' = "Done"
         /\ UNCHANGED << tm, s, t, ts, h, p >>
Lbl_6 == /\ pc = "Lbl_6"
         /\ WB' = 0
         /\ pc' = "Done"
         /\ UNCHANGED << tm, s, t, ts, h, p >>
(* Allow infinite stuttering to prevent deadlock on termination. *)
Terminating == pc = "Done" /\ UNCHANGED vars
Next == Lbl_1 \/ Lbl_2 \/ Lbl_3 \/ Lbl_4 \/ Lbl_5 \/ Lbl_6
        \/ Terminating
Spec == Init /\ [][Next]_vars
Termination == <>(pc = "Done")
\* END TRANSLATION
-------------------------------------------------------------------
```

Fig. 3. Well-being model in PlusCal and its translation to TLA

mPlane Semantics	PlusCal/TLA representation	Description	unit	hierarchy	possible values	Trustworthy boundaries
rC0	tm	timestamp	Seconds	atomic	1-60	-
rC1	t	temperature	Fahrenheit	atomic	60-80	67-71
rC2	h	humidity	Gcm[gram/cm]	atomic	30-50	42-48
rC3	p	Air quality	Pcm[particle/cm]	atomic	0-10	0-4
rC4	s	Security level	custom	atomic	1-3	3
rC5	WB	Well being	stars	composite	1-4	3-4

Fig. 4. Capabilities and their Range of possible and accepted values for the well-being composite feature

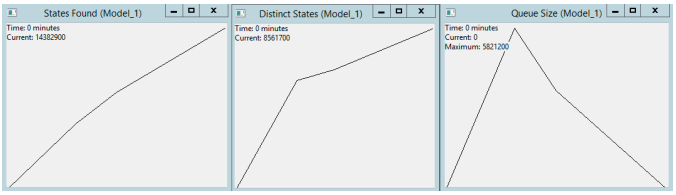

Fig. 5. Symbolic Execution Results as a function of time: States, Distinct States, Queue Size

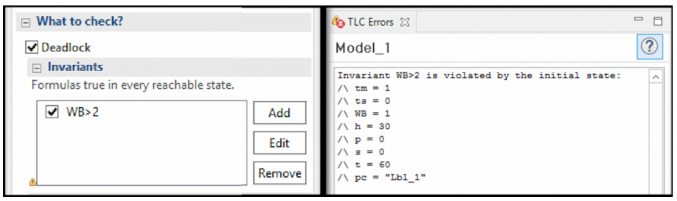

Fig. 6. Deadlock and Trustworthiness Verification

foundations. The well-being verticle receives data from temperature and humidity sensors (provided by sensor DHT22 AM2302), and air-quality sensors (provided by sensor SDS011 PM2.5). Code for the project is available in the GitHub repository [31]. An Automated Driving System testbed [32] on the NIST's UCEF co-simulation environment is also being built [33]: The goal is to be able to simulate define autonomy functions as composite features. This will enable trustworthiness assessment of safety critical maneuvers such as emergency braking.

Conclusion and Future work

The ICCF framework and its formal criteria derived from a composition-enabling framework, straightforward and expressive semantics, and strong formal verification language and techniques for composing CPS and IoT capabilities were introduced. A comparison of existing environments, frameworks, semantics, and formal specification and verification techniques enabled the selection of formal components of a composition framework that enables specification, prototyping, and assessment of IoT and CPS capabilities. The goal is to provide stakeholders the tools to innovate in the IoT and CPS space while addressing their corresponding concerns. NIST CPS framework composition guidelines, and powerful semantics inspired from the mPlane protocol, as well as formal specification and verification techniques provided by the TLA/PlusCal package, enable such framework. Composition requirements, services, and interactions were described, and based on that, well-being in a smart building was studied as an example. Results of model checking were generated and an analysis of the state space was performed to understand non-trustworthy results through a deadlock invariant. The following objectives are targeted as future work: strengthening the well-being model

as well as tackling the composition concerns in other domains of interest namely: preventing oxygen toxicity in an autonomous ventilator and preventing collisions during emergency braking in the case of autonomous vehicles. Also, as capabilities composition concerns can be mutually exclusive (simplicity vs performance), studying this challenge in-depth is a target milestone.

NIST Disclaimer

Certain commercial equipment, instruments, or materials (or suppliers, or software, ...) are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose. Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

References

[1] C. Greer, M. J. Burns, D. A. Wollman, and E. R. Griffor, "Cyber-physical systems and internet of things. (no. special publication (nist sp)-1900-202)," tech. rep., 2019.

[2] Cisco, "Cisco iot prediction for 2030. a technical report." https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf.

[3] S. Yun, H. Kim, H. Shin, H. S. Chin, and W.-T. Kim, "A novel reference model for cloud manufacturing cps platform based on onem2m standard," *KIPS Transactions on Computer and Communication Systems*, vol. 8, no. 2, pp. 41–56, 2019.

[4] E. R. Griffor, C. Greer, D. A. Wollman, and M. J. Burns, "Framework for cyber-physical systems: Volume 1, overview," 2017.

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

[6] 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.

[7] oneM2M Partners, "OneM2M Technical Report : Study of Abstraction and Semantics Enablements." https://onem2m.org/images/files/deliverables/Release2/TR-0018-Industrial_Domain_Enablement-V2_0_0.pdf, 2016.

[8] A. Preventis, K. Stravoskoufos, S. Sotiriadis, and E. G. Petrakis, "Iot-a and fiware: Bridging the barriers between the cloud and iot systems design and implementation.," in *CLOSER (2)*, pp. 146–153, 2016.

[9] A. Bassi, M. Bauer, M. Fiedler, R. van Kranenburg, S. Lange, S. Meissner, and T. Kramp, *Enabling things to talk : Designing IoT solutions with the IoT Architectural Reference Model*. Springer Nature, 2013.

[10] M. Brumbulli, E. Gaudin, and C. Teodorov, "Automatic verification of bpmn models," in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020.

[11] W. Li, G. Privat, J. M. Cantera, M. Bauer, and F. Le Gall, "Graph-based semantic evolution for context information management platforms," in *2018 Global Internet of Things Summit (GIoTS)*, pp. 1–6, IEEE, 2018.

[12] I. Boneva, J. E. L. Gayo, and E. G. Prud'hommeaux, "Semantics and validation of shapes schemas for rdf," in *International Semantic Web Conference*, pp. 104–120, Springer, 2017.

[13] A. Kazmi, M. Serrano, and J. Soldatos, "Vital-os: An open source iot operating system for smart cities," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 71–77, 2018.

[14] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "Fogflow: Easy programming of iot services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2017.

[15] 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.

[16] L. Lamport, "The pluscal algorithm language," in *International Colloquium on Theoretical Aspects of Computing*, pp. 36–60, Springer, 2009.

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

[18] A. Kurniawan, *Learning AWS IoT: Effectively manage connected devices on the AWS cloud using services such as AWS Greengrass, AWS button, predictive analytics and machine learning*. Packt Publishing Ltd, 2018.

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

[20] V. Nundloll, Y. Elkhatib, A. Elhabbash, and G. S. Blair, "An ontological framework for opportunistic composition of iot systems," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 614–621, IEEE, 2020.

[21] F. Sivrikaya, N. Ben-Sassi, X.-T. Dang, O. C. Görür, and C. Kuster, "Internet of smart city objects: A distributed framework for service discovery and composition," *IEEE Access*, vol. 7, pp. 14434–14454, 2019.

[22] D. L. McGuinness, F. Van Harmelen, *et al.*, "Owl web ontology language overview," *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[23] G. Simko, D. Lindecker, T. Levendovszky, S. Neema, and J. Sztipanovits, "Specification of cyber-physical components with formal semantics–integration and composition," in *International Conference on Model Driven Engineering Languages and Systems*, pp. 471–487, Springer, 2013.

[24] Y. Li, S. Zhao, H. Diao, and H. Chen, "A formal validation method for trustworthy services composition," in *2016 International Conference on Networking and Network Applications (NaNA)*, pp. 433–437, IEEE, 2016.

[25] C. Laneve and L. Padovani, "An algebraic theory for web service contracts," *Formal Aspects of Computing*, vol. 27, no. 4, pp. 613–640, 2015.

[26] N. Le Sommer, Y. Mahéo, and F. Baklouti, "Multi-strategy dynamic service composition in opportunistic networks," *Information*, vol. 11, no. 4, p. 180, 2020.

[27] H. Groefsema, N. R. van Beest, and M. Aiello, "A formal model for compliance verification of service compositions," *IEEE Transactions on Services Computing*, vol. 11, no. 3, pp. 466–479, 2016.

[28] M. Sozeau, S. Boulier, Y. Forster, N. Tabareau, and T. Winterhalter, "Coq coq correct! verification of type checking and erasure for coq, in coq," *Proceedings of the ACM on Programming Languages*, vol. 4, no. POPL, pp. 1–28, 2019.

[29] T. Ali, M. Nauman, and M. Alam, "An accessible formal specification of the uml and ocl meta-model in isabelle/hol," in *2007 IEEE International Multitopic Conference*, pp. 1–6, IEEE, 2007.

[30] 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.

[31] K. HALBA, "Iotcap : A platform based on the vert.x toolkit and iccf foundations." https://github.com/usnistgov/ICCF.

[32] K. Halba, E. Griffor, P. Kamongi, and T. Roth, "Using statistical methods and co-simulation to evaluate ads-equipped vehicle trustworthiness," in *2019 Electric Vehicles International Conference (EV)*, pp. 1–5, IEEE, 2019.

[33] M. Burns, T. Roth, E. Griffor, P. Boynton, J. Sztipanovits, and H. Neema, "Universal cps environment for federation (ucef)," in *2018 Winter Simulation Innovation Workshop*, 2018.