

A Match Made in Semantics: Physics-infused Digital Twins for Smart Building Automation

Ganesh Ramanathan

Siemens AG

Zug, Switzerland

ganesh.ramanathan@siemens.com

Simon Mayer

University of St. Gallen

St.Gallen, Switzerland

simon.mayer@unisg.ch

Abstract—Buildings contain electro-mechanical systems that ensure the occupants' comfort, health, and safety. The functioning of these systems is automated through control programs, which are often available as reusable artifacts in a software library. However, matching these reusable control programs to the installed technical systems requires manual effort and adds engineering cost. In this article, we show that such matching can be accomplished fully automatically through logical rules and based on the creation of semantic relationships between descriptions of *physical processes* and descriptions of technical systems and control programs. For this purpose, we propose a high-level bridging ontology that enables the desired rule-based matching and equips digital twins of the technical systems with the required knowledge about the underlying physical processes in a self-contained manner. We evaluated our approach in a real-life building automation project with a total of 34 deployed air handling units. Our data show that rules based on our bridging ontology enabled the system to infer the suitable choice of control programs automatically in more than 90% of the cases while avoiding almost an hour of manual work for each such match.

I. INTRODUCTION

Buildings contain systems (e.g., for heating, ventilation, electrical energy management, plumbing, etc.) that comprise sub-systems and components (e.g., boilers, pumps, fans, etc.), each of which manages physical mechanisms that are involved in the transformation of substance or energy (i.e., physical *stuff* [3]). Domain experts have developed libraries of reusable control programs for automating many types of systems commonly found in buildings. In many cases, this avoids the automation engineer having to develop (and test) control strategies from scratch. For example, a control program for a room's heating system can be developed once and reused across multiple installations because the system's technical components and topology are similar and the underlying physical processes are the same [14].

However, the automation engineer today still has to manually read and understand the descriptions of the control programs available in the library and also the design documentation of the technical system in order to select a suitable program to reuse—i.e., they need to match specific system designs with control programs. An *automated* way to select the correct control program and adapt its configuration to a technical system would reduce the cost and effort required for such re-engineering [4], [16].

Researchers in both process engineering and automation have developed several ontologies to create machine-understandable knowledge about technical systems, control programs, and physical processes. Several authors have hinted that an integration of these might enable automated matching of a technical system to a suitable control program [5], [14], [15], [17].

However, the concepts that are required for interlinking the individual abstractions have not been fully explored and the engineering ontologies currently available in the building automation (BA) domain do not address this need. On the other hand, establishing a new set of ontologies for this purpose is difficult to achieve and cumbersome to maintain considering the multi-vendor and fragmented landscape in automation systems, including in BA [4], [19]. This setup with multiple heterogeneous vendors also poses the challenge of achieving a centralized knowledge base since it requires coordination and collaboration for data ownership and governance.

In this paper, we argue and demonstrate that both these challenges can be conceptually overcome through a *high-level ontology* that coordinates (but does not replace) vendor ontologies. Our approach is applicable in the context of BA as well as in other domains with similar challenges. By providing bridging concepts, the high-level ontology enables automated matching and, by its ability to integrate knowledge, also enables the creation of self-contained descriptions of technical systems and control programs, which can be embedded into digital twins, thereby achieving a *decentralized knowledge repository*. Our core contributions are:

- 1) We report on the results of our study of the existing ways control programs, technical systems, and physical processes are described and further explain that the knowledge of the underlying physical processes is essential for reasoning about the suitability of a control program to a technical system towards automatic matching.
- 2) We propose a high-level ontology that can bridge concepts in the existing ontologies for modeling technical systems and physical processes. This helps designers of both technical systems and control programs to include the knowledge of the physical process in their (machine-understandable) design descriptions, which enables creation of generic semantic rules for match-

ing.

- 3) We introduce the World Wide Web Consortium's Web of Things *Thing Description* standard as a way to embed machine-understandable descriptions of the physical processes into a standardized and interoperable representation of the *digital twin* of the technical system. Such *physics-infused digital twins* enable the rule-based matching to operate on knowledge that is accessible in a distributed and self-contained manner.
- 4) We show the results of evaluating our approach in a real-life BA system modeled using currently available engineering ontologies. Our results confirm that our approach permits the automatic matching of technical systems and control programs. Furthermore, we discuss which aspects of our bridging concepts were particularly relevant for automated matching.

II. RELATED WORK

This section introduces existing work that is relevant to integrating knowledge towards enabling automated selection or configuration of control programs, and highlights the gaps which we addressed in our approach.

A. Technical Systems

Design descriptions of technical systems (TS) are traditionally created in human-readable form only (e.g., CAD drawings, texts, etc.). In recent years, there has been an increased use of Semantic Web technologies, which allows experts to create reusable ontologies that capture *machine-understandable* knowledge about their specific domains. Over the past decade, the Semantic Web has propelled the creation of ontologies in the building systems domain too [2], [12]. For example, ontologies such as BRICK [1], Haystack¹, RealEstateCore², SAREF4BLDG³ (for building systems), SSN/SOSA⁴ (for sensors and actuators), and Tubes [11] (system design) can be used to describe the sub-systems, components, and their inter-dependencies. The principal language used for this purpose is the Resource Description Framework (RDF) and its extension Web Ontology Language (OWL)⁵ which is based on a decidable subset of first-order logic and can hence be used for automated reasoning. A similar approach for model-based engineering, though not based on the Semantic Web, is offered by the AutomationML standard⁶ which is widely used in the factory and process automation domains; [6] proposes an OWL ontology for this standard.

Such machine-understandable system descriptions can be used during both design [4], [16] and operation [5], [13] of an automation system. Further, [14], [17] show that structural descriptions can be extended with knowledge about the physical interfaces of components and their relationship with the underlying physical processes.

¹<https://project-haystack.org/doc/lib-phIoT>

²<https://www.realestatecore.io/>

³<https://saref.etsi.org/saref4bldg/v1.1.2/>

⁴<https://www.w3.org/TR/vocab-ssn/>

⁵<https://www.w3.org/OWL/>

⁶<https://www.automationml.org/>

B. Control Programs

Several standards like IEC 61131 and its implementation in PLCOpen⁷, CIF⁸, Control Description Language (CDL)⁹ etc. exist to support the reusability of control programs (CPs) for common types of systems. On this basis, ontologies have been proposed to permit machines to understand the purpose of a CP and the use of its external interfaces—prominent amongst these are CTRLOnt [17] and a semantic extension to CDL described in [20].

However, current approaches to semantically describing CPs remain limited to annotating their IO interfaces with information about the physical quantity, measurement units, and type of *stuff* they deal with. We propose to in addition formalize the relationship of CPs with TSs and their underlying physical processes. To this end, we suggest a high-level ontology which allows the designer of a CP to relate its external interfaces to both: interfaces of the TS and variables of the underlying physical process.

C. Physical Processes and Mechanisms

Physical mechanisms like energy conversion, heat exchange, or fluid flow are modeled using mathematical equations which are based on laws in physics that describe the static and dynamic behavior. Such mechanisms can be coupled to each other through the flow of substance or energy (often referred to as physical *stuff*) to form a physical process. The process engineering community has developed several ontologies to semantically describe physical processes in terms of connected physical mechanisms and the variables which represents its state. We will use the term *physical process description* (PPD) to denote this knowledge. This knowledge is essential because control programs are often designed taking into consideration the type and composition of the physical mechanisms involved in a process [7], [9], [23]. Prominent amongst the ontologies to model PPDs are OntoCAPE (in OWL) [9] and PhySys [3]. Ontologies like Qudt¹⁰ and Scientific Variables Ontology¹¹ provide an extensive vocabulary of physical quantity kinds, constants, and units. In [14], an ontology called Elementary provides a high-level abstraction of the physical process, that can be linked to components in a technical system.

D. Web of Things and its Relevance to Automation Systems

Automation systems are increasingly adopting architectures and technologies behind the Internet of Things (IoT), which is also one of the key drivers behind the idea of creating *digital twins*. However the general lack of both technical and semantic interoperability in the practical implementations of IoT [21] led the standardization body W3C to put forward a recommendation under the title *Web of*

⁷<https://plcopen.org/>

⁸<https://eclipse.dev/escet/v2.0/cif/>

⁹<https://obc.lbl.gov/specification/cdl.html>

¹⁰<https://www.qudt.org/>

¹¹<https://scientificvariablesontology.org/>

Things¹². The recommendation includes a schema to *semantically* describe *things* and the interfaces they provide – this is called the *Thing Description* (TD)¹³. Listing 1 shows an illustrative example of a digital twin of a boiler modeled as a TD – later in section III-C, we show how knowledge about the physical process this boiler is involved in can be included *in-band* in this TD, and how this enables reasoning about the physical consequences of using the boiler’s controls. In the context of automation systems, a TD, therefore, fits well as a part of the digital twin of a thing [8] because it provides the opportunity to integrate machine-understandable knowledge of its design as well as that of the underlying physical processes the thing is involved in.

Listing 1. An example Thing Description of a boiler

```
{ "@context": "<references to ontologies>",
  "title": "Boiler for heating system",
  "@id": "urn:boiler-01",
  "base": "http://blr-01",
  "@type": "brick:Boiler",
  "properties": { "water-out-temp": {
    "title": "Outlet water temperature",
    "@type": "brick:WaterTemperature",
    "readOnly": true, "type": "number",
    "forms": [{"href": "/sensors/twout"}]
  },
  "actions": { "fuel-valve-actuation": {
    "title": "Change fuel flow to burner",
    "forms": [{"href": "/actuators/fvalve"}]
  },
  "events": { "high-temperature-alarm": {
    "title": "High temperature alarm",
    "forms": [{"href": "/alarms/htemp"}]
  }
}
```

III. APPROACH

To determine the relevant aspects for the matching of TS to CP, we examined how these are currently described in practice and how this knowledge serves the manual process of matching. We first summarize our findings and point out that a lack of means to integrate knowledge about physical processes in the machine-understandable versions of design descriptions hinders their automated matching (see Section III-A).

The existing fragmentation in machine-understandable knowledge served as a motivation towards building a high-level integrating ontology, which we describe in detail in Section III-B. Our core contribution here is that we designed an ontology that is based on knowledge from established engineering practices, which allows the workflow for matching to be replicated in an automated manner while reusing descriptions based on available ontologies.

Finally, in Sections III-C and III-D, we show how the concepts in this ontology can be used to embed the required knowledge directly into the TDs and, further, to create the semantic rules to match them to the CPs. We make our ontology, its bridging to existing ontologies, and all examples openly available to the interested readers¹⁴.

A. The Need For Knowledge About Physical Processes

The design of a TS described in the form of components and their topological relationships (e.g., *part-of* or *connects-to*) is broadly termed as the *system design description* (SDD) [10], [16]. Knowledge about the physical processes and the mechanisms that the system needs to manage is only selectively described in SDDs because the designer assumes that a human expert reading the SDD will be able to infer them. Semantic Web ontologies for SDDs also reflect this by providing only taxonomies and relationships to describe components and their topology and do not exploit the opportunity to integrate semantic model of physical processes [11], [22].

Many commonly encountered system types (e.g., boilers) have similar technical features and are based on the same or similar physical principles. For such cases, designers devise reusable CPs that are based on a *generalization* of the system’s design: the *abstract system design* (ASD). This is the CP designer’s conception of the TS. The relation between the interfaces of a CP, i.e., its IOs, and the sensors and actuators in the ASD are described in text or diagrams. Finally, in the documentation of the CP, the goals of the program and the expected roles of the sensors and actuator in the physical process are also described. We use the term *control program description* (CPD) to collectively refer to such explanation (which is currently only in human-readable form).

Achieving a machine-understandable version of the CPD (to enable automated matching) faces three major gaps. Firstly, though ASD is a vital part of a CPD, current approaches to semantically describe CPs do not include it as a *first-class abstraction*. Instead, they only have a provision to annotate the IOs with selective information regarding the physical variables. We argue that ASDs, because they are in effect *stereotypical* systems, can be semantically described using the same ontologies as for SDDs. We will show later in this section that a first-class abstraction is indeed required for formulation of generic semantic rules for matching.

Secondly, like SDDs, ASDs need to be linked to the underlying physical process model because this knowledge is required when describing a CP.

Finally, given the possibility of semantically modeling the ASD, concepts to integrate it into the description of the CP are currently missing.

We show that these gaps can be addressed by developing bridging concepts that can interlink knowledge modeled using current ontologies for CPs, SDDs, and physical processes.

In order to determine the particular bridging concepts that are essential for matching, we analyzed descriptions of several existing CPDs—namely, those in the ASHRAE Guideline 36 library¹⁵ for HVAC functions, boiler combustion control programs from four different manufacturers (described in texts and SAMA diagrams), and a CP library

¹²<https://www.w3.org/WoT/>

¹³<https://www.w3.org/TR/wot-thing-description/>

¹⁴See <http://w3id.org/phydit/doc>

¹⁵<https://ashrae.org>, machine-readable models are here <https://simulationresearch.lbl.gov>

of a large BA manufacturer (described in texts and P&I diagrams). In total, 87 different CPDs in the BA domain for applications ranging from air handling units, variable air volume units, ventilation units, chilled water plants, to boiler plants were analyzed. We also examined exemplary SDDs that were manually matched to the CPD, noting the conceptual aspects that contributed to a successful match.

Our finding from this study was that amongst the CPDs, some could be matched to SDDs *solely* based on taxonomic and topological features —for example, in the case of a radiator-based heating CP, it is sufficient to know that the TS should consist of a *radiator* that has a control valve and a temperature sensor. However, in a large majority (78 of the 87) of the cases, the *description about the physical process* that a CP was conceived for and the *contextualization of the inputs, outputs, or parameters* in the process played an *essential* role in matching it to an SDD. This demonstrates that the taxonomic and topological knowledge captured by current ontologies for describing TS in SDD and ASD of a CP is not sufficient for the purpose of automated matching.

We could classify the aspects of a CPD that are relevant when reasoning about its match to a TS as 1. *general specification* which describes compatibility of the CP to type of technical system (and in several cases the underlying physical process), 2. *IO specification* that describes the sensors and actuators that their IOs are expected to be connected to, and 3. *specification of design-related parameters* that the CP requires. Table I lists these aspects of the CPDs along with examples.

Our study confirms our previous observation that machine-understandable semantic description of CPDs require: 1. an explicit model of the ASD to be a part of the CPD as a first-class abstraction, and, 2. the description of the technical system, both in the form of SDD and ASD, to be linked to description of the physical process (PPD).

We propose to achieve the above through a high-level bridging ontology, which we call PhyDiT (standing for **Physics-infused Digital Twins**) — we will refer to concepts in this ontology using the namespace abbreviation of pdt.

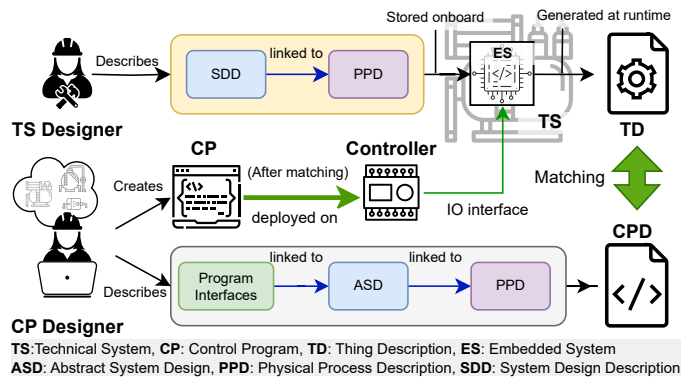


Fig. 1. Overview of how the integrated knowledge in the two design workflows are used for automated matching. The blue arrows indicate the conceptual bridging that is accomplished by PhyDiT

Figure 1 illustrates how our proposed approach integrates

knowledge in the two design abstractions.

B. The PhyDiT Ontology

1) *The Required High-level Abstractions:* As high-level abstractions, PhyDiT models three aspects (see Figure 2: 1. The control program and its external interfaces, 2. the technical system in the form of components and its external control interfaces (sensors and actuators), and 3. the physical process in the form of mechanisms and the process variables which represent their state.

We examined whether the classes and relationships within each aspect corresponded well to those in existing concrete ontologies — this was done to ensure that PhyDiT can be easily bridged to them. The classes and relationships involved in describing control programs have direct equivalences in CTRLOnt and CDL. Similarly, the model for physical processes corresponds directly to concepts in OntoCAPE, Elementary, Tubes, and PhySys.

There is a much larger variety of ontologies for describing technical systems. We studied the ontologies that apply to the building systems domain and found that the classes and relationships modeled in PhyDiT can largely be bridged to existing system design ontologies (see Table II).

We now describe the concepts provided by PhyDiT to semantically inter-link the three individual aspects.

2) *Linking System Design Description to Physical Process Model:* We introduce the relationship manages to link the structural aspects of the technical system and the physical process models. A System is linked to a PhysicalProcess through the relationship manages. Similarly, a Component, which is a part of a System manages a PhysicalMechanism that is a part of the PhysicalProcess. For example, this can be used to state that a boiler *manages* heat generation process, while its component, the burner, *manages* the mechanism of fuel combustion.

Further, an Actuator can be linked to a ProcessVariable through two possible relationships: 1. *manipulates*, which signifies the immediate independent variable that the actuation will influence, and 2. *affects*, which goes through the chain of PhysicalMechanism(s) to list the dependent variables that will be affected. Similarly, a Sensor can be linked to a ProcessVariable through the relationship *observes*. By coupling these relationships to the structural relationships, the *system context* of a sensor or actuator can be inferred in terms of Component and its Port where it can be observed, manipulated, or affected.

3) *Linking Descriptions of Control Program and System Design:* In the general description of a CPD the kind of technical system and, and optionally, also the kind of underlying physical process for which it is suitable is stated. In our approach, this specification can be achieved by linking the instance of ControlProgram to the instance of System (that is part of the ASD) using the property *intendedFor*. Consequently, the PhysicalProcess and its PhysicalMechanisms can be inferred.

TABLE I

SUMMARY OF HOW CPs ARE DESCRIBED IN TERMS OF ITS COMPATIBILITY TO TECHNICAL SYSTEMS AND PHYSICAL PROCESSES

CP Aspect	Technical system-related aspect	Physical process-related aspect	Example specification
General	Suitable technical system type(s)	Physical process and mechanisms	Boiler with heat generation through combustion
Inputs	Sensor type and system context	Observed variable (quantity kind, etc.)	Water pressure (in kPa) sensor at pump inlet
Outputs	Actuator type and system context	Manipulated and affected variables	Fan motor drive (to influence air flow in combustion)
Parameters	Configuration and sizing	Limits for variable values	Boiler capacity, minimum water flow rate

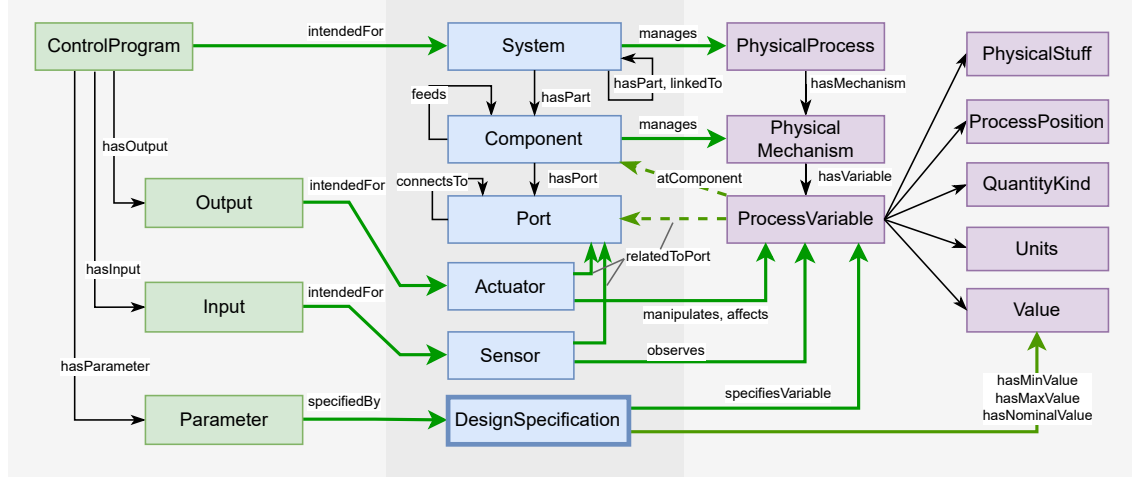


Fig. 2. Overview of the PhyDiT ontology. The bold lines and borders indicate the bridging concepts we have introduced. The dotted lines indicate relationships that are inferred (e.g., `atComponent` is inferred if the variable is associated to mechanism that is managed by the component.)

TABLE II

MAPPING OF CLASSES IN PHYDIT TO EXISTING SDD ONTOLOGIES

(*=PORT IS INFERRED **=NO DISTINCT CLASS TO MODEL COMPONENTS)

PhyDiT Class	BRICK	ASHRAE 223p	Haystack	RealestateCore	SAREF4BLDG	SOSA-SSN
System	System	System	system	Equipment	Device	System
Component	Equipment	Equipment	equip	Component	Device**	System**
Port	feeds*	ConnectionPoint	-	-	FlowTerminal	-
Sensor	Sensor	Sensor	input	Sensor	Sensor	Sensor
Actuator	Actuator	Actuator	output	Actuator	Actuator	Actuator
DesignSpecification	-	-	-	Document	Via data properties	SystemProperty

Similarly, by linking sensors and actuators to process variables in the previous step, we have in effect coupled the description of the external interface of the system to the underlying PPD. Therefore, now by (solely) relating (through `intendedFor`) the IOs to sensors and actuators in the ASD, the specification of both TS- and physical process-related aspects of the IOs listed in Table I are achieved.

Finally, a CP may also require operational parameters that must to be obtained from the design specification of the TS. For example, the combustion control program may require to know the minimum air flow rate to maintain while operating the blower. To enable the TS designer to specify such parameters and its values, PhyDiT provides the class `DesignSpecification`. In the SDD, an instance of this class can be linked to a `ProcessVariable` through relation `specifiedVariable`, and the value can be stated using the data properties `hasMinimumValue`, `hasMaximumValue` and `hasNominalValue`. From the CPD's perspective, the need for a design specification to parameterize the CP is expressed by a similarly formulated

`DesignSpecification` (however without value specification), which is linked to a `Parameter`.

C. Synthesizing TDs

In section II-D, we highlighted that the computational and network communication capability provided by the ES of a TS is an opportunity to integrate the SDD directly into the TS in form of WoT TD. We developed a scheme (shown in Table III) that identifies the required knowledge in the SDD and the abstractions of a TD to which it can be mapped. In other words, we are infusing the TD with knowledge that is relevant for matching *it* (i.e., the TS) to one or more suitable CPD. In effect, the TD becomes the *physics-infused digital twin* of the TS.

Listing 2 shows an example snippet of a TD¹⁶ that has been enhanced by knowledge from SDD. The TD also provides links to the source models. The `systemModel` provides the URL of the SDD itself (which could be, for example, stored on-board the ES), while `simulationModel`

¹⁶The complete example TD is available in the supplementary material

TABLE III
MAPPING OF CONCEPTS TO GENERATE THE TD FROM SDD.

TD abstraction	Metadata	Mapping to SDD and its semantics	Example
Thing	rdf:type	Domain class corresponding to the pdt:System instance	brick:Boiler
Thing	pdt:manages	pdt:PhysicalProcess that the TS manages	hvac:EnergyConversion
PropertyAffordance		Corresponds to an instance of pdt:Sensor in the SDD	:temperature-sensor
PropertyAffordance	pdt:sensor	Domain class corresponding to pdt:Sensor instance	brick:TemperatureSensor
PropertyAffordance	pdt:component	The pdt:Component which related to the observed variable	brick:Burner
PropertyAffordance	pdt:observes	The pdt:ProcessVariable that the sensor observes	:water-inlet-temperature
ActionAffordance		Corresponds to an instance of pdt:Actuator in the SDD	:fuel-valve
ActionAffordance	pdt:actuator	Domain class corresponding to pdt:Actuator	brick:ValveActuator
ActionAffordance	pdt:manipulates	The pdt:ProcessVariable that is directly affected	:fuel-flowrate
ActionAffordance	pdt:affects	The pdt:ProcessVariable that is consequently affected	:water-outlet-temperature
ActionAffordance	pdt:mechanism	The mechanism that the manipulated variable is associated to	hvac:Combustion
ActionAffordance	pdt:component	The pdt:Component that manages the above mechanism	brick:Burner

can be linked to a resource server from where the FMU of the TS can be downloaded.

Listing 2. The TD of a boiler augmented with knowledge of physical process

```
{...
  "@type": "brick:Boiler",
  "manages": {"@type": "hvac:EnergyConversion"},
  "systemModel": "/model/tsd-blr-01.rdf",
  "simulationModel": "http://server/boiler.fmu",
  "specification": { "min-water-flowrate":
    { "specifiedVariable": { "unit": "qudt:L-PER-MIN",
      "quantityKind": "qudt:VolumeFlowRate",
      "stuff": "brick:Water", "hasMinValue": 50 } },
  "actions": {
    "fuel-valve-actuation": {
      "actuator": {"@type": "brick:Valve"},
      "relatedTo": {"@type": "hvac:Combustion"},
      "manipulates":
        { "atComponent": {"@type": "hvac:Burner"},
          "stuff": "brick:Fuel",
          "position": "elem:inlet",
          "quantityKind": "qudt:VolumeFlowRate" },
      "affects":
        { "atComponent": {"@type": "hvac:BoilerTube"},
          "stuff": "brick:Water", ... }
    },
  "properties": {
    "water-outlet-temp": {
      "sensor": {"@type": "brick:TemperatureSensor"},
      "observes":
        { "atComponent": {"@type": "hvac:BoilerTube"},
          "stuff": "brick:Water", ... }
    }
  }
}
```

D. Matching TDs to CPDs

The mappings that we described in the preceding subsections allow us to now create formal rules for matching a TD ($\mathcal{T}$) to a CPD, i.e. an instance of ControlProgram ($\mathcal{C}$). We define $\tau_x = C(x)$ as a function that returns the OWL class of an RDF individual x . The predicate $subClassOf(\tau_{sub}, \tau_{super})$ is true if $\tau_{sub} \sqsubseteq \tau_{super}$.

For mapping $\mathcal{T}$ with $\mathcal{C}$, we define three formal rules to validate (i) Type Compatibility, (ii) IO Compatibility, and (iii) Parameter Availability. We wish to emphasize that the rules proposed below are meant to illustrate the conceptual benefit of our approach — the exact formulation may be adapted to suit the application- or domain-specific needs.

Type Compatibility A CP can be specified as being suitable for one or more types of TS (e.g., a temperature

control program can be stated to be suitable for both heating and cooling systems). We also found from our study that a CP that has been developed for a more specific TS type can also be used with the general type. Therefore, the following rule checks if a TS type conceived by the CP designer (in the ASD) is the same or sub-type of the TS type $\tau_t = C(\mathcal{T})$ in the SDD.

$$\exists s_c (System(s_c) \wedge intendedFor(\mathcal{C}, s_c) \wedge subClassOf(C(s_c), \tau_t)) \rightarrow isSystemCompatible(\mathcal{C}, \mathcal{T})$$

Similarly, to match that the type of physical process(es) conceived for the CP is the same or sub-type of that the TS instance manages:

$$\begin{aligned} \exists p_c, s_c ((System(s_c) \wedge intendedFor(\mathcal{C}, s_c) \wedge manages(s_c, p_c)) \\ \rightarrow \exists p_t (manages(\mathcal{T}, p_t) \wedge subClassOf(C(p_c), C(p_t))) \\ \rightarrow isProcessCompatible(\mathcal{T}, \mathcal{C}) \end{aligned}$$

IO Compatibility In principle, here, descriptions of the CPs' IOs are matched to the TD's interaction affordances. For every Input interface of a CP, a match is sought to a TD PropertyAffordance, and similarly, Output are matched to ActionAffordance. The inference rules check if the sensor or actuator type related to the TD's interaction affordance is same or sub-class of the sensor or actuator in the CPD, and if the corresponding variables that is stated as being observed, manipulated, or affected are *equivalent*. We define a function *equivalent*(x, y), which returns true if the variables x and y have the same stuff, quantity, quantity kind, and process position. We do not compare the units; we use the quantity kind to evaluate the semantic compatibility. This function can be extended to also compare the component or the physical mechanism to which the variable is related.

$$\begin{aligned} \forall i (Input(i) \wedge hasInput(\mathcal{C}, i) \wedge intendedFor(i, s) \\ \wedge observes(s, v_c) \rightarrow \exists a (hasPropertyAffordance(\mathcal{T}, a) \\ \wedge observes(a, v_t) \wedge sensor(a, s_t) \wedge equivalent(v_c, v_t))) \\ \wedge subClassOf(C(s), C(s_t)) \rightarrow isInputCompatible(\mathcal{C}, \mathcal{T}) \end{aligned}$$

$$\begin{aligned} \forall o (Output(o) \wedge hasOutput(\mathcal{C}, o) \wedge intendedFor(o, r) \\ \wedge manipulates(r, v_{mc}) \wedge affects(r, v_{ac}) \rightarrow \\ \exists a (hasActionAffordance(\mathcal{T}, a) \wedge manipulates(a, v_{mt}) \\ \wedge affects(a, v_{at}) \wedge subClassOf(C(r), C(a)) \\ \wedge equivalent(v_{mc}, v_{mt}) \wedge equivalent(v_{ac}, v_{at}))) \\ \rightarrow isOutputCompatible(\mathcal{C}, \mathcal{T}) \end{aligned}$$

Further, similar to the above rule, we define $hasActionCoverage(\mathcal{T}, \mathcal{C})$ to be true if each action affordance of the $\mathcal{T}$ has a compatible output in $\mathcal{C}$. This rule will ensure that we do not match a CP only based on a subset of the TS actuation interfaces.

Parameter Availability To check if a `DesignSpecification` required by a CP is available in the TD, we compare the equivalence of the variables each specifies:

$$\begin{aligned} & \forall p (hasParameter(\mathcal{C}, p) \wedge specifiedBy(p, d_c) \\ & \wedge specifiesVariable(d_c, v_c) \rightarrow \exists d_t (specification(\mathcal{T}, d_t) \\ & \wedge specifiesVariable(d_t, v_t) \wedge equivalent(v_c, v_t)) \\ & \rightarrow isSpecCompatible(\mathcal{T}, \mathcal{C}) \end{aligned}$$

Given a specific $\mathcal{T}$ and a specific $\mathcal{C}$, compatibility is inferred through the conjunction of the above rules.

IV. EVALUATION

Our evaluation aimed to verify if SDDs and CPDs, each originating from independent designers (and created using different ontologies), could be matched based on the high-level concepts provided by PhyDiT. Additionally, instead of having a centralized knowledge graph or even using the complete SDD of a TS, we wished to infuse the required knowledge in the form of metadata in the TD so that it is self-contained. Therefore, we want to confirm if the approach of embedding the selective parts of the SDD into the TD is sufficient for matching to CPDs in real-life applications.

To evaluate our approach, we chose a real-life setup of office buildings where thirty-four air-handling units (AHUs) were installed and commissioned. No two AHUs were exactly the same — they differed in their topology and the type of components involved. We created the OWL version of the SDDs for the thirty-four AHUs using the Haystack ontology, while using Tubes to describe the underlying physical process.

The AHUs were already commissioned using one of eight choices of CPs meant for AHUs available in the BA manufacturer’s CP library. This means that even if two AHUs differed in terms of component types and topology, the same CP could fulfill the automation requirement. From the commissioning documents, we knew the engineers’ choice of CP for each AHU. From the project logs, we estimated that, on average, it takes an hour (varying between 20 and 90 minutes) to read and understand an SDD and search for a suitable CP in the library.

The eight CPs were each described using text and diagrams. In principle, they contained the ASD of the AHU that the CP was suitable for along with the specifications of the sensors and actuators required to be connected to the programs’ IO. Based on such descriptions, we created corresponding CPDs in OWL using the BRICK and Elementary ontology. Additionally, we also created CPDs for applications that are functionally close to AHUs — namely, for ventilation, fan-coil, and variable air volume units. This is because even though two CPs may match from physical mechanism and IO perspective, the sequence of operation

and timing of control actions often differ based on the type of the technical system.

We chose the above-described setup because it presented the essential characteristics to generalize our evaluations — i.e., a heterogeneous system described using different ontologies.

The thirty-four AHU instances were each equipped with an ES in which a Java-based program was deployed that could synthesize a TD from SDD according to the mappings identified in Table III. This program can work with any SDD based on an ontology that is bridged to PhyDiT. Through the ES’s network interface, an AHU’s TD could be obtained at run-time.

We developed another Java-based program, which, given the network address of a TS, retrieves its TD and then uses SPARQL queries corresponding to the rules listed in section III-D to find the CPD(s) that matches the TD.

Results We compared the matches suggested by our program with the choice of CPs that were deployed (and tested) at the installation.

For our 34 AHUs, 31 (>90%) were correctly matched; one was matched to two CPs, and two were not matched to any CP.

For one of the AHUs, two CPs were (correctly) suggested as a match, though only one was suitable from the process dynamics perspective (in this case, a cascade control loop was required for temperature control). For the two remaining AHUs, our approach did not match any CP because the physical context of an actuator did not conform to any description in the CP library. In this case, the SDDs had a valve in the outlet of the heat exchanger instead of in the inlet, which is what the CPDs in our library specified.

To address multiple matches, we intend to develop a method to specify and match requirements regarding process dynamics, and further, a way to employ simulation models (which can be linked to the TDs) for verifying the chosen CP.

To address the case of the AHU, where no match was found, we intend to develop a method to specify design options/variants in the ASD.

We furthermore evaluated the relevance of the different aspects (e.g., system type, sensor type, etc.) of the TD metadata for matching performance. This was done by selectively removing aspects and verifying whether the implementation still produced a correct match. Across all 32 successful matches, Table IV shows the relevance of each aspect that we embedded in the TDs of the AHUs towards enabling successful matching. We grouped the SDDs according to the set of aspects that were required to correctly match them. Table IV shows these groups, the number of AHUs in each, and the metadata that was necessary: In group A, only knowledge about the system type, sensor type, and actuator type is required for correctly matching—illustrating that for these matches, no knowledge about the “elaborate” physics is required. On the other hand, without knowledge about the observed, manipulated, and affected physical process

TABLE IV

THE ROLE OF TD METADATA IN ENABLING A SUCCESSFUL MATCH. THE GROUPING (A-D, WITH THE NUMBER OF SDDS SHOWN IN BRACKETS) SHOWS HOW THEY OCCUR TOGETHER.

TD metadata and the percentage of matches where they were necessary	Grouping according to the metadata necessary (marked with •)			
	A (4)	B (8)	C (16)	D (4)
General: System type (100%)	•	•	•	•
General: Physical process type (88%)		•	•	•
Property: Sensor type (37%)	•	•		
Property: Observed variable (62%)			•	•
Property: Component and position (12%)				•
Action: Actuator type (100%)	•	•	•	•
Action: Manipulated variable (62%)			•	•
Action: Affected variable (62%)			•	•
Action: Component and position (62%)			•	•
General: Design parameters (12%)				•

variables, 20 AHUs (>62%) would have been matched incorrectly. In groups C and D, we see that describing the observed physical variable makes the sensor type information redundant (but not vice versa), which shows that semantics related to the physical process play a more fundamental role in describing the CP's IOs.

Through our evaluation, we also confirm that the knowledge required for matching could be embedded into the TDs, thereby achieving a decentralized (and dynamically adaptable) method of making system knowledge accessible.

Limitations Apart from the need to model the specification of process dynamics and design variants of TS that we recognized in our evaluation, we also recognize that our approach currently does not include the dimensions of system requirements and goals. Though the requirements and goals are often implicit in the inter-linked descriptions of the technical system and the physical processes that it manages, an explicit integration of semantic description of requirements (for e.g., through goal-oriented requirements engineering approach [18]) will enable formal verification that the matched CP fulfills the process goals. We are currently researching this possibility.

V. CONCLUSION

Our proposed approach of using the bridging concepts in the PhyDiT ontology to link ASD and SDD to PPD showed demonstrable benefit in enabling automated matching of a TS to a CP. The ontology also enabled the extraction and embedding of knowledge of physical mechanisms into the TDs, which serve as a distributed knowledge base.

The PhyDiT ontology could be easily bridged to existing engineering ontologies, so semantic rules and software program implementations based on it are not tightly coupled to a specific choice of ontologies. The high-level concepts also make the rules for TD metadata extraction and matching flexible for modifications and extensions to suit specific domains.

Our approach shows the concrete benefits of infusing knowledge of the underlying physical mechanisms into digital twins, which will open up further avenues to explore and achieve autonomous automation.

REFERENCES

- [1] Balaji, B., et al.: Brick: Towards a unified metadata schema for buildings. In: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments. pp. 41–50 (2016)
- [2] Biffl, S., Sabou, M.: Semantic web technologies for intelligent engineering applications. Springer (2016)
- [3] Borst, P., Akkermans, J., Pos, A., Top, J.: The physsys ontology for physical systems. In: Working Papers of the Ninth International Workshop on Qualitative Reasoning QR. vol. 95, pp. 11–21 (1995)
- [4] Butzin, B., et al.: A model based development approach for building automation systems. 19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA (2014). <https://doi.org/10.1109/ETFA.2014.7005365>
- [5] Delgoshaei, P., Heidarinejad, M., Austin, M.A.: A semantic approach for building system operations: Knowledge representation and reasoning. Sustainability **14**(10), 5810 (2022)
- [6] Kovalenko, O., et al.: Automationml ontology: modeling cyber-physical systems for industry 4.0. IOS Press Journal **1** (2018)
- [7] Lee, E.A.: Cyber physical systems: Design challenges. In: 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC) (2008). <https://doi.org/10.1109/ISORC.2008.25>
- [8] Mayer, S., et al.: An open semantic framework for the industrial internet of things. IEEE Intelligent Systems **32**(1), 96–101 (2017)
- [9] Morbach, J., Wiesner, A., Marquardt, W.: Ontocape—a (re) usable ontology for computer-aided process engineering. Computers & Chemical Engineering **33**(10), 1546–1556 (2009)
- [10] Nielsen, C.B., et al.: Systems of systems engineering: basic concepts, model-based techniques, and research directions. ACM Computing Surveys (CSUR) **48**(2), 1–41 (2015)
- [11] Pauen, N., Schlütter, D., Frisch, J., van Treeck, C.: Tubes system ontology: Digitalization of building service systems. In: Proceedings of the 9th Linked Data in Architecture and Construction Workshop (2021)
- [12] Pauwels, P., Zhang, S., Lee, Y.C.: Semantic web technologies in aec industry: A literature overview. Automation in construction **73**, 145–165 (2017)
- [13] Ramanathan, G., Husmann, M.: Semantic description of equipment and its controls in building automation systems. In: Proceedings of the Extended Semantic Web Conference (2022)
- [14] Ramanathan, G., Mayer, S.: Reasoning about physical processes in buildings through component stereotypes. In: Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation. pp. 120–129 (2023)
- [15] Runde, S., et al.: Engineering of building automation systems—state-of-the-art, deficits, approaches. In: IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010). pp. 1–8. IEEE (2010)
- [16] Runde, S., et al.: Integrated automated design approach for building automation systems. In: IEEE International Conference on Emerging Technologies and Factory Automation. pp. 1488–1495 (2008)
- [17] Schneider, G.F., Pauwels, P., Steiger, S.: Ontology-based modeling of control logic in building automation systems. IEEE Transactions on Industrial Informatics **13**(6), 3350–3360 (2017)
- [18] Van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In: Proceedings fifth ieee international symposium on requirements engineering. pp. 249–262. IEEE (2001)
- [19] Vogel-Heuser, B., et al.: Challenges for software engineering in automation. Journal of Software Engineering and Applications (2014)
- [20] Wetter, M., Grahovac, M., Hu, J.: Control description language. In: Proceedings of The American Modelica Conference 2018, October 9–10, Somberg Conference Center, Cambridge MA, USA. pp. 17–26. No. 154, Linköping University Electronic Press (2019)
- [21] Wilde, E.: Putting things to rest (2007)
- [22] Yang, L., Cormican, K., Yu, M.: Ontology-based systems engineering: A state-of-the-art review. Computers in Industry **111**, 148–171 (2019)
- [23] Yoshioka, M., et al.: Physical concept ontology for the knowledge intensive engineering framework. Advanced engineering informatics **18**(2), 95–113 (2004)