Received September 30, 2021, accepted October 23, 2021, date of publication November 8, 2021,
date of current version November 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3126000

Intention as a Context: An Activity Intention Model for Adaptable Development of Applications in the Internet of Things

VICTOR PONCE, (Student Member, IEEE), AND BESSAM ABDULRAZAK, (Member, IEEE)
AMbient Intelligence Laboratory (AMI-Lab), Université de Sherbrooke, Sherbrooke, QC J1K 2R1, Canada

Corresponding authors: Victor Ponce (victor.ponce@usherbrooke.ca) and Bessam Abdulrazak (bessam.abdulrazak@usherbrooke.ca)

This work was supported by SENESCYT, Ecuador and Université de Sherbrooke, Canada.

ABSTRACT End-users, such as aging people, look for dynamic applications to help them in their daily activities while usually assisted by domain experts (e.g., physicians) in performing these activities. Nowadays, the Internet of Things (IoT) augments applications with a changing context based on the activities, environments, and services. Unfortunately, most IoT application development tools are restricted to specific scenarios or involve technical challenges. We propose an activity intention model for quick application development, targeting domain experts who are not traditional software developers. Our model is based on the ContextAA micro context-awareness approach with autonomic computing-based components providing pervasive adaptations. Unlike other Internet of Things application definitions, our model promotes elicitation of the activity semantics and provides mechanisms to compute the semantics. We then generate intention as a context (IaaC), which contains a self-described activity intention context with compiled knowledge ready to be assessed in pervasive smart environments for augmented adaptations. Experimental results show a potential resource optimization for dynamic Internet of Things applications in smart homes and smart cities.

INDEX TERMS Activity model, context-aware application, intention model, Internet of Things, semantics.

I. INTRODUCTION

Computer applications contain a representation of end-user intentions (i.e., target goals/actions obtained in requirement elicitation). In general, we can determine how suitable an application is by asking end-users if an application's intention aligns with their intention when performing an activity. To better achieve end-user intentions, application developers collect knowledge from diverse sources and describe their interpretation when creating applications. However, omissions in intention representation can produce applications with incorrect system actions/responses. Since end-users constantly demand applications to help in their daily activities, the intention representation becomes more complex but essential.

The Internet of Things (IoT) provides a promising new generation of surrounding pervasive technology and smart environments [1], e.g., smart cities, smart transportation. With the flooding of IoT devices and smart environments, new application requirements are increasing as well [2]. Furthermore, end-users demand dynamic applications connected to IoT devices for activity automation at home [3]. Similarly, available services on the Internet increase the context for applications, e.g., Amazon Alexa skills (i.e., domains developed by third-party developers) have grown from 40 000 in 2018 [4] to more than 100 000 by the end of 2019[1]. Among other challenges (e.g., integrating available devices, protocols, and services; energy optimization; providing security and big data [1], [5]), we have identified the representation of activity intentions as an essential aspect to perform smart environment actions.

Representing activity intentions starts by identifying end-user goals, which complicates application development, especially when developers are not aware of the end-user goals. Domain knowledge also plays an essential part when developing an application, but developers are not experts in all domains. Consequently, the gap between applications' intention and end-user intention increases because it depends on a) the developers' interpretation of additional/unknown attributes and knowledge; and b) changing context from

The associate editor coordinating the review of this manuscript and approving it for publication was Sabah Mohammed.

[1]https://voicebot.ai/2019/12/18/amazon-announces-100k-smart-home-products-support-alexa/ Last accessed on November 3, 2021.

IoT devices and available services. To decrease this gap, developers have introduced configuration/application templates, parameterization [6], and machine learning domain classification to avoid erroneous system actions/responses [4]. Even though these mechanisms allow end-users to personalize and run the applications, they restrict the applications to changing fixed attributes and machine learning accuracy, limiting end-user intentions' achievement. It is necessary to introduce new personalization approaches because pervasive applications involve dynamic attributes that make difficult the goal interpretation. It is also essential to reduce the implementation time because IoT devices are in constant deployment.

A valuable source of knowledge for applications is domain experts (e.g., practitioners). They are well prepared to assist end-users; they recognize end-user situations and behavior by efficiently combining experiences with available and external resources. Furthermore, domain experts can support the development process with their expertise. They can even decrease the level of requirement interpretation and implementation time by becoming application developers. Thus, increasing the domain expert's involvement in IoT application development is key for achieving end-user activity intentions. However, domain experts do not have application development expertise. Therefore, it is necessary to empower domain experts with adaptable tools that enable them to manage applications.

We present in this paper an approach to represent activity intentions in applications. Our approach involves a model to elicit the activity semantics and mechanisms to compute the semantics. Following our approach, domain experts[2] can develop application flows representing: first, the activity terminology with relevant features of the domain, and then, key attributes to achieve end-user intentions. The semantics' computation produces an enriched context with the activity intention (which we name Intention as a Context (IaaC) applications). IaaC applications are context-reduced and self-contained, i.e., they contain a complete representation of the application logic and domain knowledge required to achieve the intention. IaaC applications can be deployed in smart IoT environments, and their self-contained logic and knowledge increase the features to improve system activity recognition/interpretation accuracy. To validate our proposed approach, we designed a testbed to simulate a complete development process, from building to deploying applications. The testbed allows gathering context from real IoT devices and deploys IaaC applications to run in a smart environment IoT nodes [7]. We also evaluated the scalability of IaaC in a distributed system scenario.

The major contributions of this paper are summarized as follows:

- Intention as a Context (IaaC): **A novel representation for context-aware applications**, where the applications are self-described with compiled knowledge to provide augmented adaptations in pervasive smart environments. This self-described representation is used to define activity elements (e.g., user preferences, environment actions). With the activity elements, domain experts (i.e., non-technical people) can create self-described applications in an easy-to-use flow paradigm.
- **An approach to model activity intentions**, including the application domain knowledge to enhance application descriptions. The intentions can be reused, interrelating intentions by connecting their activity elements. Furthermore, the activity elements are extensible; they allow us to describe corresponding attributes (e.g., synonyms). They are also adaptable for connecting them in sequence (flow) to simplify the expression of activity intentions.
- **Semantic constructs** to describe applications with an algorithm for semantic reduction, matching the application semantics and reducing it to a minimum expression to run in smart environments. The semantic constructs are also represented as programming building blocks in a graphical user interface (GUI). Then, domain experts can use the programming constructs to build context-aware applications.
- **A testbed to develop and deploy adaptable IoT applications** connected to a sensor/actuator platform and a smart city simulator. Using the testbed, we simulated the complete development process: Domain and activity intention elicitation, semantic reductions, automatic deployment, and running in smart environments.

We organize the paper as follows: Section II summarizes existing artifacts to model intentions and activities. In Section III, we propose our approach to represent and compute the activity intention. In Section IV, we describe our model and implementation, providing details about semantic computation. In Section V, we present the validation and preliminary test results. In Section VI, we discuss our approach and future work. Finally, we present the conclusions in Section VII.

II. LITERATURE REVIEW: MODELING INTENTIONS AND ACTIVITIES

End-user intentions are analogous to high-level goals that represent desired outcomes (e.g., drinking water), having a higher abstraction than concrete end-user actions (e.g., grasp, reach) [8]. Thus, an ***intention model*** involves the representation of anticipated outcomes that guide end-user activities required to achieve a goal. The corresponding ***activity model*** introduces appropriate definitions for supporting the accomplishment of end-user intentions. Following, we present the relevant background of existing modeling approaches.

A. END-USER INTENTION

User intentions in human-computer interaction (HCI) are becoming an important input data for smart environments, together with other user data such as users' location,

[2]Domain experts are the "users" of our approach to develop applications for end-users.

TABLE 1. Approaches for activity modeling.

	Key aspects	Example of model artifacts (representation and methods)	Our approach (details in Section III)
Expressive	• Exploits the activity semantics • Describes activity elements and relationships • Uses the semantics to interpret activities	• **Immutable components:** Fixed description, e.g., Attributes to describe activities such as names, action behavior (animation), effect, preconditions for ADL simulation [17] • **Logic formalism:** Rules, e.g., Event Condition Action (ECA) rules to describe activities [18] or actions [19] • **Ontology:** Categories + rules/language, e.g., Description Logic (DL) to describe and infer activities [20]; Semantic Web Rule Language (SWRL) to describe and infer the relationship between people, activity, and other elements (e.g., time, events) [21] • **Language:** Domain-specific language (DSL), natural language (NL). E.g., the Asbru language adopted to create a DSL for hierarchical ADLs (activity and sub-activities) [22]; NL grammatical relationship through case frames (i.e., concept hierarchy of actions, e.g., agent, objective, localization, source, goal) [23]	• Logic formalism with rules to describe smart environment actions to support activities • Ontic perspective (instead of ontology) to keep a compiled knowledge of each application • Programming language is abstracted to simplify application elicitation for non-software experts
Discriminative	• Creates a predictive model • Associate activity labels • Apply a method to discriminate the labels	• **Heuristic:** an algorithm or rule-based activity patterns to simplify the discrimination, e.g., neural networks [24], linear or nonlinear discriminant learning [20] • **Probabilistic:** Discriminate using the conditional probability given an observation of activity components, generally modeling using undirected graphs [25], e.g., associate sensed data to following activities using the conditional random field (CRF) [26] • **Regression** [27] **and classification** [28] **trees** to discriminate activity labels • **Enhance training data set**, e.g., labeling from social media collecting human behavior [29]	• Our approach focuses on pattern matching at this stage of our work.
Generative	• Create a complete map of the activity and elements • The design is static, regarding the model	• **Probabilistic:** Model of the activity and elements (e.g., data from sensors, end-user preferences, tasks, actions), e.g., Bayes network to represent "situation templates" [30]; Dynamic Bayesian Network (DBN) considering changing events in time [31]; Hidden Markov Model (HMM) considering hidden states (i.e., activities) and observable states (i.e., data from sensors) [32]. • **Labeling:** Combine with the discriminative approach, e.g., DBN with labeled actions to infer actions and then activities [33]. Incremental learning [34].	• Our approach focuses on pattern matching at this stage, and we target incremental learning as future work.
Pattern matching	• Exploits the activity syntax • Describe patterns • Use a matching algorithm considering the patterns	• **Matching rules:** e.g., model activity elements (e.g., resources, services, roles, rules), and matching rules [35] • **Matching events:** e.g., hierarchical activity/action model and matching an event corresponding to single actions (e.g., "turning on a light"), and then inferring activities (sequence of events) [36] • **Matching semantics:** e.g., business activity ontology model performing semantic matching using SPARQL (i.e., a query language for semantic constructs) [37] • **Matching context:** e.g., context descriptions and matching semantic distances between contexts [38] • **Matching domain:** Model activity domain and matching activity elements against domain definitions, e.g., matching activity with concepts and relations [39]	• Our approach matches the activity semantics and related elements (i.e., rules, context, domain) with pattern matching algorithms. • We consider domain experts (e.g., physicians) as application builders who will describe appropriate patterns.

posture, emotions, and habits [9]. Thus, HCI designers regard end-user intentions for subsequent evaluation of the design, i.e., matching intentions with the current system state. Furthermore, the software development process regards intentions to improve quality. Software developers use diverse mechanisms (e.g., annotations, semantics) to maintain their intended design in software artifacts, e.g., use cases into the source code [10]. Other mechanisms allow expressing the intention of informed end-users (i.e., non-technical users with domain knowledge such as smart home users) in applications, e.g., through domain modeling [11], end-user software engineering [12], and intent elicitation on intelligent assistants such as Google Assistant, Amazon Alexa, Microsoft Cortana, Apple Siri [4]. Thus, representing intention is a key aspect to augment the system's understanding of end-user requirements. Likewise, end-user intentions complement Artificial Intelligence research in goal recognition algorithms [13] and intention/action recognition in human-robot interaction [14]. In general, intentions are reduced to actions to perform. For example, (a) in the Belief-Desire-Intention (BDI) architectures, an intention is a component that represents high-level plans which are refined to basic actions [15], and (b) in BDI-based context-aware frameworks, for considering end-user intentions to enrich the situation [16]. In BDI architectures, intentions cannot be reduced only to beliefs and desires, and they can contain a kind of permanent goal influenced externally, e.g., with events [15]. Consequently, BDI intentions are a unique state that links them with behavior, environment, other components of the system (e.g., rules, processes), or end-user activities.

B. END-USER ACTIVITY

Researchers have proposed diverse approaches for activity modeling (Table 1). They range from using (a) modeling artifacts to allow creating a complete model through an expressive representation to (b) advanced mechanisms to infer unknown activity elements (e.g., activity, rule):

1) The expressive representation approaches simplify manipulating known activity elements and facilitating application development by non-technical end-users [39]. They exploit the activity semantics to describe activity elements and relationships for further interpretation of activities. They integrate diverse methods to express activity terminology. The drawback is that the modeling considers fixed descriptions of the activity elements. Therefore, it is usually combined with other approaches to support complex problems (e.g., recognize an activity related to an unknown sensor input).
2) Discriminative, generative, and pattern matching representations are approaches for manipulating unknown elements [40]. They introduce variables, data sets, mathematical models, and detailed patterns to link known and unknown activity elements - for example, associate activity elements with activity labels for discriminative activity recognition [27]. The drawback of the discriminative and generative approaches is that the modeling requires large data sets for training. Also, applying a model and learning results for the same person (e.g., from one body part to another) [41] or from one person to another [20] is challenging. We discard discriminative and generative approaches because we are working on the requirement elicitation and data collection at this stage of our work. The pattern matching approach's drawback is that the modeling requires the description of static patterns for matching, making it challenging to model dynamic activities. Thus, we target our solution for domain experts who can provide their knowledge to describe the required patterns.

C. SEMANTICS IN CONTEXT-AWARE APPLICATIONS

Context-aware platforms provide features to enrich the semantics for activity modeling and activity recognition. Ye *et al.* [42] presented a review of semantic web artifacts to represent, process, and reason on the intended semantics from heterogeneous sources. One approach uses ontologies as a formalism to capture and share activity semantics in context-aware applications, e.g., an ontology to describe a set of actions to achieve a goal using smart devices context through the Web (Web of Things) [43]. Ontologies have also been combined with other artifacts (Table 1) to increase expressiveness [42], e.g., a) language or overlapping rules for expressing a richer semantic to capture the interrelation of the dynamic context (e.g., time, location, actors, resources) and b) event processing techniques/engines to match the ontology with event patterns and context from sensors.

A second approach is to limit the activity semantics to pre-defined concepts with rules connecting them as input_context → task/action → output_context, e.g., associating context conditions with activity workflows [44] or activities expressed with diverse apps such as Workflow[3] and IFTTT.[4] Similarly, other approaches restrict the semantics to a pre-defined model of actions such as activity theory-based (i.e., activity decomposed into actions and operations, relating elements such as subject and rules to produce an activity outcome). For example, Li and Landay [45] adopted the activity theory elements and proposed an activity language to describe activity characteristics. Huang and Gartner [46] extended the activity theory elements to include complimentary human activity components such as the physical environment of the activity, creating a mapping between context categories and activity theory elements, e.g., user context relates "subject," task context relates "outcome," social context relates "rules," and environmental context relates the "physical context" of the activity.

In general, context-aware application developers require to learn/apply technical artifacts such as ontologies or domain-specific languages, e.g., activity query language in [45]), or other artifacts (Table 1). Our work attempts to empower end-users such as domain experts (i.e., non-technical people) to define activity intention. Thus, instead, we provide mechanisms to compile the semantics to be processed in our context-aware platform ContextAA.

D. ContextAA PLATFORM

We adopted ***ContextAA*** (Context-Aware Agents) [7], [47], [48] for activity modeling and interpretation in pervasive environments. ContextAA enables context-aware support in dynamic settings (e.g., pervasive e-health through a city), maintaining pervasive service provision close to users. The design of ContextAA focuses on working in resource-constrained hardware and being portable to mobile devices.

We extend ContextAA's context representation to describe activities. ContextAA considers the context representation of traditional categories (e.g., location, time) and includes operations, situations, and agents' characteristics as context. ContextAA manages the meaningful context (by different agents, maintaining agent missions) and facilitates the migration of agents among components on the ContextAA platform.

The ContextAA platform is a self-organized architecture that focuses on a micro context-awareness of distributed agents, perceiving their local environment and acting based on their roles. The architecture includes host components, agents, context, and context space (for both host and agent).

- Host components serve as middleware, interacting with others and integrating external services. Also, host components isolate the network and manage the deployment and execution of agents, working with local knowledge (i.e., ontic perspective).
- Agents are context-dependent entities responsible for executing actions, differentiating between standard and

[3]Workflow for iOS, iPadOS (accessed on November 3, 2021, https://my.workflow.is)
[4]IFTTT (accessed on November 3, 2021, https://ifttt.com)

user (or domain-specific) agents. A standard agent implements host services, having privileges on its host. User agents perform context-aware tasks to solve specific problems defined by each agent's mission.
- Context is stored in context spaces – organized knowledge repositories at both the agent and host levels. An agent's contexts (stored in context space) are immutable, enabling hosts to share and synchronize context between agents. Requests for context are also described in the form of context, mediated through context space. A host consumes requests using standard agents and searches in their context space and through neighboring hosts.

In the following section, we present our approach for modeling activity intentions.

III. APPROACH: INTENTION AS A CONTEXT

We propose an ***intention model*** to facilitate and simplify context-aware application development. Our objective is to take advantage of domain experts' knowledge to describe end-user intentions' semantics on activities, i.e., experience on the outcome of activities. Describing activity intentions rather than specific situations can help to elicit an in-person purpose of an application. It also enables the preservation of the intention in diverse IoT settings, from smart homes to smart cities. We also attempt to provide an IoT smart environment independent of fixed system components (e.g., unique devices, specific context). The activity intention can be introduced into context-aware system processes as well. As a result, IoT smart environments can interpret activity intentions and provide services following end-user goals.

Our model subsumes the activity and exploits the activity semantics by defining simplified activity elements (e.g., action, end-user profile). The activity elements are extensible; they allow us to describe corresponding attributes (e.g., synonyms). They are also adaptable for connecting them in sequence, simplifying expressing activity intentions. At the same time, the intentions can be reused, interrelating intentions by connecting their activity elements. We analyzed diverse modeling artifacts (Table 1) to simplify application development for non-software experts. As a result, we combine the expressive and pattern matching artifacts (Section II) for intention description and semantic computation, respectively (Table 2):

1) We adopt the logic formalism, i.e., rules, to describe activity circumstances (e.g., triggering conditions, desired preferences). It keeps modeling simple, especially for domain experts (i.e., non-software experts) to express their knowledge. We discard the use of languages for intention elicitation because they incorporate terms that overload domain experts. Instead, we only consider a set of keywords to categorize programming constructs. However, at the implementation level (i.e., not seen by the domain expert), the semantic definition/computation follows ContextAA's language, expressed in the extended Backus-Naur form (EBNF) context-free grammar.
2) Similarly, we discard an ontology-based model because we don't attempt to define specific categories, i.e., specific activities, of a particular environment. On the other hand, we target dynamic activities whose knowledge is diverse, changing every day. However, we maintain compiled knowledge (i.e., ontic perspective) for each application. In our approach, domain experts define the activity intention, express the activity semantics, and label activities properly.
3) We designed an action machine to match rules and the semantic of the compiled knowledge. The action machine compiles related activity circumstances to create IaaC applications with simplified scripts (i.e., IaaC assignments). Then, domain experts can deploy IaaC applications in smart environments (IoT platforms), where the semantics computation is performed through context matching.

TABLE 2. Intention model and semantics computation artifacts.

Model Artifact (Table I)	Expressive	Pattern Matching	*Intention (1) description (end-user), and (2) computation*
Logic	✓		**(1) Logic rules**
Ontology (Ontic)	✓		**(1) Compiled knowledge (i.e., Ontic perspective)**
Language	✓		**(1) Keywords (2) EBNF grammar**
Matching rules		✓	**(2) Action machine**
Matching semantics		✓	**(2) Action machine**
Matching context		✓	**(2) ContextAA (IoT platform)**
Matching domain		✓	**(2) Action machine**
Label (e.g., ADL)	✓		**(1) By domain experts**

We base our model on the micro context-aware approach running in *ContextAA* – a smart environment IoT platform [7]. In this approach, autonomic computing components analyze context and produce proper responses in multiple computational spaces. Micro context-aware components include context operations to assess the context semantics for an improved adaptation. We implemented the tools to enable domain experts as developers of IaaC applications corresponding to micro context-aware applications. They include a self-description of the activity intention and knowledge in terms of the *ContextAA* semantics. A deployed application in the IoT smart environment then integrates the self-description into the autonomic computing components' running setting.

We reified our model by creating *programming constructs*, i.e., *semantic elements* used to represent activity elements. Then, domain experts can use the programming constructs to build context-aware applications. We also propose an *action machine* to compute our semantic elements. The action machine's outcome is IaaC applications with a reduced set of IaaC assignments ready to be deployed in smart environments. The constructs and the action machine are part of our *semantic framework* for micro context-aware applications

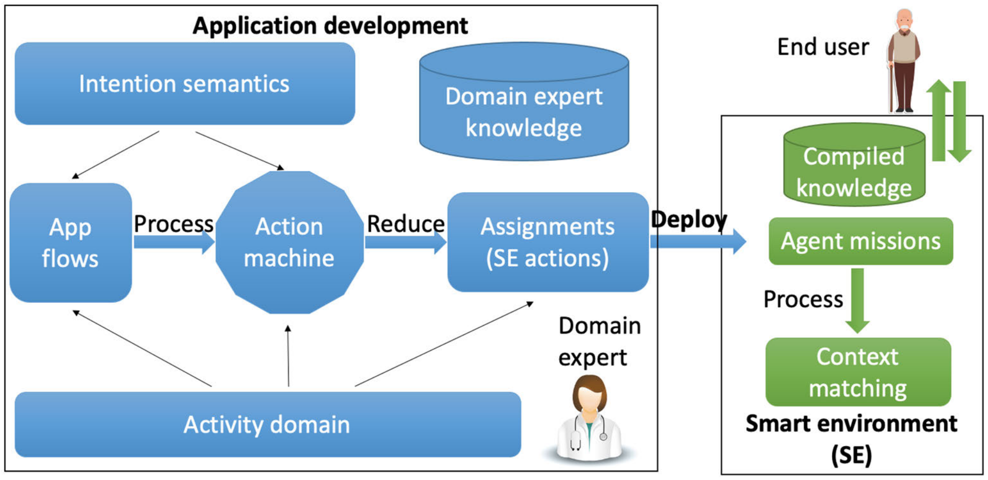


FIGURE 1. Intention model components.

(called AmI-DEU: Ambient Intelligence Domain Expert User) [11]. Our framework facilitates using the constructs, programing the action machine, and deploying applications to our *ContextAA* micro context-aware platform.

Finally, we validate our approach in a testbed designed to simulate a complete development process, from building to deploying applications. The testbed allows gathering context from real devices, which is an important aspect contributing to our design regarding real context. We tested our approach defining use cases of applications for smart homes and smart cities. Following, we address our conceptual model to represent activity intentions.

IV. INTENTION MODEL AND IMPLEMENTATION

Our model defines semantic elements to take advantage of domain expert knowledge and activity terminology to express end-user intentions in application flows. An action machine then reduces the intentions to create assignments for being deployed in smart environments (FIGURE 1).

A. ACTIVITY DOMAIN

Contrary to general purpose applications, we focus on creating dynamic context-aware applications, i.e., applications that require rapid development and continuous deployment [11]. These applications can run for everyday activities in diverse domains, making them unmanageable using domain-specific models. First, we analyzed activity modeling approaches (Section II) to identify common activity elements and terminology. Then, we define a hierarchical classification of activity intentions [39] where "actions" are the basis for modeling activities. Combining actions can produce well-known activity-related intentions that simplify the development, from simple intentions to complex intentions (FIGURE 2). Furthermore, including the possible relations between activities is a challenging problem because they depend on the combination of each aspect of activity and end-user profile. Thus, we include in our model a representation

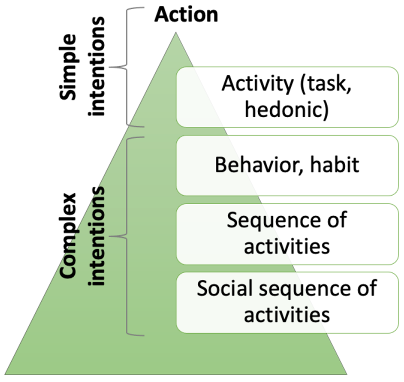


FIGURE 2. Activity intention hierarchy.

of the applications' domain to facilitate the description of activity elements and end-user profile (1).

Definition 1: a domain D is a set of the tuple $D = \{\langle \varsigma, \rho \rangle\}$ where $\varsigma \in K$ is a set of concept elements of the universe of concept knowledge K and ρ is the set $\{\langle \varsigma_i, r \rangle: \varsigma_i, r \in K \times \Phi\}$ - the relation to other concepts, where Φ is the universe of relations.

We consider domain experts as the "user" of the model to build applications. We aim to encourage domain experts to define the concepts and relations of the domain for applications, rather than using fixed ontologies or other approaches that constrain applications to pre-defined concepts/relations. Regarding the diverse aspects of domain modeling [49] and activity modeling (Section II), we only model the activity domain's concepts because of the dynamicity of activities. We discard, among others, domain constraints, organization, and division of labor modeling (e.g., regulations, roles). However, we envisage the hierarchy classification for defining "templates" of activities with further aspects of domain modeling to complement the current concept/relation modeling. For example, to define a domain for social intentions, with multiple end-users performing activities,

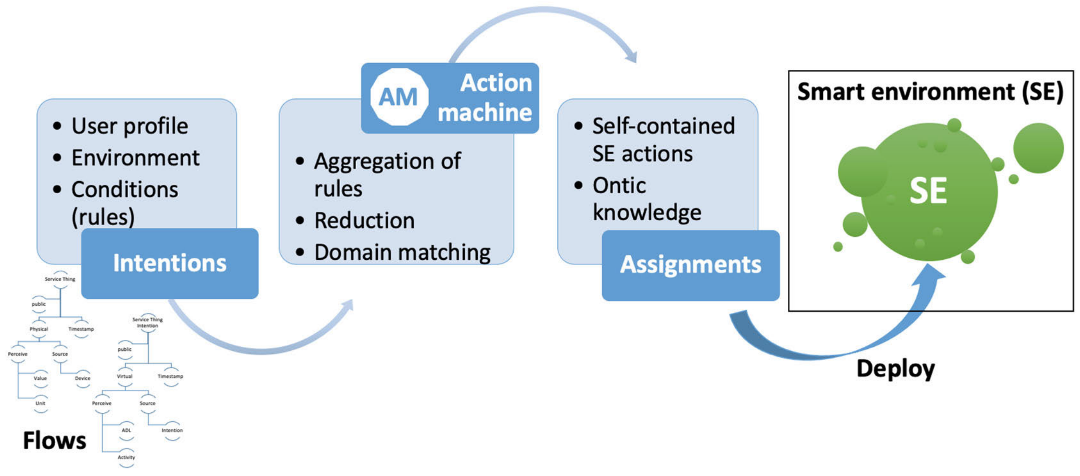


FIGURE 3. The action machine reduces all intentions to a set of IaaC assignments. Domain experts describe domain and activity intention in application flows, and the action machine processes the semantics, reducing it to a minimum expression to run in smart environments.

e.g., to describe social regulations such as community organization/guidelines.

B. INTENTION SEMANTICS

The intention semantics, which subsumes the activity semantics, allows domain experts to describe simple action flows to represent *simple intentions*. A flow can describe activity actions with input/output context (for intentions regarding activities, tasks, and end-user actions), as well as actions based on end-user preference/profile (for hedonic intentions). We aim to use intention templates for *complex intentions* (i.e., intentions regarding behavior, habit, and sequence of activities). In any case, a flow includes a time configuration for repetition, a priority to distinguish between two overlapped intentions, and an order to define the sequence of executions.

We associate the activity elements in an **intention net** (Definition 2), containing a complete application description map. The intention net represents the connection between application flows and activity domain concepts, including the relationships, attributes, values, conditions, and configurations. Each flow path is independent, including the transition's weights and conditions, having the required context in each flow step. The flows and intentions can also be interrelated, defining new paths. The end of each path must include actions to execute when all the paths' conditions are satisfied.

Definition 2: An intention net is a tuple $N = \langle I, A, P \rangle$ where I is a set of intentions, A is a set of actions, and $P \subseteq I \times (E \times E) \times A$ is a set of paths where E is an activity element. An action in a path $pa \in N$ is a tuple $pa = \langle \cdot c, a, c\cdot \rangle$, where a is an element of the set A, $\cdot c$ is the set input context of a, and $c\cdot$ is the set of output context of a.

In Definition 2, the concepts of the activity domain are part of the input/output context. We discard the state of the intention because the intention flow defines a set of non-connected actions. The state of the intention net is independent of the paths. It only depends on the context of the action regarding the concepts, weights, conditions, and configuration (i.e., schedule, order, priority). Thus, the intention net raises the level of independence for a domain expert when composing applications. An action path pa is "followed" when all its conditions are satisfied regarding its input context $\cdot c$, and its context values gathered in the smart environment. Upon the "followed" path $p \in P$, an action a is executed (in the smart environment), and values in the output context $c\cdot$ become part of the most recent context in the current path p.

Intentions and their flows are reduced to IaaC assignments ready to be deployed as smart environment actions, processing their semantics in the action machine.

C. ACTION MACHINE

Intention flows can contain many attributes, overlapping conditions, and redundant paths related to one or more activities. For example, a domain expert can overlap the intention to go for a walk with the intention to notify (an end-user) to bring medicine if going outside. Thus, we designed an **action machine** to process the intention net. The action machine's main functionality is to reduce all intentions to a set of IaaC assignments (Definition 3), which corresponds to smart environment actions. Each IaaC assignment is a self-contained context that includes the conditions and the required input/output context (defined in the intention flow paths). A central design aspect is that the action machine matches the activity domain, adding compiled knowledge to the assignments. Smart environments can then execute assignments without dependencies for performing the intention in different settings (FIGURE 3).

Definition 3: An IaaC assignment is defined by $\alpha = \langle O, X, \{X_a\}_{a \in A}, \{G_p\}_{p \in P}, \{U_a\}_{a \in A} \rangle$ where:

- $O \subseteq M \cup D$, D is an activity domain, M is the set of all contexts. O is the compiled knowledge that contains the context of α (i.e., concepts, relations, environment).

TABLE 3. IoT platform semantics.

ContextAA element / operation	Description (what they are in ContextAA, i.e., smart environment runtime) [48]
Context representation: $Context = name\{value\}$	Self-description of any context, including agent description, agent mission description, and operations in ContextAA platform, e.g.: $host\_0: agent\_1\left\{\begin{matrix} temperature\left\{\begin{matrix} value\{20\} \\ unit\{celsius\}\end{matrix}\right\} \\ source\{agent\_1\}\end{matrix}\right\}$
Criteria on Context, e.g., is_date, is_number, is{pattern}, has{pattern}	It is applied to produce a Context or a list of Contexts that matches the criterion or criteria. In ContextAA platform, it is used to keep “keywords” to a minimum and to manipulate Context, for example, in algorithms to find and filter Context
Context matching: $structured\_like\{Context_1, Context_2\}$ $valued\_like\{Context_1, Context_2\}$	Applied as an approach to compute the similarities between Contexts. It is also applied to compute the semantic distance, defined as: $distance = \frac{structured\_like + valued\_like}{2}$
Computation: $eval\{expression\}$	Semantic of agent missions and computation for agent missions, e.g.: $eval\{"migrate = battery\_val < 20"\}$

- X ⊆ O is a set of variables representing the attributes of the conditions
- For each action a ∈ A, X_a is the set of condition variables of the action a.
- For each path p ∈ N, G_p is a predicate defined over X, N is the intention net.
- For each action a ∈ A, U_a ⊆ O is the set of variables updated by a (i.e., context with new values).

In Definition 3, the predicate G corresponds to the condition to check before executing an assignment action. However, conditions are not always necessary, e.g., a “display message” action executed at the beginning of the application to show a notification, updating the variable U_{ai} = “new message.” Similarly, an intention can define triggers to execute based on (a) time, e.g., display a message every day; or (b) checking initial conditions either if X ∈ D (e.g., display a message if the room temperature is greater than 35 degrees, for the smart home domain) or X ∈ M (e.g., display a message if the temperature is greater than 35 degrees, everywhere/all domains). In any case, an assignment α_i considers the nearest context, matching the compiled knowledge with the available context in the smart environment when deployed.

D. MODEL IMPLEMENTATION

We implemented the proposed model as part of our semantic framework for context-aware applications, called AmI-DEU-Semantics (Ambient Intelligence Domain Expert User Semantics) [11]. The framework processes semantic elements (i.e., context-based value types) that we developed to build applications. The semantic elements include basic data types (e.g., numbers, Boolean, text), composite data types (e.g., functions, lists, maps), and semantic entities (abstract computational entities, e.g., concept type, process type, service type). We defined and implemented new semantic entities to describe applications (i.e., activity intention flows) based on our framework. Afterward, we implemented the action machine’s functionalities to reduce the applications to create IaaC assignments ready to run on our IoT platform. IoT platform semantics.

The implementation follows the micro context-aware approach [7] that runs in our IoT smart environment implementation *ContextAA* (Context Awareness Agent-based) platform [38]. Micro context-awareness agents are autonomic computing-based components with a self-contained *Context* (i.e., *ContextAA* context representation). They include processes to analyze and interpret the *Context* for producing proper responses. The *Context* (with capitalized ‘C’) is a simple recursive formal tuple *(name, value)*, where *name* is a globally unambiguous name (i.e., two *Contexts* are distinct if they have distinct names); and *value* is a set of *Contexts*. A value in *Context* follows the specification for self-descriptive data, where the value description has a meaningful significance for individual agents. In the *ContextAA* platform, *Context* expresses the usual context, as well as operations and entities that request and publish *Context*. Table 3 lists examples of applied ContextAA semantics in AmI-DEU (the complete ContextAA semantics is summarized in [48] annexes A to E). The ContextAA core semantics is expressed in the extended Backus-Naur form (EBNF) context-free grammar. The grammar defines the Context representation that allows a self-description of any context (i.e., vocabulary convention), which we simplify in Table 3 as *Context = name {value}*, where *value* ∈ *Context*. It also defines primitive operations to validate, manipulate, and compute Context expressions such as is_date, filter_if, eval, etc. The current implementation of ContextAA is in C++ based on the design and implementation of the C++ Standard Template Library (STL). ContextAA combines programming paradigms such as object-oriented and functional programming with predicates to transform Context.

This section presents our new *Context*-based elements and expresses *Context* in a canonical form $c = n\{v\}$, meaning *Context* has name n and value v.

1) CONTEXT-BASED PROGRAMMING CONSTRUCTS

AmI-DEU semantics *Context*-based programming constructs [11] are the basis for our new *Context*-based semantic elements to represent intentions. The elements inherit from the unique abstract value type CTerm (*Context* Term) defined as a tuple CTerm = ⟨ type, description⟩ where type = n {v} is a *Context* that contains a unique token representing the type of term (e.g., "concept"), and description = n {v} is a *Context* that contains the self-description of the element. For example, we describe a concept as a term CTermConcept = (type, context), where:

- type = concept_name {TERM_TYPE {"concept"} }.
- context = concept_id {properties {list_props} relation {name {a_name} weight {a_weight} } }.

Each CTerm has a unique ID (concept_id in the previous example) for being identified. Our implementation distinguishes the programming constructs through keywords, e.g., TERM_TYPE, INTENTION_PATH. Unique IDs and keywords allow high-level processing of the semantic elements in the AmI-DEU semantics framework. However, after processing to create IaaC assignments to deploy, the framework only maintains fewer keywords required for being interpreted in *ContextAA*.

We define a finite set of semantic elements CTERM = {CTermConcept, CTermIntention, CTermEntity, CTermCondition, CTermAction}. The element CTermConcept enables the definition of the activity domain. The element CTermIntention relates the other semantic elements to define the intention net. Following, we describe our semantic elements, their structure, and our implementation.

2) INTENTION REPOSITORY

We store the semantic elements in a singleton repository. We implemented it as a symbol table R = (idx, aTerm) where (idx, aTerm) ∈ ID × CTERM. ID ⊂ ℵ is the index set, and aTerm ∈ CTERM (the semantic element set). We implemented a mutex synchronization for accessing the repository, a function DCAST: CTERM → aTerm for dynamic casts (polymorphisms), and utility functions to add, delete, and query the repository. We also manage repository persistence (store and read functions).

3) ACTIVITY DOMAIN

We consider concepts with two kinds of direct associations: 1) part-whole relations, e.g., "living room" contains "temperature sensor," and 2) weighted relations, e.g., " a person" always - weight=100% - prefers "leisure." These two associations, including their combination (e.g., "living room" often - weight=80% - contains a robot), enable domain experts to define the activity domain (1).

The semantic element "Concept" represents any activity domain concept and its associations between concepts. Its structure is CTermConcept = (uid, name, List_attribute, List_relation), where:

- uid is a unique identification of the concept; name is the concept name (meaningful token).
- attribute = (name, value), where: **name** is the attribute's name; **value** is the attribute's value.
- relation = (type, description, weight), where: **type** is the relation category, which can be a part-whole relation or a weighted relation; **description** is to add information about the relation; **weight** ∈ [$\text{weight}_{\min}$, $\text{weight}_{\max}$] to determine a relation preference degree.

We implemented the activity domain concepts and associations as a directed acyclic graph A = (Ω, P). The node set Ω is a set of Concepts of the domain $\{\omega_1, \ldots, \omega_n\} \in K$, where K is the universe of concepts. The set of edges P is a set of relations $\{\langle\omega_1, \omega_2, r\rangle: \omega_1, \omega_2 \in \Omega, r \in \Phi\}$, where Φ is the universe of relations. A concept has zero or more relations (FIGURE 4).

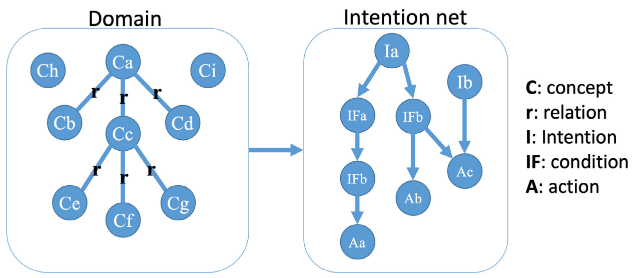


FIGURE 4. Domain and intention graphs.

4) INTENTION NET

We implemented the intention net (Definition 2) as a tuple CTermIntention = (uid, name, List_property, List_element). We include properties such as **priority** (the predilection for the intention); **category** (the intention "type," e.g., hedonic—which considers preference degrees, goal-based—which best matches concepts/relations); and **schedule** to represent time constraints, e.g., for repetition of the intention. An element e_i ∈ List_element represents the next element in the flow. It could be **a condition** which structure is (uid, name, List_property, List_element); **action** which structure is (uid, name, List_property, order), where order defines the sequence or predilection for execution; or another **intention**.

Each path of the intention net contains a sequence of elements that describes the action. A "condition" element enables the decomposition of an action path into context validations to accomplish the action. Conditions allow to describe operators on strings and numbers (e.g., equals, greater than, is true), as well as to combine with statements (e.g., else) and logical operators (i.e., and, or) for creating composite rules.

We implemented the intention net as a directed acyclic graph Γ = (E, Π). The node set E is a set of semantic elements

$\{\varepsilon_1, \ldots, \varepsilon_n\}$, $\varepsilon \in$ {Intention, Entity, Condition, Action}. The set of direct edges Π is the set $\{\langle\varepsilon_1, \beta, \varepsilon_2\rangle$: $\varepsilon_1, \beta, \varepsilon_2 \in E_\beta \times M\times E\}$, where M is the set of all contexts. An edge $\varepsilon_1 \rightarrow \varepsilon_2 = (\pi_1, \beta, \pi_2)$ in the edge set Π expresses that the environment surrounding the element ε_1 is affected by the context β, and all together form the environment surrounding the element ε_2. Thus, $\beta_{1,2}$ represents the payload context, and the union of all paths' payloads forms the accumulated knowledge. In the graph, the semantic element Intention is a root node (i.e., a node with no parents), and the semantic element Action is a leaf node (i.e., a node with no children) (FIGURE 4).

5) ACTION MACHINE

The action machine enables a reduction of the intention net through semantic processing. The output of the action machine is a set of IaaC assignments (Definition 3) corresponding to the actions of the intention net composed by the set of conditions to accomplish the action. Algorithm 1 shows the recursion implemented in the action machine to reduce the intention net to a set of IaaC assignments.

Since our model is based on a flow of actions starting from the root node (i.e., Intention), Algorithm 1 follows paths, accumulates the *Context* of each path, and creates an assignment with action elements. A particular case is the element "condition," where it is necessary to accumulate all paths from the root node (currentIntention) to the current condition node (currentElement) (Algorithm 1 line 9). Then, each assignment's set of conditions becomes the context to be evaluated in the path to achieve an action. Finally, the algorithm queries the repository R to enrich the *Context* with the activity domain (Algorithm 1-line 13), creating assignments with accumulated knowledge.

The action machine also compiles the assignments to produce a minimal, self-contained *Context*. The compilation aims to match the intentions' elements (Definition 2) with the activity domain (Definition 1) to create compiled knowledge. We implemented two approaches for domain matching: 1) concept matching to find the closest concept to a *Context*, e.g., *Context* "walking" refers to the concept "walk," and 2) preference matching to infer the most appropriate association inside a domain, e.g., an elderly (i.e., end-user) activity preference is walking rather than sleeping.

Algorithm 1 Intention Reduction

```
1:  Input: Semantic repository R, Intention graph $\Gamma$; Output:
     Assignments $\alpha$
2:  for each CtermIntention I in R do
3:     for all path$_x$ (paths of I) do ReduceIntention(path$_x$, I, $\phi$)
     end for
4:  end for

5:  Void Function ReduceIntention(currentPath,
     currentIntention, currentEnvironment)
6:  Add currentEnvironment to $\cdot c_{currentPath}$
7:  for all currentElement in currentPath do
8:     if currentElement.term_type == "condition" then
9:        for all entity between $\Gamma$.getAllNodes
          (currentIntention, currentElement) do
          Add node.properties to $\cdot c_{currentPath}$
          end for
10:        ReduceIntention(currentElement.IF_path,
          currentIntention, $\cdot c_{currentPath}$)
11:        ReduceIntention(currentElement.ELSE_path,
          currentIntention, $\cdot c_{currentPath}$)
12:     else if currentElement.term_type == "action" then
13:        for all entities in $\Gamma$.getAllNodes
          (currentIntention, currentElement) do
              Add R.getConcept(entity) to activityDomain
          end for
14:        $\alpha$.add(action, activityDomain, $c\cdot_{currentPath}$)
15:     else if currentElement.term_type == path then
16:        for each nextPath in currentElement do
          ReduceIntention(nextPath, currentIntention, $\phi$)
          end for
17:     end if
18:  end for
```

1) Concept matching: We apply semantic distance between *Contexts* [38] to match domain concepts ς with intention input/output *Context* $\cdot c$, $c\cdot$. Our implementation requires first to define a domain for activities. Then, a user creates applications linking the concepts from the domain. However, two concepts can have similar names (e.g., walk and watch), or users can introduce/select a similar *Context*, e.g., walking. We use ContextAA semantic distance [38] to select an appropriate Context as follows: A self-contained *Context $c = n\{v\}$* (read from left to right) is used to describe concepts and other AmI-DEU elements. The ContextAA semantic distance s is a function F: s $\rightarrow$ [0, 1] which represents the distance between two context c_1 and $c_2 \in C$ (i.e., all Context). Likewise, the action machine computes semantic, matching names (i.e., words) from left to right, calculating the semantic distance. For example, a domain expert user describes the activity preferences for elderly people, considering a range of preferences between 1 and 10. Thus, the user describes two concepts ς_1: walk {preference {9 }} and ς_2: watch tv {preference {3 }}. When creating an application, the user describes the intention to perform some activities, introducing an output *Context* $c\cdot_1$: walking {preference {5 } }. The action machine then applies semantic distance between ς_1 and $c\cdot_1$, and between ς_2 and $c\cdot_1$. Finally, it selects the target concept ς_1 as the concept with less distance (i.e., closest *Context*).

2) Preference matching: We use Fuzzy Logic reasoning for matching assignments with domain concept-relation. We consider a fuzzy relation reasonably expressed by domain experts, depending on their knowledge representation. Thus, we adopt fuzzy logic to enhance our action machine for selecting appropriate domain concepts based on weighted relations. We define a degree of truth as a map F $:\rho \rightarrow$ [0, 1] which represent the ρ weight w; w $\in$ fuzzy set weight W, $\rho \in \varsigma \times \varsigma\times$ r. We define three memberships for a relation (FIGURE 5-a):

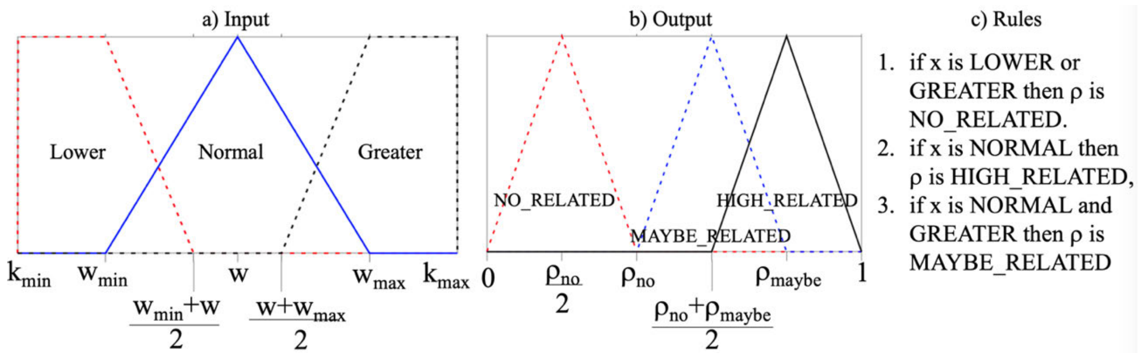

a) Input
Lower
Normal
Greater
k_{min} w_{min} w w_{max} k_{max}
$\frac{w_{min}+w}{2}$ $\frac{w+w_{max}}{2}$
b) Output
NO_RELATED
HIGH_RELATED
MAYBE_RELATED
0 $\frac{\rho_{no}}{2}$ ρ_{no} ρ_{maybe} 1
$\frac{\rho_{no}+\rho_{maybe}}{2}$
c) Rules
1. if x is LOWER or GREATER then ρ is NO_RELATED.
2. if x is NORMAL then ρ is HIGH_RELATED,
3. if x is NORMAL and GREATER then ρ is MAYBE_RELATED

FIGURE 5. Fuzzy (a) input, (b) output, and (c) rules.

Lower (2), Normal (1), and Greater (3). We use triangular and trapezoid membership functions at this stage because they are a basic membership representation for simple relations with a weight (e.g., high preference for walking, middle for running, lower for watching tv). The membership functions represent a relation with a target weight (i.e., normal value), which belongs to a fuzzy range (e.g., ranking of activities, a recommendation based on health conditions). We define a lower limit w_{min} and an upper limit w_{max} to represent the range of a weight $w \in W$, e.g., preference for walking is between 1 and 5, temperature preference is between 22 and 26 degrees Celsius. The selected membership functions also require extreme values. Thus, we define an extreme constant k_{min} and k_{max} to represent the extreme minimum and maximum weight values, respectively, e.g., the extreme minimum age is 0 years old, the extreme maximum age is 110 years old.

We define our output variable ρ_{out} for belonging to three truth ranges represented as triangular functions (FIGURE 5-b): NO_RELATED if $\rho_{out} \in [0, \rho_{no}]$, MAYBE_RELATED if $\rho_{out} \in [\rho_{no}, \rho_{maybe}]$, and HIGH_RELATED $\rho_{out} \in [\rho_{maybe}, 1]$; where the constants ρ_{no} and $\rho_{maybe} \in (0,1)$ and $\rho_{no} < \rho_{maybe}$. We design our rules (FIGURE 5-c) to consider "NORMAL" membership (i.e., expected relation weight) as the target value to activate the relation.

The design of the fuzzy logic inference engine considers the rules as local, i.e., each ρ only depends on the tuple (ς1, ς2, r), and not in other assumptions or complementary concepts/relations. Thus, we adopted the Mamdani implication algebraic product, which suits local rules [50]. The constants ρ_{no} and ρ_{maybe} represent the priority for relations, and as a result, allow discarding relations (e.g., when NO_RELATED). We also define a Threshold τ for activation of rules. Then, the action machine introduces a level of restrictions for fuzzy outputs when $\rho_{out} \geq \tau$ which enable to adjust the machine, i.e., ρ is only related when $\rho \in [\tau, 1]$.

Following, we present the evaluation of our model.

$$\mu_{\rho Normal}(x) = \begin{cases} \frac{x - w_{min}}{w - w_{min}}, & x \in [w_{min}, w] \\ \frac{x - w_{max}}{w - w_{max}}, & x \in [w, w_{max}] \\ 0, & otherwise \end{cases} \tag{1}$$

$$\mu_{\rho Lower}(x) = \begin{cases} 1, & x \in [k_{min}, w_{min}] \\ \frac{2x - w_{min} - w}{w_{min} - w}, & x \in \left[w_{min}, \frac{w_{min} + w}{2}\right] \\ 0, & otherwise \end{cases} \tag{2}$$

$$\mu_{\rho Greater}(x) = \begin{cases} \frac{2x - w_{max} - w}{w_{max} - w}, & x \in \left[\frac{w + w_{max}}{2}, w_{max}\right] \\ 1, & x \in [w_{max}, K_{max}] \\ 0, & otherwise \end{cases} \tag{3}$$

6) DEVELOPMENT PROCESS

We implemented an adapter to convert application definitions in JSON format to our semantic elements. We envisage the adapter for interoperability, e.g., with other development environments. The JSON format definition is Flow = (uid, name, type, properties, List_paths), where: **uid** is a unique element identification; **name** is the element name; **type** is the correspondence to a semantic element (i.e., concept | intention | entity | condition | action); **path** = (List_uid) is a set of element uids corresponding to the next elements in the flow of the current element.

Algorithm 2 shows the recursion implemented in the adapter to convert JSON flows to our semantic elements. Afterward, we applied Algorithm 3 to create the intention net. Our action machine then reduces the intentions, matches the activity domain with Fuzzy Lite – an open source fuzzy logic engine [51], and generates the assignments. Finally, we simulated an automatic deployment to a smart environment by loading the assignments into ContextAA for a subsequent mission generation. Then, ContextAA executes the missions.

Algorithm 2 JSON Parser

```
1:  **Input:** JSON Flow; **Output:** Intention repository R
2:  for each $path_x$ in Flow do
3:    if $path_x$.type == "concept" then
4:      Adapt $path_x$ to CTermConcept and Store in the repository R
5:    else if $path_x$.type == "intention"
6:      Adapt (for all $path_y$ (paths of $path_x$) do
          ParsePath ($path_y$)
          end for) to CTermIntention and Store in the repository R
7:      end if
8:    end if
9:  end for
10: return repository R
--------------------------------------------------
11: Function ParsePath(currentPath)
12: if currentPath. type == "entity" then
13:   Adapt (for all $path_y$ (paths of currentPath) do ParsePath
      ($path_y$) end for) to CTermEntity and Store in the
      repository R
14: else if currentPath. type == "condition" then
15:   Adapt (for all $path_y$ (paths of currentPath) do ParsePath
      ($path_y$) end for) to CTermCondition and Store in the
      repository R
16: else if currentPath. type == "action" then
17:   Adapt (for all $path_y$ (paths of currentPath) do ParsePath
      ($path_y$) end for) to CTermAction and Store in the
      repository R
18: end if
```

Algorithm 3 Intention Net Construction

```
1:  Input: Intentions from R; Output: Intention graph $\Gamma$
2:  for each $intention_x$ in Intention do
3:      for each $path_x$ in $intention_x$ do
        AddElementToGraph($path_x$, $intention_x$.ID)
        end for
4:  end for
5:  return $\Gamma$
--------------------------------------------------
6:  Function AddElementToGraph(currentPath,
    originOfThePath)
7:  for all currentElement in currentPath do
8:      If currentElement.type == "path" then
9:          for each $path_x$ in currentElement do
              AddElementToGraph($path_x$, currentElement.ID)
            end for
10:     else if currentElement.type == "entity" or
        currentElement.type == "action" then
11:         $\Gamma$.addEdge(originOfThePath,currentElement.ID)
12:     else if currentElement.type == "condition" then
13:         for all $path_{if}$ in currentElement.IF_path do
            AddElementToGraph($path_{if}$, currentElement.ID)
            end for
14:         for all $path_{else}$ in currentElement.ELSE_path do
            AddElementToGraph($path_{else}$, currentElement.ID)
        end for
15:     end if
16: end for
```

V. EVALUATION

We focus in this paper on the application descriptions (in diverse domains) and semantic compilation at development time (before deployment). The semantic compilation is measured in terms of Context size and semantic matching. We also evaluated IaaC's scalability for deployment in terms of *ContextAA* response time (as a reminder, our proposed approach is based on our novel IoT platform *ContextAA*. Previous runtime semantic matching performance of ContextAA has been evaluated in [11], [38], [39]). Following, we detail our evaluation before runtime and present empirical results in two scenarios: smart homes and smart cities.

A. SMART HOMES

We implemented a testbed to evaluate our approach through a real scenario of creating applications that include real sensor devices. Our testbed implementation combines open-source technologies to facilitate both gathering context from devices and simplifying end-user interactions.

1) TESTBED

We designed and implemented a complete infrastructure for performing tests (FIGURE 6). Our testbed infrastructure comprises three parts: 1) Development environment (FIGURE 6-a,b,c), 2) AmI-DEU/ContextAA implementation (FIGURE 6-d,e,f,g,h,i), and 3) Sensor/actuator platform (FIGURE 6-i,j).

2) DEVELOPMENT ENVIRONMENT

We used Node-red (https://nodered.org) to facilitate the composition of applications. Node-red GUI includes a composition area where users can drag and drop nodes and connect them, forming flows (FIGURE 6-b). We created nodes (FIGURE 6-a) to represent our semantic constructs (i.e., concept, relation, intention, entity, condition, action). We restricted our nodes to one-direction flows to simplify the development process. The flows start with intention nodes and end with action nodes. A user can connect nodes from two intentions through the flow, i.e., reuse activities and hierarchical composition. A user can also fork/join conditions and apply a condition to a different environment (adding an entity node before a condition). The development environment stores the application definitions in a Node-red JSON format (FIGURE 6-c) that we parse and load in AmI-DEU (Algorithm 2, FIGURE 6-d). Then, we create the intention net (Algorithm 3, FIGURE 6-e) and process the app semantics (FIGURE 6-f) to generate IaaC assignments (FIGURE 6-g). Finally, we simulated an automatic deployment to a smart environment by loading the assignments into ContextAA for a subsequent mission generation (FIGURE 6-h). Then, ContextAA executes the missions (FIGURE 6-i).

3) SENSOR/ACTUATOR PLATFORM

We developed DomoSense, our home automation platform, to interact with the environment (FIGURE 6-j). DomoSense includes a set of functionalities to connect diverse sensing technologies. We connected Z-Wave devices to gather the indoor environment context, simulating a smart house.

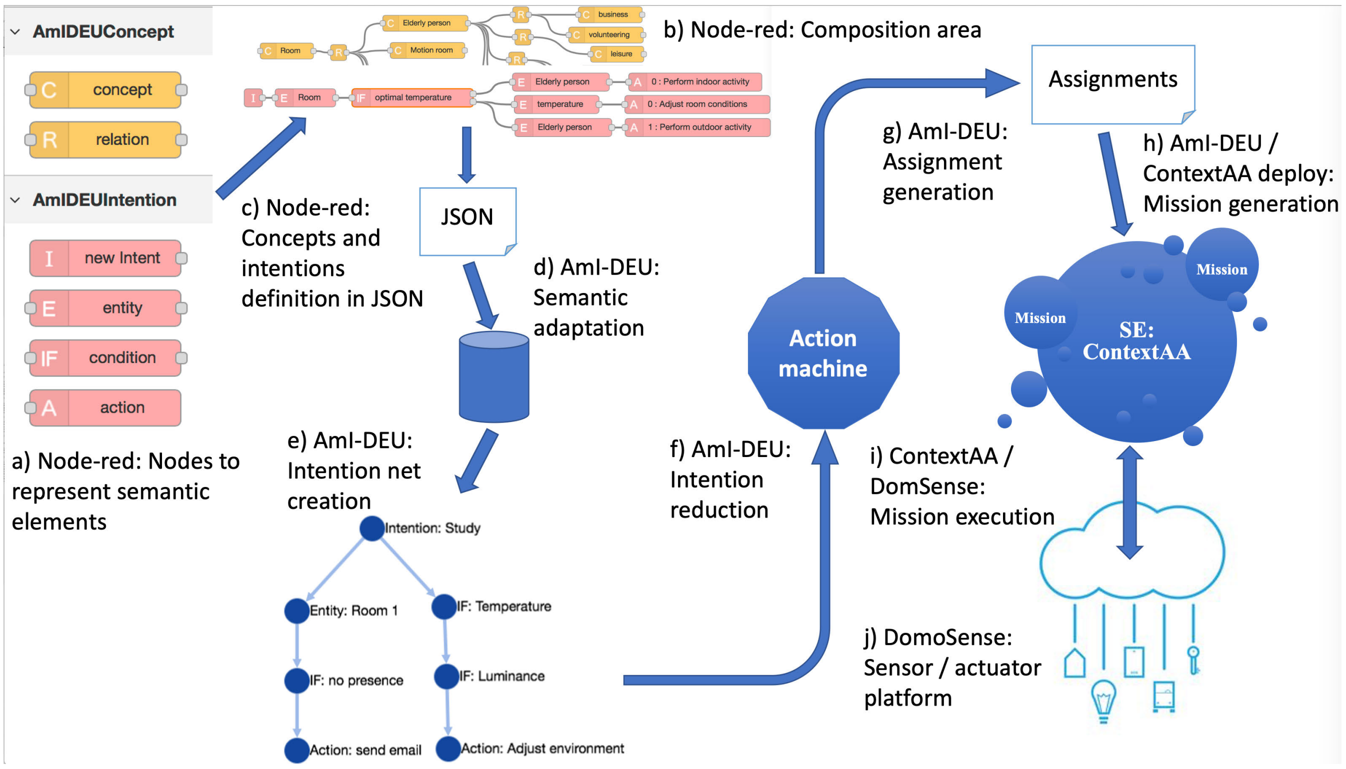

AmIDEUConcept
C concept
R relation
AmIDEUIntention
I new Intent
E entity
IF condition
A action
a) Node-red: Nodes to represent semantic elements
C Room
R
C Elderly person
C Motion room
R
C business
C volunteering
C leisure
I
E Room
IF optimal temperature
E Elderly person
A 0 : Perform indoor activity
E temperature
A 0 : Adjust room conditions
E Elderly person
A 1 : Perform outdoor activity
b) Node-red: Composition area
c) Node-red: Concepts and intentions definition in JSON
JSON
d) AmI-DEU: Semantic adaptation
e) AmI-DEU: Intention net creation
Intention: Study
Entity: Room 1
IF: Temperature
IF: no presence
IF: Luminance
Action: send email
Action: Adjust environment
f) AmI-DEU: Intention reduction
Action machine
g) AmI-DEU: Assignment generation
Assignments
h) AmI-DEU / ContextAA deploy: Mission generation
Mission
Mission
SE: ContextAA
i) ContextAA / DomSense: Mission execution
j) DomoSense: Sensor / actuator platform

FIGURE 6. Testbed infrastructure.

TABLE 4. Use cases.

ID	Use case	Description
UC-1	Presence in a room (FIGURE 7-2)	The aim is to determine whether a person intends to do some activity in a room, e.g., is an elderly person in the kitchen? (FIGURE 7-2a)
UC-2	Environment conditions (FIGURE 7-1)	The aim is to maintain in-person environment conditions for doing an activity, e.g., suitable needs for cooking (FIGURE 7-1a,1b)
UC-3	Keep doing an activity (FIGURE 7-1,2)	The aim is to combine the previous intention to define a third one, which maintains in-person conditions, but alert/adjust when conditions change or stop the activity. E.g., monitoring an elderly person in the kitchen, alerting/adjusting temperature if the temperature increases (FIGURE 7-1c), or the person left the kitchen (FIGURE 7-2b)
UC-4	Activity recommendation (FIGURE 7-3).	The aim is to recommend the most appropriate activity based on end-user profile and preferences, e.g., recommend indoor activities if optimal room temperature (FIGURE 7-3a); otherwise, adjust room conditions (FIGURE 7-3b) and recommend outdoor activity (FIGURE 7-3c)

These devices enable ContextAA to receive context related to temperature, humidity, motion, door-window open/close, switches, doorbell, among others. Then, ContextAA evaluates the rules in deployed missions, matching them with changing context to execute the actions (FIGURE 6-i).

4) SCENARIO IMPLEMENTATION

We created a scenario to perform tests simulating a domain expert defining a domain for an ''elderly person'' living in a smart house. As a first step, we defined four use cases (Table 4). The first use case corresponds to an intention with a simple activity that includes a simple action triggered by a condition. The second use case corresponds to an intention with a complex activity that validates multiple conditions and executes multiple actions. The third use case corresponds to two interrelated intentions. The fourth use case evaluates hedonic activity intentions.

As a second step, we use the testbed to create applications. We use Node-red to add elements, modify attributes, and connect elements of domain and intentions.

Table 5 shows an example of the concept-relation description we created for the evaluation (with a relation weight between 1 and 10). FIGURE 7 shows the Node-red composition area, with flow examples for the use cases for domain description (FIGURE 7-a) and intentions description (FIGURE 7-b).

TABLE 5. Example of concepts and relations for the scenario.

Concept 1	Relation	Relation weight [1-10]	Concept 2
Room	Contains	10 (e.g., always)	Motion, contact, temperature
	Contains	5 (e.g., sometimes)	Elderly person
Elderly person	Prefer activity	8 (e.g., high)	Business, volunteering
	Prefer activity	7 (e.g., often)	Leisure
	Prefer activity	10 (e.g., highest)	Walking, jogging, reading
	Prefer activity	2 (e.g., low)	Sleeping, watch tv

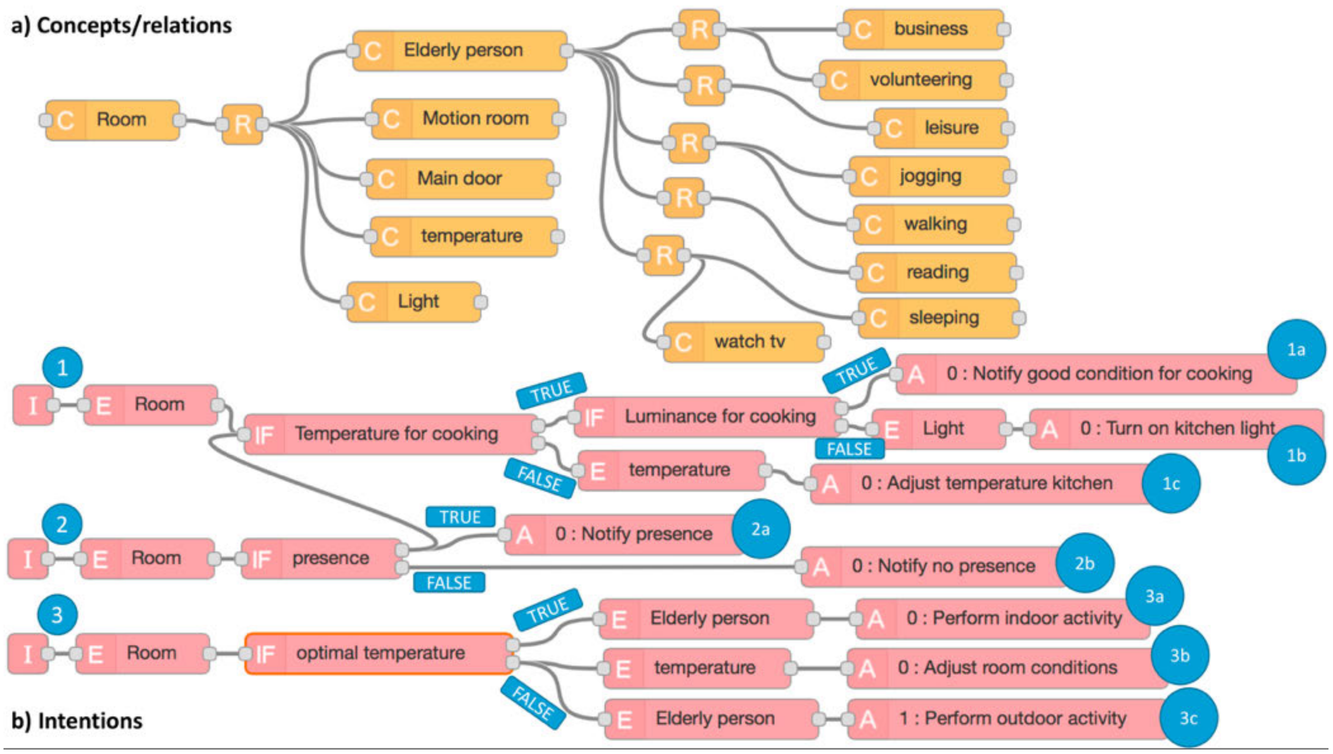


FIGURE 7. Examples of the node-red composition of activity domain and intentions.

5) RESULTS

After the composition of applications for the use cases, the system generates semantic elements using Algorithm 2. Then, it creates the intention net using the implementation of Algorithm 3. Afterward, the system processes the semantic elements through the action machine using Algorithm 1 and domain matching. We performed two tests for the use cases: computation without domain matching and with domain matching. We maintain the same domain (14 concepts with their relations) to compare the generated/processed data for the use cases. Table 6 lists the number of generated semantic elements and assignments, the rate of knowledge compilation with domain matching, and the reduction rate when deployed. These rates depend on the complexity of the application and represent the resource optimization for pervasive IoT apps. The rate of generated *Context* for knowledge compilation tends to decrease matching the domain (FIGURE 8). On the other hand, the rate of generated *Context* when deployed tends to decrease when the size of the *Context* to deploy is bigger than a threshold size, i.e., a minimal size to include a set of keywords of the ContextAA platform processing language, e.g., "eval" {expression}.

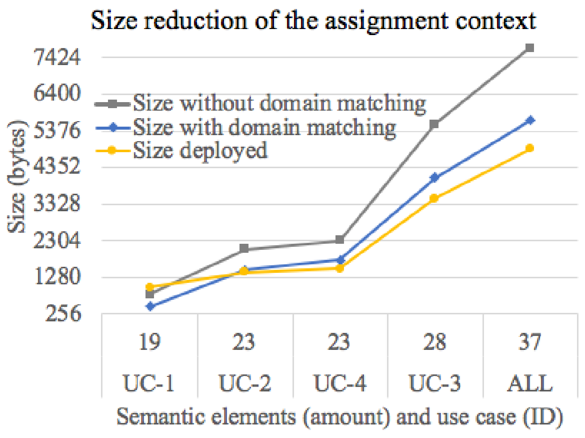


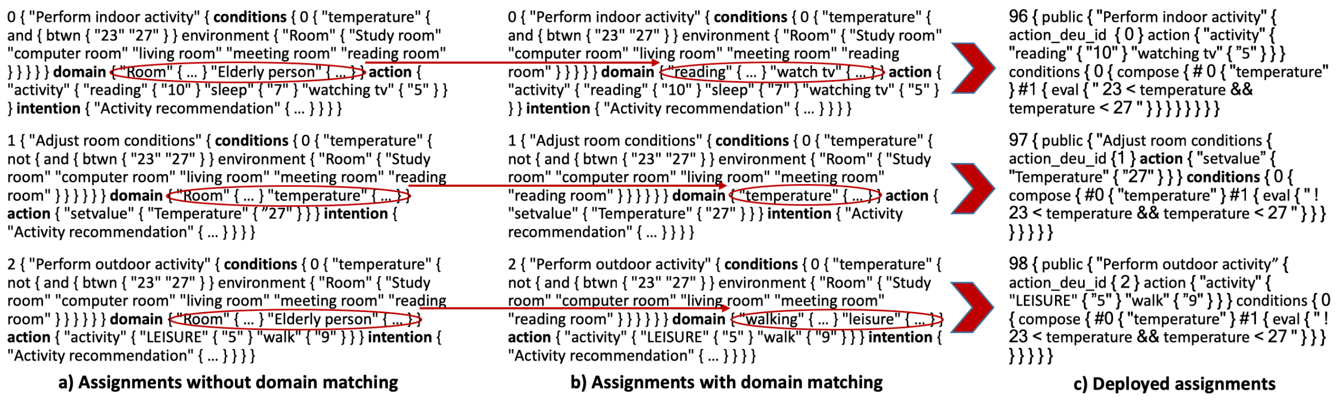
FIGURE 8. Action machine compilation.

TABLE 6. Use cases test data.

Use case	#Intentions	#Concepts	#Entities / #Conditions / #Actions	Total semantic elements	#Assignments	%Compiled knowledge	%Reduction (deployed)
UC-1	1	14	1 / 1 / 2	19	2	- 40.90	+ 111.89
UC-4	1	14	4 / 1 / 3	23	3	- 23.41	- 4.77
UC-2	1	14	3 / 2 / 3	23	3	- 28.09	- 13.10
UC-3	2	14	4 / 3 / 5	28	8	- 26.78	- 14.22
ALL	3	14	8 / 4 / 8	37	11	- 26.37	- 14.05

TABLE 7. Similar semantic elements evaluation.

Application semantic element	Domain concept	Semantic distance (S)
reading	reading	0
sleep	sleeping	0.0218
watching tv	watch tv	0.0168
watching tv	walking	0.1531
Temperature	temperature	0
LEISURE	leisure	0
walk	watch tv	0.1620
walk	walking	0.0498
kitchen light	Light	0.5725
temperature	temperature	0



```
0 { "Perform indoor activity" { conditions { 0 { "temperature" {
and { btwn { "23" "27" } } environment { "Room" { "Study room"
"computer room" "living room" "meeting room" "reading room"
} } } } } domain { "Room" { … } "Elderly person" { … } } action {
"activity" { "reading" { "10" } "sleep" { "7" } "watching tv" { "5" } }
} intention { "Activity recommendation" { … } } } }

1 { "Adjust room conditions" { conditions { 0 { "temperature" {
not { and { btwn { "23" "27" } } environment { "Room" { "Study
room" "computer room" "living room" "meeting room" "reading
room" } } } } } } domain { "Room" { … } "temperature" { … } }
action { "setvalue" { "Temperature" { "27" } } } intention {
"Activity recommendation" { … } } } }

2 { "Perform outdoor activity" { conditions { 0 { "temperature" {
not { and { btwn { "23" "27" } } environment { "Room" { "Study
room" "computer room" "living room" "meeting room" "reading
room" } } } } } } domain { "Room" { … } "Elderly person" { … } }
action { "activity" { "LEISURE" { "5" } "walk" { "9" } } } intention {
"Activity recommendation" { … } } } }
```

a) Assignments without domain matching

```
0 { "Perform indoor activity" { conditions { 0 { "temperature" {
and { btwn { "23" "27" } } environment { "Room" { "Study room"
"computer room" "living room" "meeting room" "reading
room" } } } } } domain { "reading" { … } "watch tv" { … } } action {
"activity" { "reading" { "10" } "sleep" { "7" } "watching tv" { "5" }
} } intention { "Activity recommendation" { … } } } }

1 { "Adjust room conditions" { conditions { 0 { "temperature" {
not { and { btwn { "23" "27" } } environment { "Room" { "Study
room" "computer room" "living room" "meeting room"
"reading room" } } } } } } domain { "temperature" { … } } action {
"setvalue" { "Temperature" { "27" } } } intention { "Activity
recommendation" { … } } } }

2 { "Perform outdoor activity" { conditions { 0 { "temperature" {
not { and { btwn { "23" "27" } } environment { "Room" { "Study
room" "computer room" "living room" "meeting room"
"reading room" } } } } } } domain { "walking" { … } "leisure" { … } }
action { "activity" { "LEISURE" { "5" } "walk" { "9" } } } intention {
"Activity recommendation" { … } } } }
```

b) Assignments with domain matching

```
96 { public { "Perform indoor activity" {
action_deu_id  { 0 } action { "activity" {
"reading" { "10" } "watching tv" { "5" } } }
conditions { 0 { compose { # 0 { "temperature"
} #1 { eval { " 23 < temperature &&
temperature < 27 " } } } } } } } }

97 { public { "Adjust room conditions {
action_deu_id {1 } action { "setvalue" {
"Temperature" { "27" } } } conditions { 0 {
compose { #0 { "temperature" } #1 { eval { " !
23 < temperature && temperature < 27 " } } }
} } } } }

98 { public { "Perform outdoor activity" {
action_deu_id { 2 } action { "activity" {
"LEISURE" { "5" } "walk" { "9" } } } conditions { 0
{ compose { #0 { "temperature" } #1 { eval { " !
23 < temperature && temperature < 27 " } } }
} } } } }
```

c) Deployed assignments

FIGURE 9. Example of assignments for UC-4.

The action machine's basic semantic processing is walking through the Intention Net to generate self-contained assignments, e.g., UC-3 produces eight assignments interrelating two intentions. The subsequent semantic processing matches the *Context* of the applications with the *Context* of the domain. The current evaluation implements the semantic distance S between names (i.e., a string representing the *Context*/concept name). We evaluate each name to compare the matching (Table 7), defining close *Contexts* when $S < 0.1$. For example, S(walk, walking) $= 0.0498$, then the strings are close. S(kitchen light, Light) $= 0.5725$; then, the strings are distant; therefore, they are different and discarded.

The final semantic processing of the action machine is matching the domain relations through Fuzzy logic. We use UC-4 to test domain relations matching. FIGURE 9 shows the IaaC assignments generated by the action machine without domain matching (FIGURE 9-a), with domain matching (FIGURE 9-b), and IaaCs deployed in the ContextAA IoT platform (FIGURE 9-c). We configured the Fuzzy logic engine to generate outputs with an activation threshold $>= 0.5$ for relations of the type HIGH_RELATION and MAYBE_RELATION. Table 8 lists the reasoning results for UC-4. The results show that the context size decreases, e.g., discarding "sleep," as well as the preferences for activities is reordered, e.g., from LEISURE (5), walk (9) to walking (0.889), leisure (0.667) (which does not affect the deployment because preference walk $>$ leisure). We can also apply the activation threshold to suggest or improve intention/action configurations such as priority and order.

B. SMART CITIES

We also implemented deployment functionalities for our testbed (FIGURE 6-h). We evaluated our approach's deployment scalability for smart cities through the InterSCity

TABLE 8. Fuzzy engine reasoning results for UC-4.

Intention (FIGURE 9-a): [element { target }]	Domain (Table V): [concept { weight }]	Reasoning result (FIGURE 9-b): OUTPUT: degree
reading {10} sleep { 7 } watching tv {5}	reading {10} sleeping {2} watch tv {2}	HIGH_RELATION_reading: 1 HIGH_RELATION_watch tv: 0.625
Temperature {27}	temperature {10}	HIGH_RELATION_temperature: 0.571
LEISURE {5} walk {9}	leisure {7} walking {10}	HIGH_RELATION_walking: 0.889 HIGH_RELATION_leisure: 0.667

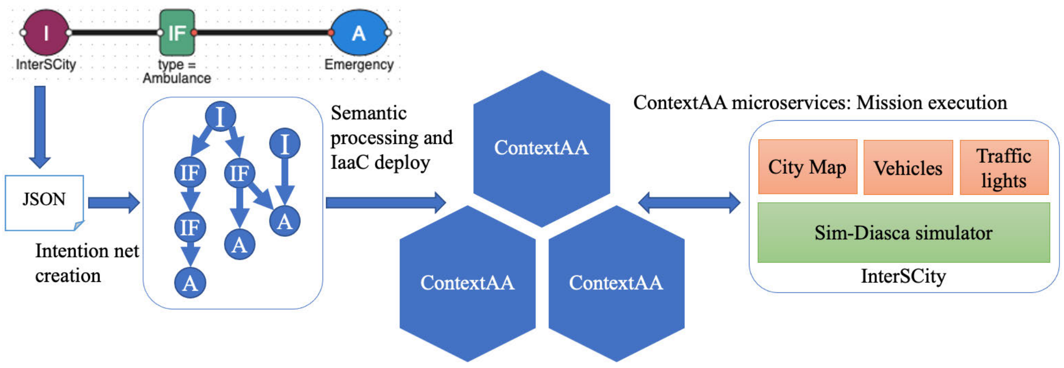


FIGURE 10. Smart city simulation.

platform and InterSCSimulator [52]–[54]. InterSCity (platform and simulator) is an open-source microservices-based middleware to simulate smart city applications. InterSCity provides facilities to manage, store, and process heterogeneous IoT services and resources.

1) INTERSCITY

InterSCity is built on top of the Sim-Diasca simulator (FIGURE 10), which executes simulation processes such as deployment and time management and simulation result collection. Both are implemented in Erlang (based on the actor model) with Wooper (a wrapper for object-oriented programming in Erlang).

InterSCity receives config files representing smart cities' resources and configurations such as city maps, traffic lights, trips, etc. It provides mobility models for vehicles and other city actors (e.g., pedestrians). Actors execute tasks during a simulation clock tick, representing the simulation of one second in the real world [52]. We adapted InterSCity to interact with our IoT platform ContextAA via MQTT and REST API.

2) SCENARIO IMPLEMENTATION

We extended the InterSCity traffic lights scenario to simulate vehicle emergencies in an area of Sherbrooke city, Canada. We consider two types of vehicles: cars and ambulances. A car represents a vehicle moving in normal conditions, i.e., respecting traffic lights. An ambulance represents a vehicle that needs to move faster from origin (e.g., home) to destination (e.g., hospital). We created an application to represent the emergency response for ambulance proximity to a red traffic light, changing it to green, i.e., **IF car_type is an ambulance, THEN turn traffic_light to green.** Each traffic light keeps an IaaC, and we deployed the IaaC to multiple ContextAA docker containers connected to InterSCity (FIGURE 10). Results

The process of composing and deploying the app is the same as described previously for smart homes. Furthermore, the testbed facilitates the deployment to multiple containers for testing. We simulated trips for 162 cars and five ambulances (emergencies). The vehicles travel different distances, respecting traffic lights with homogeneous phases of 130 ticks (i.e., simulation unit of time). We consider only green and red lights with 30 to 60 seconds in green based on the number of crossing streets. We simulated ten different traffic lights through the city deployed on each container (FIGURE 11). The results show that Ambulance 1 and 2 travel time is the same because they were close to the hospital and had no traffic lights. However, Ambulance 3, 4, and 5 trips are adapted by the IaaC deployed. Specifically, Ambulance 5, which is traveling farther distance (from 6363 m), arrives earlier (in 568 ticks) when the IaaC is deployed on all 30 traffic lights (FIGURE 12).

VI. DISCUSSION AND CHALLENGES

We aim to augment activity representation in the current IoT era when smart devices enable higher automation of everyday activities. Our research and experimental results brought challenges to address, in particular, to describe complex intentions such as habits or events. Following, we discuss key aspects of our approach and future research.

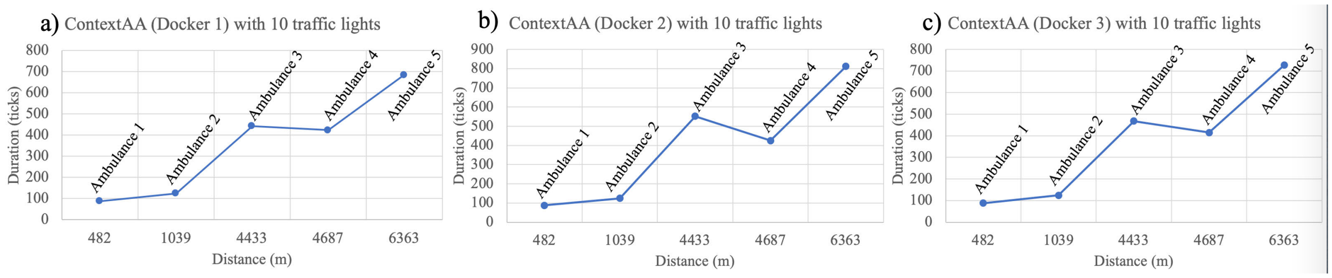


FIGURE 11. Simulation with three containers, each one with different traffic lights through the city.

Distance	Duration (ticks)			
(m)	Docker 1	Docker 2	Docker 3	Docker 1+2+3
4433	442	552	468	442
4687	425	425	415	415
6363	685	812	728	586

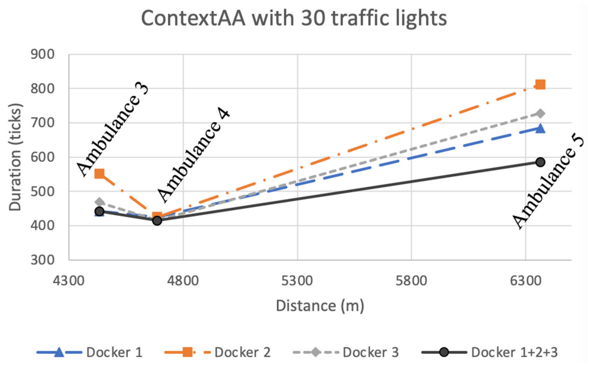


FIGURE 12. Simulation with 30 traffic lights.

A. ACTIVITY INTENTION ELICITATION IN PERVASIVE COMPUTING APPLICATIONS

Traditional applications restrict end-user intentions because they are designed for a general end-user profile and developed considering only specific actions. For example, they restrict activities based only on an ontology-based model with atomic actions such as "obtain milk vessel," "pour milk" [55]. Similarly, new solutions (e.g., cloud-based smart homes) and intelligent assistants (e.g., Google Assistant, Amazon Alexa) introduce an improved interaction (e.g., voice, widgets) [56]. However, these solutions restrict end-user intentions to fixed environment actions (e.g., Google Actions, Alexa skills at home). With the flooding of IoT smart devices, defining applications is tedious, i.e., more actions/skills to configure. In our approach, the actions are not specific and depend on the activity domain.

Our approach abstracts any context (e.g., environments, concepts, actions) and describes them in terms of ContextAA *Context*-based semantic elements. Adopting our approach, domain experts can generate dynamic applications that use a common domain (e.g., smart home shown in Table 5). The applications can be interrelated, reusing the abstracted context based on end-user needs (e.g., smart home activities shown in Table 4). The action machine processes the semantic elements and matches the context to generate IaaC assignments. The generated assignments in our approach contain compiled knowledge that can augment applications' achievement, decreasing the dependence on fixed environments. However, it is a challenge to process dynamic semantics (i.e., the same application in a new environment) and, at the same time, improve the semantic matching (closer to 100%). Future research can apply another semantic compilation level using machine learning, e.g., with pre-trained sentence encoders [57] or incremental learning [34].

B. ENHANCE THE INTENTION DESCRIPTION

We include the semantic element Concept to describe the activity domain. We discard complex relations and axioms to describe domain phenomena because we attempt to simplify domain experts' descriptions. At this stage of our work, the semantic element Concept allows describing concepts with part-whole and weighted associations, which are adequate to build simple everyday activity applications. We also include the semantic elements representing entities and common operations, the statements to create rules, and the required configurations (e.g., priority, order) to create action sequences. With the action machine semantic processing, we can improve intention/action configurations to automatize action sequences.

Our approach also enables us to compose hierarchical intentions (FIGURE 2) (e.g., activity plans) using rules' composition/aggregation. However, we identified the need for additional domain descriptions for complex intentions such as end-user roles and temporal constraints between concepts (e.g., preference for outdoor activity during a day). We also identified the need for a simplified UI to help non-technical people compose IoT applications.

C. COMPLEX INTENTIONS

By reducing intentions, we prepare them for running in IoT smart environments. This reduction is essential because the semantic elements allow flexibility to promote domain experts' involvement. For example, a domain expert can describe the intention to turn on a light and the intention to go to the bathroom. Each intention has a goal; however, both can be: 1) described independently (two independent intentions, each one with each context/conditions); or 2) described together (two intentions, but interrelating context/conditions). This situation creates different paths for the intentions, which are the input for the action machine. The action machine processes intentions and creates atomic assignments, i.e., self-described context with logic and compiled knowledge.

We are exploring the complexity of applications and how to compile complex intentions. Complex intentions require additional descriptions such as temporal constraints or regulations. One option is to create activity templates that domain experts can reuse. Then, we can improve the compilation with pre-compiled application templates.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed an approach to empower domain experts who apply their knowledge and new technologies to assist people. Our approach comprises an intention model to describe the activity intention as a context (IaaC)—an anticipated outcome of the activity represented as a self-described context. The model is part of our context-aware framework AmI-DEU which provides the semantic elements to simplify creating meaningful applications for our ContextAA IoT smart environment platform. In the first step, domain experts describe the activity domain using semantic elements to represent concepts and relations. Afterward, diverse semantic elements enable the description of activity intention flows. Finally, our action machine reduces and matches the flows, generating IaaC applications with simplified scripts (i.e., IaaC assignments). IaaC applications are reduced in size but contain enriched context with the conditions, actions, and the assignments' compiled knowledge. The enriched context provides relevant information to enhance the assessment and adaptation of the pervasive IoT smart environment to support end-user activities.

The results of applying our approach have shown that the action machine (1) decreases the context size matching intention flows and activity domain, which reduce the implementation/interpretation time; and (2) the semantic processing can improve the activity intention (e.g., ordering activities by preference, discarding unnecessary context) which augment the available features to enhance system actions.

We plan to add domain descriptions (e.g., temporal constraints) to elicit complex intentions. We will then improve the action machine semantic matching of sentences and other patterns (e.g., events, behaviors). We are also developing a UI representing our hierarchical model, with activity templates, for making our framework useful/usable to domain experts.

REFERENCES

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[2] A. Ghasempour, "Internet of Things in smart grid: Architecture, applications, services, key technologies, and challenges," *Inventions*, vol. 4, no. 1, p. 22, 2019.

[3] L. Yao, Q. Z. Sheng, B. Benatallah, S. Dustdar, X. Wang, A. Shemshadi, and S. S. Kanhere, "WITS: An IoT-endowed computational framework for activity recognition in personalized smart Homes," *Computing*, vol. 100, no. 4, pp. 369–385, Apr. 2018.

[4] J. K. Kim and Y. B. Kim, "Joint learning of domain classification and out-of-domain detection with dynamic class weighting for satisficing false acceptance rates," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, Sep. 2018, pp. 556–560.

[5] L. Farhan, A. E. Alissa, S. T. Shukur, M. Alrweg, U. Raza, and R. Kharel, "A survey on the challenges and opportunities of the Internet of Things (IoT)," in *Proc. 11th Int. Conf. Sens. Technol. (ICST)*, Dec. 2017, pp. 1–5.

[6] V. Ponce, J. P. Deschamps, L. P. Giroux, F. Salehi, and B. Abdulrazak, "QueFaire: Context-aware in-person social activity recommendation system for active aging," in *Proc. Int. Conf. Smart Homes Health Telematics*, 2015, pp. 64–75.

[7] B. Abdulrazak, P. Roy, C. Gouin-Vallerand, Y. Belala, and S. Giroux, "Micro context-awareness for autonomic pervasive computing," *Int. J. Bus. Data Commun. Netw.*, vol. 7, no. 2, pp. 48–68, Apr. 2011.

[8] S. Thill, D. Caligiore, A. M. Borghi, T. Ziemke, and G. Baldassarre, "Theories and computational models of affordance and mirror systems: An integrative review," *Neurosci. Biobehav. Rev.*, vol. 37, no. 3, pp. 491–521, Mar. 2013.

[9] C. Stephanidis, G. Salvendy, and J. Y. C. Chen, "Seven HCI grand challenges," *Int. J. Hum.–Comput. Interact.*, vol. 35, no. 14, pp. 1229–1269, 2019.

[10] M. Bystricky and V. Vranic, "Preserving use case flows in source code," in *Proc. 4th Eastern Eur. Regional Conf. Eng. Comput. Based Syst.*, Aug. 2015, pp. 9–16.

[11] V. Ponce, P. Roy, and B. Abdulrazak, "Dynamic domain model for micro context-aware adaptation of applications," in *Proc. Int. IEEE Conferences Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 98–105.

[12] M. M. Burnett and B. A. Myers, "Future of end-user software engineering: Beyond the silos," in *Proc. Future Softw. Eng.*, 2014, pp. 201–211.

[13] G. Sukthankar, C. Geib, H. H. Bui, D. Pynadath, and R. P. Goldman, *Plan, Activity, and Intent Recognition: Theory and Practice*. London, U.K.: Newnes, 2014.

[14] J. Lindblom and B. Alenljung, "The ANEMONE: Theoretical foundations for UX evaluation of action and intention recognition in human-robot interaction," *Sensors*, vol. 20, no. 15, p. 4284, 2020.

[15] A. Herzig, E. Lorini, L. Perrussel, and Z. Xiao, "BDI logics for BDI architectures: Old problems, new perspectives," *KI-Künstliche Intelligenz*, vol. 31, no. 1, pp. 73–83, Mar. 2017.

[16] C. K. Chang, H. Y. Jiang, H. Ming, and K. Oyama, "Situ: A situation-theoretic approach to context-aware service evolution," *IEEE Trans. Services Comput.*, vol. 2, no. 3, pp. 261–275, Jul. 2009.

[17] S. Liu, S. Helal, and J. W. Lee, "High fidelity simulation and visualization of activities of daily living in persim 3D," in *Proc. Int. Conf. Smart Homes Health Telematics*, vol. 10461, 2017, pp. 136–148.

[18] A. Aztiria, J. C. Augusto, R. Basagoiti, A. Izaguirre, and D. Cook, "Learning frequent behaviors of the users in intelligent environments," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 6, pp. 1265–1278, Nov. 2013.

[19] C. A. Kamienski, F. F. Borelli, G. O. Biondi, I. Pinheiro, I. D. Zyrianoff, and M. Jentsch, "Context design and tracking for IoT-based energy management in smart cities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 687–695, Apr. 2018.

[20] L. Chen, C. D. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart Homes," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 961–974, Jun. 2012.

[21] S. Zhang, P. McCullagh, C. Nugent, H. Zheng, and N. Black, "An ontological framework for activity monitoring and reminder reasoning in an assisted environment," *J. Ambient Intell. Humanized Comput.*, vol. 4, no. 2, pp. 157–168, 2013.

[22] U. Naeem, J. Bigham, and J. Wang, "Recognising activities of daily life using hierarchical plans," in *Smart Sensing and Context*. Berlin, Germany: Springer, 2007, pp. 175–189.

[23] A. Kojima, T. Tamura, and K. Fukunaga, "Natural language description of human activities from video images based on concept hierarchy of actions," *Int. J. Comput. Vis.*, vol. 50, no. 2, pp. 171–184, 2002.

[24] F. Gu, K. Khoshelham, S. Valaee, J. Shang, and R. Zhang, "Locomotion activity recognition using stacked denoising autoencoders," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2085–2093, Jun. 2018.

[25] A. Benmansour, A. Bouchachia, and M. Feham, "Multioccupant activity recognition in pervasive smart home environments," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 1–36, Feb. 2016.

[26] K. C. Hsu, Y. T. Chiang, G. Y. Lin, C. H. Lu, J. Y. J. Hsu, and L. C. Fu, "Strategies for inference mechanism of conditional random fields for multiple-resident activity recognition in a smart home," in *Proc. Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst.*, vol. 6096, 2010, pp. 417–426.

[27] B. Minor and D. J. Cook, "Forecasting occurrences of activities," *Pervas. Mobile Comput.*, vol. 38, pp. 77–91, Jul. 2017.

[28] Y. Gu, F. Ren, and J. Li, "PAWS: Passive human activity recognition based on WiFi ambient signals," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 796–805, Oct. 2016.

[29] Z. Zhu, U. Blanke, and G. Tröster, "Recognizing composite daily activities from crowd-labelled social media data," *Pervas. Mobile Comput.*, vol. 26, pp. 103–120, Feb. 2016.

[30] K. Häussermann, C. Hubig, P. Levi, and F. Leymann, "Understanding and designing situation-aware mobile and ubiquitous computing systems," *World Acad. Sci. Eng. Technol.*, vol. 4, no. 39, pp. 972–981, 2010.

[31] C.-H. Lu and L.-C. Fu, "Enhancement of human-preference assisted activity recognition using a cooperative ADL infrastructure," in *Proc. 9th Int. Conf. Ubiquitous Intell. Comput. 9th Int. Conf. Autonomic Trusted Comput.*, Sep. 2012, pp. 1009–1014.

[32] N. P. Cuntoor, B. Yegnanarayana, and R. Chellappa, "Activity modeling using event probability sequences," *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 594–607, Apr. 2008.

[33] S. Wang, W. Pentney, A. Popescu, T. Choudhury, and M. Philipose, "Common sense based joint training of human activity recognizers," in *Proc. IJCAI*, 2007, pp. 2237–2242.

[34] D. Tao, Y. Wen, and R. Hong, "Multicolumn bidirectional long short-term memory for mobile devices-based human activity recognition," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1124–1134, Dec. 2016.

[35] D. Kulkarni and A. Tripathi, "A framework for programming robust context-aware applications," *IEEE Trans. Softw. Eng.*, vol. 36, no. 2, pp. 184–197, Mar. 2010.

[36] P. Rashidi and D. J. Cook, "Keeping the resident in the loop: Adapting the smart home to the user," *IEEE Trans. Syst., Man, Cybern., A, Syst. Humans*, vol. 39, no. 5, pp. 949–959, Sep. 2009.

[37] V. Tietz, A. Rümpel, M. Voigt, P. Siekmann, and K. Meißner, "Tool support for semantic task modeling," in *Proc. 3rd Int. Conf. Web Intell., Mining Semantics (WIMS)*, 2013, pp. 1–12.

[38] P. Roy, B. Abdulrazak, and Y. Belala, "Quantifying semantic proximity between contexts," in *Proc. Int. Conf. Smart Homes Health Telematics*, 2014, pp. 165–174.

[39] V. Ponce and B. Abdulrazak, "Activity model for interactive micro context-aware well-being applications based on ContextAA," in *Proc. Int. Conf. Smart Homes Health Telematics*, 2017, pp. 99–111.

[40] E. Kim, S. Helal, and D. Cook, "Human activity recognition and pattern discovery," *IEEE Pervasive Comput.*, vol. 9, no. 1, pp. 48–53, Jan./Mar. 2010.

[41] Y. Chen, J. Wang, M. Huang, and H. Yu, "Cross-position activity recognition with stratified transfer learning," *Pervas. Mobile Comput.*, vol. 57, pp. 1–13, Jul. 2019.

[42] J. Ye, S. Dasiopoulou, G. Stevenson, G. Meditskos, and E. Kontopoulos, "Semantic web technologies in pervasive computing: A survey and research roadmap," *Pervasive Mobile Comput.*, vol. 23, pp. 543–549, Oct. 2015.

[43] M. Noura, S. Heil, and M. Gaedke, "GROWTH: Goal-oriented end user development for web of things devices," in *Proc. Int. Conf. Web Eng.*, vol. 10845, 2018, pp. 358–365.

[44] V. Realinho, T. Romao, and A. E. Dias, "A language for the end-user development of mobile context-aware applications," *J. Wireless Mobile Netw., Ubiquitous Comput. Dependable Appl.*, vol. 11, no. 1, pp. 54–80, 2020.

[45] Y. Li and J. A. Landay, "Activity-based prototyping of ubicomp applications for long-lived, everyday human activities," in *Proc. 26th Annu. Conf. Hum. Factors Comput. Syst. (CHI)*, 2008, pp. 1303–1312.

[46] H. Huang and G. Gartner, "Using activity theory to identify relevant context parameters," in *Location Based Services TeleCartography II*. Berlin, Germany: Springer, 2009, pp. 35–45.

[47] P. Roy, B. Abdulrazak, and Y. Belala, "A distributed architecture for micro context-aware agents," *Proc. Comput. Sci.*, vol. 5, pp. 296–303, Feb. 2011.

[48] P. Roy, "ContextAA: Plateforme sensible Au Contexte pour aborder le problème de l'espace intelligent ouvert," M.S. thesis, Univ. Shrerbrooke, Sherbrooke, QC, Canada, 2019. [Online]. Available: http://hdl.handle.net/11143/15817

[49] D. Bjorner, *Software Engineering 3: Domains, Requirements, and Software Design* (Texts in Theoretical Computer Science. An EATCS Series). Secaucus, NJ, USA: Springer-Verlag, 2006.

[50] A. Konar, *Computational Intelligence: Principles, Techniques and Applications*. Berlin, Germany: Springer, 2006.

[51] J. Rada-Vilela. (2018). *The FuzzyLite Libraries for Fuzzy Logic Control*. [Online]. Available: https://fuzzylite.com/

[52] A. de M. Del Esposte, E. F. Z. Santana, L. Kanashiro, F. M. Costa, K. R. Braghetto, N. Lago, and F. Kon, "Design and evaluation of a scalable smart city software platform with large-scale simulations," *Future Gener. Comput. Syst.*, vol. 93, pp. 427–441, Apr. 2019.

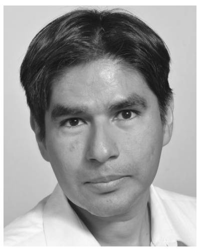
[53] A. M. Del Esposte, F. Kon, F. M. Costa, and N. Lago, "InterSCity: A scalable microservice-based open source platform for smart cities," in *Proc. SMARTGREENS*, vol. 1, 2017, pp. 35–46.

[54] E. F. Z. Santana, N. Lago, F. Kon, and D. S. Milojicic, "Interscsimulator: Large-scale traffic simulation in smart cities using Erlang," in *Proc. Int. Workshop Multi-Agent Syst. Agent-Based Simulation*, 2017, pp. 211–227.

[55] J. Rafferty, C. D. Nugent, J. Liu, and L. Chen, "From activity recognition to intention recognition for assisted living within smart Homes," *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 3, pp. 368–379, Jun. 2017.

[56] J. Kiseleva, K. Williams, J. Jiang, A. H. Awadallah, and A. C. Crook, "Understanding user satisfaction with intelligent assistants," in *Proc. ACM Conf. Hum. Inf. Interact. Retr.*, vol. 2016, pp. 121–130.

[57] I. Casanueva, T. Temčinas, D. Gerz, M. Henderson, and I. Vulić, "Efficient intent detection with dual sentence encoders," 2020, *arXiv:2003.04807*.



VICTOR PONCE (Student Member, IEEE) received the B.Sc. degree in computer science from the Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador, the M.Sc. degree in distributed systems from the Polytechnic University of Madrid (UPM), Madrid, Spain, and the M.Sc. degree in open source from the Universitat Oberta de Catalunya (UOC), Barcelona, Spain. He is currently pursuing the Ph.D. degree in computer science with the Université de Sherbrooke, Sherbrooke, Canada.

He specializes in software development and architecture. He has been collaborating in technological projects with companies and institutions, and is also involved in research and academic work. His current research interests include the IoT, distributed systems, ambient-intelligence, software architectures, context awareness, and software engineering.

BESSAM ABDULRAZAK (Member, IEEE) received the B.Sc. degree in electronics from USTHB, Algeria, the M.Sc. degree in robotics from Paris 6, France, and the Ph.D. degree in computer science from Telecom SudParis, France.

He is a Professor of computer science with the University of Sherbrooke and the Director of the AMI Laboratory, Sherbrooke, QC, Canada. He is an active Researcher at the Research Center on Aging and the Interdisciplinary Institute for Technological Innovation. His research interests include the IoT, ubiquitous and pervasive computing, ambient intelligence, smart environments, assistive living technologies, context awareness, and software engineering. He has over 180 peer-reviewed publications, served as the general chair for a number of conferences and workshops, and serves on the editorial board of numerous international journals, as well as program committee of several conferences related to his research interests.