An Automated Cyclic Planning Framework Based on Plan-Do-Check-Act for Web of Things Composition

Mahda Noura
Martin Gaedke
mahda.noura@informatik.tu-chemnitz.de
martin.gaedke@informatik.tu-chemnitz.de
Technische Universität Chemnitz
Chemnitz, Germany

ABSTRACT

Empowering end users to be directly involved in the development and composition of their smart devices surrounding them that achieves their goals is a major challenge for End User Development (EUD) in the context of Web of Things (WoT). This can be achieved through Artificial Intelligence (AI) planning. Planning is intended as the ability of a WoT system to construct a sequence of actions, that when executed by the smart devices, achieves an effect on the environment in response to an end user issued goal. The problem of planning specifically for the WoT domain has not been sufficiently dealt with in the existing literature. The existing planning approaches do not deal with one or more of the following important factors in the context of WoT: (1) random unexpected events (2) unpredictable device effects leading to side effects at runtime, and (3) durative effects. In this work, we propose a cyclic planning system which adopted a PDCA (Plan-Do-Check-Act) process solution to deal with the existing shortcomings for continuous improvement. The planner employs domain knowledge based on the WoTDL (Web of Things Description Language) ontology.The cyclic planner enables continuous plan monitoring to cope with inconsistencies with user issued goals. We demonstrate the feasibility of the proposed approach on our smart home testbed. The proposed planner further enhances the ease of use for end users in the context of our goal-oriented approach GrOWTH.

CCS CONCEPTS

• **Human-centered computing** → *Ubiquitous and mobile computing*; • **Computing methodologies** → **Planning and scheduling**; Knowledge representation and reasoning; Intelligent agents.

KEYWORDS

Internet of Things; Web of Things; Plan-Do-Check-Act; Artificial Intelligence; Planning; Semantic Web

WebSci '19, June 30-July 3, 2019, Boston, MA, USA

ACM ISBN 978-1-4503-6202-3/19/06…$15.00
https://doi.org/10.1145/3292522.3326044

ACM Reference Format:
Mahda Noura and Martin Gaedke. 2019. An Automated Cyclic Planning Framework Based on Plan-Do-Check-Act for Web of Things Composition. In *11th ACM Conference on Web Science (WebSci '19), June 30-July 3, 2019, Boston, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292522.3326044

1 INTRODUCTION

The term Internet of Things (IoT), coined by Kevin Ashton [1], has been an emerging technological trend in recent years in a broad range of domains. The idea of IoT is to enrich physical things ("objects") and places with wireless accessible sensing, computing, and actuating capabilities [4]. The increase in use of smart devices have already changed the way people live, ranging from interactive smart homes to smart parking systems in cities.

One of the main research challenges is how to enable end users to exploit the diverse behaviors of these smart devices and be able to control/monitor and compose them to create new, added-value services [5]. To reach this objective, it is vital that users can easily control how to use their smart objects (i.e., sense data from sensors and affect the physical environment) and how to combine the behavior of different objects to reach some desired goal. The integration of existing Web standards with IoT devices, called the Web of Things (WoT) [8] has simplified access to these smart devices for web developers by allowing the functionality of physical devices to be abstracted and exposed as services on the Web either using RESTful Web Services or WS-* protocol stack.

Automated planning approaches in the Artificial Intelligence (AI) domain would further contribute towards empowering end user's without technical expertise to automatically interact with different WoT devices to reach their goal of interest. They enable the user to specify a coarse-grained goal (What) without providing a concrete way (How). For example, in the smart home domain there are many devices that can change the current situation of the home (like smart lamps, shades, etc.), and can be exploited to provide contextual (i.e., luminosity, humidity, etc.,) information and act automatically in response to a user goal. Some goals in a smart home setting include "increase the temperature", "I want to read a book".

From academia, some approaches [12, 13, 15, 16, 20, 22] have adopted AI planning techniques with the aim to facilitate users achieve their goal in the context of smart environments. However, the existing planning solutions do not consider one or more of the following problems in the context of WoT:

(1) **Unpredictable device effects**: The majority of the solutions assume that each device actuation is defined in terms of preconditions and effects. This means that at design time, the device provider must predict all the possible effects of each WoT device actuation in different scenarios. However, in most cases the device providers only considers the direct effects while either ignoring or simply not being able to predict indirect effects in different situations. Even the predicted effects are not quantifiable since it is impossible to model any kind of complex interaction between arbitrary combinations of devices and the physical environment. For example, how well a heating device can warm the house depends on many criteria's such as where the heater is located, type of heater, whether a window is open, etc. Therefore, it cannot be assumed that the device provider will offer a quantifiable and precise description of all the possible effects.
(2) **Random unexpected events**: In the context of real-world WoT applications such as smart home, smart city, etc. they are characterized by dynamic internal and external factors that can influence reaching the goal of a user. For example, the weather conditions or the unpredictable human behavior can affect the correctness of the produced plan (external event). Moreover, the different WoT devices may join or leave the network at arbitrary times during execution, making device availability unpredictable at runtime (internal event). Therefore, a plan can fail because of an event like malfunctioning device.
(3) **Durative effects**: For some smart devices the effects of an actuation are not instantaneous. This means that the controller must wait for the effects of an actuation to take place. Unlike Web Service composition, in WoT environments the link to the physical world adds a duration dimension to the planning problems because physical actuations require a duration to show the effects and cannot be accelerated on a software level (e.g., by using faster CPU, faster algorithms, etc.,).

Clearly, the above-mentioned factors will have a high impact on the actual execution of device operations which may lead to unexpected behaviour and undesirable user experience and should therefore not be disregarded.

To address the shortcomings of the existing approaches, the contribution of this work is to propose and develop a cyclic planner inspired by the Plan-Do-Check-Act (PDCA) [2] paradigm in business management to deal with unpredictable device effects, random unexpected effects and durative effects. To achieve a level of automation, the planner uses domain knowledge based on the WoTDL[1] (Web of Things Description Language) ontology and WoTDL2API [19] for it's automatic conversion to the OpenAPI specification.The WoTDL ontology has been designed based on our analysis in [17]. The domain knowledge in this work describes the capability of each WoT device in terms of the possible device actuations (preconditions and effects) without restricting the device provider to predict all the possible effects in different environmental contexts. The solution to the plan is realized by employing state of the art AI planners. A fully-working prototype has been implemented which provides the backend to facilitating end users achieve their goals automatically at runtime and on-demand, depending merely on the knowledge of WoT device capabilities, and the user's personal goals. The feasibility of the proposed solution is demonstrated on our smart home testbed consisting of various physical devices by examining a set of realistic case studies. The current contributions are developed in the context of GrOWTH [18], which aims to support EUD for WoT using a goal-oriented technique.

The rest of the paper is structured as follows. Section 2 discusses the related work for planning in smart environments. Then, in Section 3 we show the motivation of the work through a scenario in the smart home domain. In Section 4, the definition of planning for WoT composition regarding the proposed approach is provided. The WoT composition domain is formulated to planning domain definition language in Section 5, and the proposed PDCA-based cyclic planning solution is introduced in Section 6. Section 7 provides the evaluation and results. Finally, Section 8 concludes the paper and provides future insights.

2 RELATED WORK

There is a great deal of research related to planning for web service composition and ubiquitous computing separately. Planning in the context of WoT environments is a relatively new topic and there are few solutions that address this issue directly. This section studies the related work concerning some of the specific AI planning techniques employed from representative studies in different domains, providing a base for extracting the advantages and limitations.

2.1 AI planning in Ubiquitous Computing

The benefits of integrating AI planning techniques with ubiquitous and pervasive computing have long been recognized. Some of the related examples are [3, 10, 14]. In 2003, Heider [10] proposed a solution to simplify complex infrastructures such as multimedia systems. He describes a method to employ AI-based planning techniques to end user applications using a goal-based approach. In this solution, the intelligent components provide a semantic description, so that the planner can independently choose the right devices to reach the user's goal. Although a detailed information about error handling is not presented, the system can respond to the dynamic situation in the application environment. Krüger et al. [14] provides a user interface for controlling intelligent environments to proactively support end users. However, the planning step is performed at design time. In contrast, solutions targeted for end users without the required expertise should be capable of adapt themselves automatically at run-time without the intervention of a domain expert. Chen et al.[3] provides a goal-based solution for service robots using a automatic Hierarchical Task Network Planner (HTN) to achieve the goal. The system also uses a semantic description supported by ontology to make decisions in a context-sensitive context and also takes into account the preferences learned from the user. All of these solutions suffer from the same limitation as they do not consider any kind of uncertainty which makes them unsuitable for highly dynamic environments. However, in smart environments this must be considered as the default and not an option.

[1] https://vsr.informatik.tu-chemnitz.de/projects/2019/growth/wotdl/

2.2 AI planning for Web Service Composition

AI planning techniques have also been used in many approaches to automate (Web) service composition. In this category, the composition process is regarded as the planning problem and services are considered as actions. For example, the SWORD [21] project is an automated web service composition solution based on planning algorithms aimed at minimizing development time and effort. The services are considered as rules and then an expert system decides whether the desired service composition can be realized using the available resources. Based on the planning problem, a service is formulated as an action, with the inputs representing the precondition and the outputs the effects. However, they assumed that the services have a description of the respective inputs and outputs which is a time consuming task to model. In contrast, the project is not aimed for end user's but rather for developers. If there is no plan due to any circumstances, the developer must intervene manually. The PORSCE II framework [9] uses semantic web technologies for service composition. Similar to our approach, they transform the service composition problem into a planning problem and describe it according to the standard PDDL. The problem is then solved by an AI planner and they transfer the solution back to the terminology of the service compositions. It is also capable of identifying service failures by using equivalent services. In contrast, the WoT devices in this work are described using WoTDL (Web of Things Description Language) which is an extension to OWL-S ontology supporting WoT devices. On the other hand, a more elaborate solution for service composition is provided in [12] which also takes the problem of uncertainty into account. In this work planning is performed continuously using Constraint Satisfaction Problem (CSP), such that the steps are predicted offline and revised at execution time. By comparison in our approach we do not perform offline planning and the plans are generated dynamically at runtime per user goal. Wang et al [23] proposed a Graphplan algorithm which considers the problem of uncertainty of execution effect for web service composition.

2.3 AI Planning for Web of Things

The problem of Web service composition is similar to planning for WoT devices based on the REST architectural style, because the main objective of both problems is to combine actions (represented as web services) in a dynamic manner. However, due to the lack of physical devices in this domain executing requests to Web services is almost instantaneous and unlike physical devices do not require processing time and a duration for the actual effect to take place. Furthermore, WoT environments are non-deterministic due to it's connection to the physical world. Towards the approaches developed for WoT composition we can highlight the works of [7, 16, 22, 24]. Mayer et al. [16] proposed a goal-based solution for automatic configuration of intelligent environments. Users use a visual programming tool to indicate which properties their environment should have according to their goals and the system checks whether the target can be reached with the available resources. Since the composition is generated at runtime, the proposed system can also operate in highly dynamic environments which are common in the IoT field. In contrast, they only consider device failures by informing the user with a status code and assume complete knowledge about device effects. Yau et al. [24] attempts to meet the user-expected requirements for a mobile cloud IoT system through markov decision planning. The user communicates the functional and quality requirements to the scheduling system, which would then identify all the smart and non-smart devices in the application environment and after analyzing the device status, would enable the designer to calculate the actions to be taken. More recently, [7] aims to find an automatic solution for detecting and resolving conflicts between the actions of different IoT devices using an AI planner. A recent approach, [22] extends the CSP planner in [12] to a weighted CSP in order to attain partial goal fulfillment for resolving conflicts in a multi-user smart home setting.

To the best of our knowledge, none of the currently proposed approaches consider random unexpected events, unpredictable device effects and durative effects at the same time. The aspect of device failure has been partially addressed in some of the existing literature but the other two aspects are ignored and not dealt with.

3 MOTIVATING SCENARIOS

To show the problems considered in this work, a scenario from the smart home domain is described. Lets suppose Mary is in her smart home and would like to issue the goal: "I want to read a book". For reading a book, she has preferences for light and noise condition. The smart home first measures the ambient luminosity in the room, and finds out that it is too dark, then reacts by switching the lamp on. It also senses the amount of noise in the room and finds out that the noise is above a certain threshold, then turns the TV off. After some time, Mary wants to chill on the couch and issues this goal to the smart home. Based on Mary's preferences the system knows she likes dimmed lights, soft jazz music and the temperature set to 22 degrees. The current temperature in the room is 18 degrees. The fulfillment of this goal cannot be predicted in advance and requires the smart home to be able to produce an initial plan for the increase of temperature and then to repeat the cycle of monitoring the temperature over a span of time until the desired effect (temperature) is reached. Depending on the time of the day, type of heater, location of the heater, an open window, etc., the plan and the time to reach the desired effect differs and obviously cannot be anticipated. Some time passes, and Mary utters to the smart home ("increase the brightness in the room"). The system finds out that it is daylight and decides to open the shutters. However, it is a very sunny hot day outside, and eventually the sun heats up the room and the temperature increases. Here the side effect of opening the shades is increase in the room temperature. The system monitors the environment finding an alternative plan depending on the available devices and acts by closing the shutter, switching the lamp on and turning the fan on. To further complicate things, when trying to close the shades, it does not work, and the service request fails. In this situation, the system should, if possible, be able to choose another device which produces a similar effect on the environment.

4 PLANNING FOR WEB OF THINGS COMPOSITION CONCEPTUAL MODEL

The WoT composition is considered as a planning domain, where actions link to the different operations of physical devices and the

goal is derived from a user request that calls for a state change in the physical environment. All WoT devices are described according to the WoTDL ontology which is an extension to OWL-S [2] ontology for supporting the WoT domain. In this work, we assume that the descriptions of the WoT device operations is provided by a knowledge expert, who is responsible for formalizing the different operations of devices using RDF syntax in OWL ontology. Semantic Web ontologies have been previously used in the domain of planning, interested readers may refer to [9], however it is not the focus of this work.

Definition 1 (WoT composition): The WoT domain is defined as 5-tuple WoTD=$< S, s^o, A, s^*, \gamma >$, where:
$S = s_1, s_2, \ldots, s_n$ is a finite set of states including the initial state s^o and the goal state s^*. Each state is represented by a finite non-empty set of parameters $Par_s = p_1, p_2, \ldots, p_k$ expressing the facts known about this state. Each parameter $p_i \in Par_s$ has a finite domain of V^{p_i}.
A is the set of actions. An action $a \in$A defines its functionality as 4-tuples: a=($name(a)$,$preconditions(a)$,$effects(a)$,$actuations(a)$), where: $name(a)$ is a unique name of the device action, e.g., ("TurnLampOn"), $preconditions(a)$ is a set of propositional formula over Par_s, which follows: $\phi ::= prop|\phi \wedge \phi|\phi \vee \phi|\neg\phi$ and $prop \subset N \times O \times V$ represents an atomic fact, combining the name of the parameter $n_(p_i) \in N$, operator $o_(p_i) \in O = \{=, <, >, \#, \leq, \geq\}$ and the value $v_(p_i) \in V^{p_i}$, $effects(a)$ is a combination of $(E(a), t_E)$, where effect $E(a)$ of action a on parameter $p_i \in Par_s$ can be of the following types:

- $increase(p_i)$ or $decrease(p_i)$ for increasing or decreasing p_i (e.g., increase VOLUME)
- $toggle(p_i)$ for switching between the possible values of p_i (e.g., from ON to OFF)
- $assign(p_i, v)$, for assigning constant value v to p_i (e.g., set TEMP to 22°C)

t_E (time to effect) is the duration stating the amount of time for an effect to occur cf. (Figure 1) (e.g., the time required for the actual increase in temperature in a room after the thermostat has completed its operation)
$actuations(a)$ are a set of HTTP requests that can be sent to the physical devices that invokes the operations on the devices. It is a tuple $(method, url, body, headers, t_0)$, where $method$ indicates the desired action to be performed on a Web device $GET|POST|PUT$, url the mechanism for retrieving the resource, $body$ the body of the HTTP message, $headers$ and t_0 is the time required for the physical operation triggered by this actuation cf. (Figure 1) (e.g., the time required to close a rolling shutter after the request is sent to the shutter).
$\gamma : S \times A \rightarrow S$ is the transition function between the states associated with actions from A performed by devices.
The solution to the planning problem described by the WoT composition is a finite sequence $\pi \in A^*$ of actions. This plan $\pi = a_1 \ldots a_n$ is executable from state s_0, if there exists a sequence of states $\sigma = s_0 \ldots s_n$ so that, for $i = 0, \ldots, n-1, s_(i+1) = \gamma(s_i, a_(i+1))$ the states form a consistent sequence of transitions and $s_n = s^*$ this sequence ends in the target state.

[2] https://www.w3.org/Submission/OWL-S/

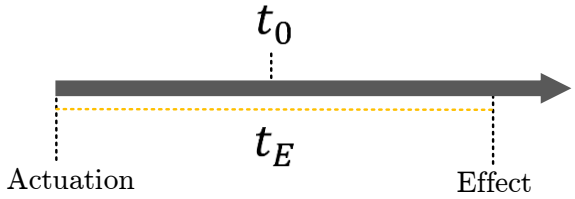


Figure 1: Timeline from the actuation to the actual effect

5 ENCODING THE WOT COMPOSITION IN PLANNING DOMAIN DEFINITION LANGUAGE

While most planning systems in different domains assume that the planning domain is defined by a domain expert at design time in the Planning Domain Description Language (PDDL) or similar, that is not possible for dynamic smart environments. In this work our method is driven by the requirement that the planning domain is constructed at run-time automatically in response to a user goal. Therefore, in this section we show how, given a user goal, the planning domain and the problem domain is automatically constructed from the WoTDL ontology in RDF to PDDL. The planning domain and the problem domain can then be used as input to any state-of-the-art planning technology to find a plan. In particular, we represent them making use of PDDL2.1 [6], which provides numeric features that are relevant for this work, e.g. to represent the value of temperature.

5.1 Planning Domain

In order to generate the planning domain file automatically, a parser is developed to transform the WoTDL ontology instances expressed in RDF to PDDL. A minimal example of TTL representation to PDDL planning file generation is presented in the top right of Figure 2, where the mappings are marked in different colours. It is worth noting that the ontology instance is not complete and only the relevant instances are shown for the sake of clarity.

The domain file, in it's minimal state, consists of a description of all *types*, *predicates*, and the *actions* in the environment. The *types* describe the different types of WoT devices that exists in the smart environment (red) like a thermostat, lamp, shutter, etc as well as the environmental characteristics such as the room temperature, ambient brightness. *Predicates* are the properties of objects and help to describe the state of the domain, for example, the *LampPowerState* can be used to check whether the Lamp is switched on or off (light green). The actions express the operations provided by the available devices and influence the state of the world. They are described in PDDL by a name (yellow), a list of parameters, preconditions, and effects. The example illustrates an action to turn off the lamp (*SwitchOffLamp*) and expects an argument for the variable *a1* of type *lamp* as a parameter. The preconditions and effects are represented by logically linked predicates and refer to the variables passed to the action by the variable name. In the example, the action assumes as a precondition (aqua blue) that the *lamp* object passed to it must be in the "on" state. If the condition is met, the planner can select this action to produce the described effect (light blue). In this case,

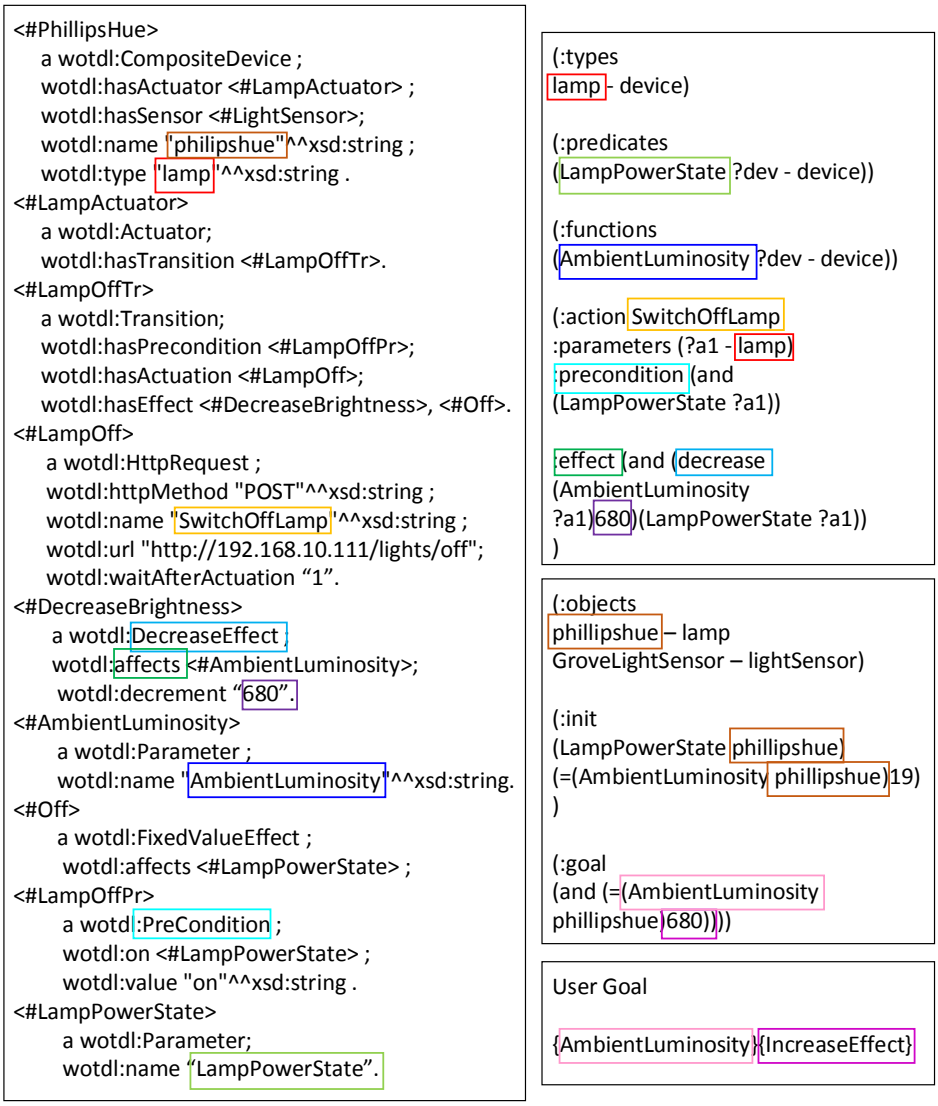


Figure 2: WoTDL instance to PDDL transformation

the (*PowerLampState*) of the *lamp* object would be "true" after the operation is performed.

5.2 Problem Domain

The generation of the TTL to PDDL problem file is presented in the bottom right of Figure 2, The problem definition specifies the problem to be solved in PDDL syntax. This consists of the objects that are currently in the domain, the initial state and the desired target state. Initially, all object instances are listed with their assigned type a *philipshue* of the type *lamp* (dark orange). Following is the description of the initial state, which is displayed using the predicates from the domain definition, for example the Phillips Hue lamp is currently switched on and the ambient luminosity is equal to 19. The initial state of the environment is identified by sending an HTTP GET request to the different sensors. Finally, the description of the desired target state follows (pink), which is also displayed using the predicates. The planner should find a way to increase the brightness to 680. The goal state is expressed by the end user and converted into an appropriate format. All this information is then enough for a PDDL-capable AI planner to generate a plan as output.

6 PLAN-DO-CHECK-ACT BASED AUTOMATED PLANNING

In this section we present the methodology that is employed for planning for WoT composition. We propose a cyclic planner inspired by the Plan-Do-Control-Act (PDCA) [2] paradigm in the context of continuous improvement process in business management. This concept can be depicted as an analogy to the problem presented in this work where the achievement of the goal desired

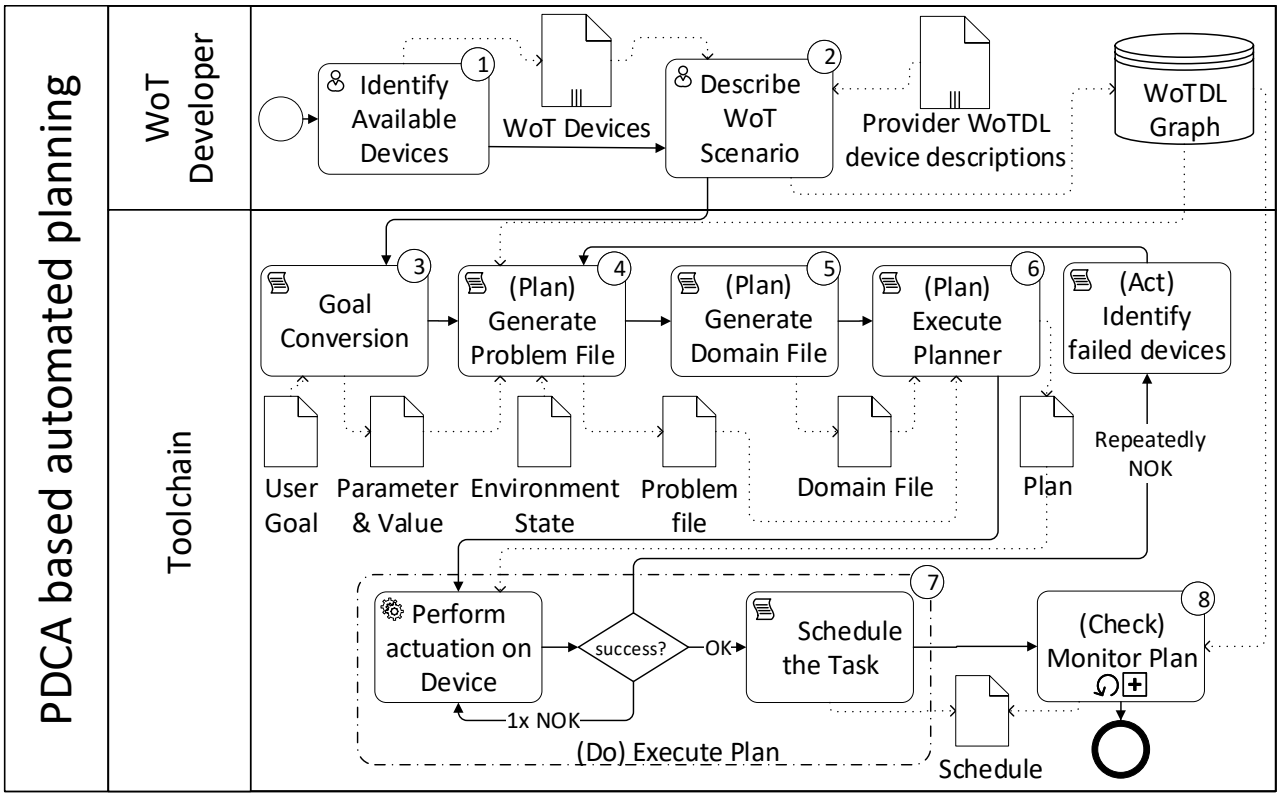


Figure 3: PDCA-based automated planning methodology for WoT composition

by the user can be regarded as a process which requires continuous observation and one or more possible re-planning iterations. The main idea behind our cyclic planner is to sense the environment continuously after an initial plan is executed in order to find and resolve inconsistencies with user issued goals. Figure 3 illustrates an overview of the workflow using BPMN. The PDCA-based automated planning for WoT composition architecture involves three main roles:

- *IoT Device Provider* is the role in charge of manufacturing and describing IoT devices and sensors according to the WoTDL ontology. The WoTDL ontology describes the key concepts of AI planning-based applications in the context of WoT.
- *End User* is the actor who will ultimately use the system and interacts with it by issuing goals (e.g., "It's too warm") using voice interactions.
- *Automated Planner Toolchain* is a system role representing our toolchain for supporting user issued goals through automatic generation and execution of a plan.

The planner involves four major steps:

- **Plan**: The planner generates an initial plan depending on the user goals, context, and available WoT devices (cf. steps 4-6)
- **Do**: The list of actions in the generated plan is executed by physical WoT devices through HTTP requests to produce the desired effect (cf. step 7)
- **Check**: The current environment state is compared to the goal state (cf. step 8)
- **Act**: If the current state and the goal state matches the cycle ends, otherwise the required procedure is taken to deal with unexpected events, unpredictable device effects and durative effects.

In the following we describe the steps of this process in sequential order.

Step 1- Identify Available Devices: Before the planning process starts the WoT developer needs to identify the set of IoT devices which will be utilized for the users scenario.

Step 2- Describe WoT Scenario : The WoT developer creates the WoTDL description by aggregating the device descriptions provided by the device providers for each device involved in the user scenario. Then, the resulting WoTDL description is stored as RDF graph according to the WoTDL ontology.

Step 3- Goal Conversion : The end users can then interact with the system for issuing goals to the WoT environment using voice interactions. The goal from the user is passed to this component and this phase is responsible for performing natural language processing to convert it in a form that can be processed by the machine. The output of this step is a JSON object representing a set of parameter and values. For example, the goal "it's too warm" is converted into {"temperature","24"}.

Step 4- Generate PDDL Problem File : Once the users desired goal is processed it should be transformed into a form that can be

processed by an AI planner. In this step, the PDDL problem file is automatically generated from the WoTDL graph using a parser. To formulate the PDDL problem domain, the environment state (initial state) and the goal of the user are used as input. The initial state is identified by sensing the available sensors and actuators. More details about this step is discussed in Section 5.2.

Step 5- Generate PDDL Domain File : In this phase, the PDDL domain file is automatically generated using a parser. To create the domain file, the WoTDL graph is initially queried to retrieve all device instances that exists in the scenario as well as the environmental properties.More details about this step is discussed in Section 5.1.

Step 6- Execute Planner : Given the PDDL domain and problem files as input the planner produces an initial plan to the given planning problem to achieve the desired goal. In particular, the planner creates a planning graph with all existing states as vertices and the actions as edges. Then the Planner searches through all device actions that affect one of the goal variables to reach the goal state. After that, a sequence of steps (WoT devices actuations) to reach the goal is generated as output. It is worth noting that in this work, a state-of-the-art planner, Metric-FF [11], is used to produce the plan.

Step 7- Execute Plan : The plan generated by the Planner and the available actions from the previous step are passed as input to this phase. It then maps the steps of the plan to the actions and actuates each action by executing the associated HTTP request. These requests are defined in the device description in terms of HTTP method, URL, and body. The HTTP requests cause the physical WoT devices to produce the desired effects. This step also includes failure detection mechanism by sending a request to the unresponsive device for a maximum of three times. If the device does not respond after three times, the device is considered as fail and a re-plan is performed without the corresponding device. On the other hand, if it responds the task is scheduled for further monitoring.

Step 8- Monitor Plan : This phase constantly monitors the state of the environment to check whether the expected effect has occurred. All the available sensors from the WoTDL graph are queried at regular intervals to check whether an action for monitoring is listed. If so, the current sensor value is compared to the target state value. If these values match, the action was successful and can be excluded from further monitoring. However, if the values do not match the respective action is either executed again or the time for the next check is recalculated. If the effect of the respective action could not once again be evaluated as successful, it is marked as "failed" and a new planning process for the associated target is initiated. If there is no associated action for the respective sensor measurement parameter, the sensor value is only compared with the initial state of the measurement parameter. If these values are different from one another, there is a side effect and the user is informed accordingly. Inside this component there is a *TaskMonitor* which runs continuously as a service to monitor all sensors and the actions. Successful actions are removed because the goal has been reached, whereas failed actions inform the user or automated countermeasures are taken.

7 EVALUATION

In this section, we demonstrate the feasibility of the proposed cyclic planning approach using multiple smart home cases derived from Section 3 and explore the scalability over different number of devices. For each unique case the toolchain announced in Section 3 is tested in our smart home testbed located in the VSR laboratory to analyze the behavior of the approach based on the completion of the user issued goal.

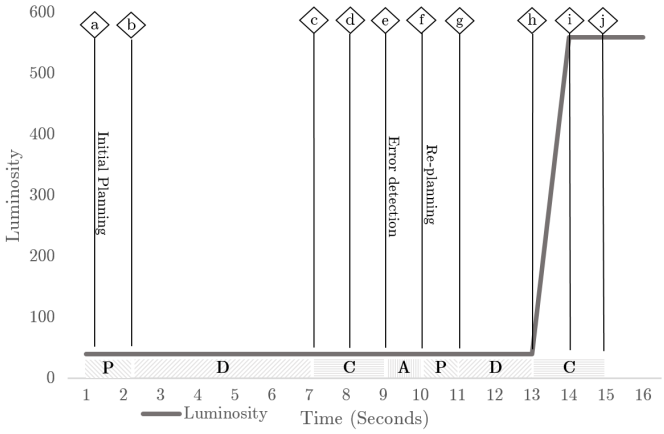
In particular the testbed includes three sensors including temperature, humidity, light, and four actuators including lamp, fan cooler, shade and thermostat as listed in Table 1. Each of these IoT devices are accessed through different non-web interfaces. The Phillips Hue lamp is the exception, since it directly targets end users and therefore provides a web-based API. Given these devices and by following our methodology, we modeled the existing testbed using the WoTDL ontology in turtle format. Furthermore, the ontology instances of the smart home scenario are extracted to compose and deploy the standard OpenAPI specification based on the WoTDL2API [3] toolchain [19]. In this way, the IoT devices supporting different communication protocols are transformed into WoT devices. In order to offer the user an easy way to enter their goals into the system the Amazon Echo Dot is used. The Amazon Alexa Skill Kit [4] language service is utilized to create different intentions, so-called intents, which can be triggered with different utterances.

7.1 Study Cases

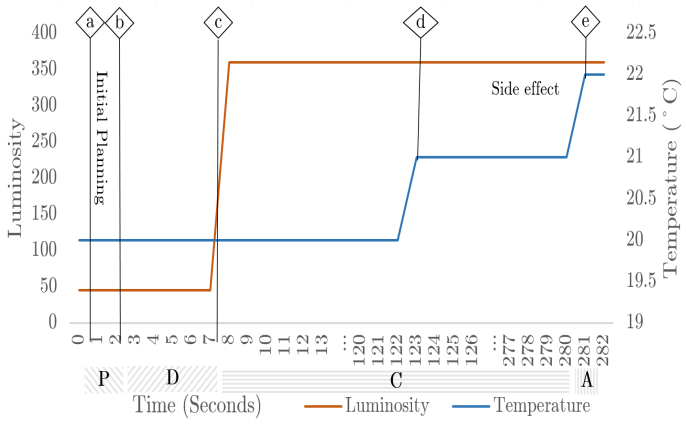
The evaluation cases applied to the testbed are extracted from the scenario mentioned previously in Section 3. For each case, we need to simulate the situation so that the PDCA planner can respond to the real situation. In the following, the cases are discussed:

Case A (device failure): This case represents random internal effects showing a malfunctioning device during runtime and if possible finding an alternative plan. At the beginning of the experiment both light sources are extinguished, i. the Philips lamp is off and the roller shutter lowered. Then the user expresses the goal "increase brightness". The system first tries to open the roller shutters to fulfill the goal. However, in order to simulate the problem situation, the power supply of the servomotor is disconnected to make the roller shutter fail. Figure 4a shows the results of using the cyclic planner to solve device failure. The X-axis shows the evaluation time since the users goal is issued at 0, and the Y-axis stands for the current value of brightness ranging from 0 to 600. The black lines with diamonds on top demonstrates important events occurred in the experiment during the user issued goal, each block of measurement shows a different period in the PDCA cycle. The main events in this figure are as follows: (a)The goal is extracted from the user and the planner delivers the plan, (b) the command to actuate the servomotor for the shutter is sent. Since the activation of the shutter requires 5 seconds (*waitForActuation*) the system waits in the meantime, (c) the execution is completed, (4) the *TaskMonitor* detects that the luminosity has not changed and therefore extends the waiting time by the value associated with the action (1 second), (d) the *TaskMonitor* again detects that the desired luminosity

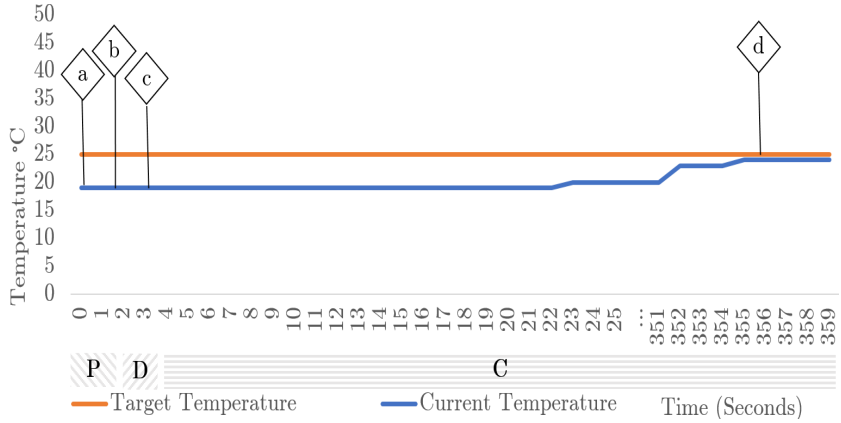
[3] https://github.com/heseba/wotdl2api
[4] https://developer.amazon.com/alexa-skills-kit/



(a) Case A Failure Detection



(b) Case B Side Effect



(c) Case C Durative Effects

Figure 4: evaluation results on different study cases

Table 1: IoT devices in testbed for evaluation

Device Name	Protocol	Description
Phillips Hue Lamp	z-wave	Controls brightness and color of light
Shades Motor	Pulse Width Modulation	Motor for the curtains
Relay + Heating Panel	Custom Interval Control	Controls the heating
DHT11	Single-bus serial protocol	Temperature and Humidity Sensor
DC motor	Analog voltage-based	Mini Cooling Fan
Eqiva Thermostat	Bluetooth	Thermostat

has not been reached and again extends the monitoring period by one second. (e) On the third monitoring, the device is identified as faulty and is no longer considered during a new planning process, (f) the new PDDL files have been generated without considering the shutter and the planner provides a re-plan, (g) The actuation command to turn the Philips Hue lamp on is executed, (h) as a result of the actuation the luminosity increases, (i) finally the *TaskMonitor* recognizes the achievement of the desired light intensity and the goal is reached.

Case B (Side effects):The purpose of this case is to detect unwanted side effects due to incomplete knowledge about all the possible effects during the execution of the plan. In order to simulate an external event, the Bluetooth thermostat of a heating located outside the testbed is switched on to simulate the sunlight heating the room through the window. The initial situation in the house is that the Phillips Hue lamp is switched off and the roller shutter is closed. The user then specifies the goal "increase the brightness". The results of this case is shown in Figure 4b. This figure contains

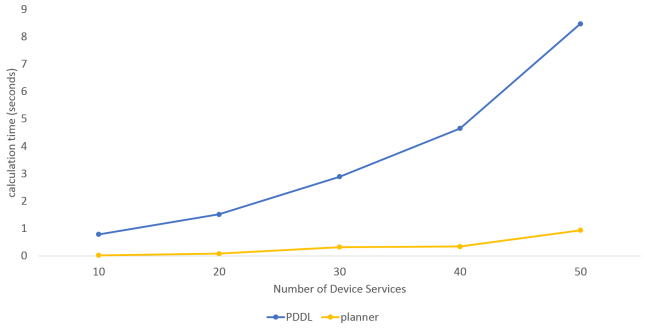


Figure 5: Planning time over many devices

two observations representing the change in temperature and luminosity. The blue line represents the temperature change and the orange line shows the luminosity. The meaning of the X-axis along with the events a and b are the same as case A. Some of the duration shown in blanks have been omitted due to brevity. The following events are worth highlighting: (c) The roller shutter is open in response to the plan executed in event b and therefore the ambient luminosity gradually increases in the room, (d) in the Control phase, the *TaskMonitor* recognizes the temperature rise from 20 to 21 degrees. In this phase, there is no detection yet, as the tolerance value is set to 1, (e) the temperature rises to 22 degrees and the *TaskMonitor* ultimately detects the unwanted side effect of opening the shutter.

Case C (Durative Effects): The purpose of this experiment is to show how the cyclic planner handles actuation's that require a longer period of time to produce the desired effect. The current temperature in the room is measured at 19 degrees and the user expresses the goal "increase the temperature to 25 degrees". By querying the WoTDL graph the *timeToEffect* of the heater is calculated as 6 minutes. The behaviour of the system is shown in Figure 4c. To explain this figure, the following events are considered: (a) the target temperature of 26 degrees is accepted by the system, the PDDL domain and problem files are generated and the planner creates a plan, (b) the thermostat is activated and set to the appropriate temperature, (c) the task is added to the *TaskList* for monitoring since the effect time is expected in 6 minutes. (d) the *TaskMonitor* realizes that the t*imeToEffect* span is expired and senses the current temperature. The temperature is sensed at 24.4 degrees which is in the tolerance range. Therefore the monitoring step is successfully completed.

To summarize, the results verify that the proposed cyclic planning approach works for the scenario we introduced before in Section 3 and can provide valid plans for user issued goals.

7.2 Scalability

One important concern with respect to all smart composition systems is how they scale with the growing number of physical devices. Therefore, We explored the scalability of our approach to demonstrate it's performance to control smart environments in realistic contexts. To demonstrate this, the WoTDL graph is supplied with additional device instances and then the planning process is initiated. In this experiment, the time required for generating the PDDL files and running the AI planner we use in our system - Metric-FF [11]- were considered separately. The results (Figure 5) demonstrates the time it takes for the Metric-FF planner to find the solution is less than 1 second for 50 devices, but there is a sharp increase compared to the duration of only 40 devices. The time required for querying the WoTDL graph and generating the PDDL problem and domain files does significantly increase, but this can be easily resolved by caching the device capabilities locally and only generating the PDDL files whenever a change in the environment state occurs.

Although, the performance of the planning process is low, it is practical for up to a number of 20 devices. Up to this number of devices, the total time required to plan is about 2.2 seconds. Based on the current forecasts there will be about 15 smart appliances per household by 2020. We conclude that our approach is suitable to control smart environments like smart homes that do not contain a high number of devices. More optimization is necessary for larger scales and application scenarios with real-time constraint.

8 CONCLUSION AND FUTURE WORK

In this paper, we initially proposed a smart home scenario based on the unpredictable situations that can occur during planning for WoT composition. Then we proposed a fully automatic cyclic planning approach to facilitate the composition of WoT devices for end users. The solution considers three important situations in the context of WoT environments : device failure, unpredictable device effects causing side effects and durative effects. The device capabilities are described using the WoTDL ontology providing *interoperaility*, breaking down interoperability barriers through semantics and *flexibility* of supporting new device descriptions in an evolving way. Furthermore, the proposed solution is targeted for end users by only stating *what* has to be performed, without providing *how* it can be met. The results on our smart home testbed illustrated that the proposed approach can automatically solve the considered scenarios at runtime successfully without the involvement of a user. In the future, we will further optimize the solution for more realistic smart application domains. We also plan to extend the GrOWTH framework by extracting the device capabilities expressed in terms of actuations, preconditions and effects automatically, rather than adding these instances in the WoTDL ontology Graph by a domain expert. We expect that this would further improve the usability of the system for end users without domain expertise.

ACKNOWLEDGMENTS

We would like to thank Phil Dietrich for his contributions towards the implementation of this research.

REFERENCES

[1] Kevin Ashton and others. 2009. That 'internet of things' thing. *RFID journal* 22, 7 (2009), 97–114.
[2] Ron Basu. 2004. *Implementing quality: a practical guide to tools and techniques: enabling the power of operational excellence.* Cengage Learning EMEA.
[3] Chia Hung Chen, Alan Liu, and Pei Chuan Zhou. 2014. Controlling a service robot in a smart home with behavior planning and learning. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* 2014-Janua, January (2014), 2821–2826. https://doi.org/10.1109/smc.2014.6974356

[4] Li Da Xu, Wu He, and Shancang Li. 2014. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics* 10, 4 (2014), 2233–2243.

[5] Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. 2017. Empowering End Users to Customize their Smart Environments. *ACM Transactions on Computer-Human Interaction* 24, 2 (2017), 1–52. https://doi.org/10.1145/3057859

[6] Maria Fox and Derek Long. 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research* 20 (2003), 61–124.

[7] Emre Göynügür, Sara Bernardini, Geeth de Mel, Kartik Talamadupula, and Murat Sensoy. 2017. Policy conflict resolution in iot via planning. In *Canadian Conference on Artificial Intelligence*. Springer, 169–175.

[8] Dominique Guinard and Vlad Trifa. 2009. *Towards the web of things: Web mashups for embedded devices* (15 ed.). Madrid, Spain.

[9] Ourania Hatzi, Dimitris Vrakas, Nick Bassiliades, Dimosthenis Anagnostopoulos, and Ioannis Vlahavas. 2013. The PORSCE II framework: Using AI planning for automated semantic web service composition. *The Knowledge Engineering Review* 28, 2 (2013), 137–156.

[10] T Heider. 2003. Goal oriented assistance for extended multimedia systems and dynamic technical infrastructures. *Proceedings of the Seventh IASTED International Conference on Internet and Multimedia Systems and Applications* 7 (2003), 62–67.

[11] JÃűrg Hoffmann. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of artificial intelligence research* 20 (2003), 291–341.

[12] Eirini Kaldeli, Alexander Lazovik, and Marco Aiello. 2016. Domain-independent planning for services in uncertain and dynamic environments. *Artificial Intelligence* 236 (2016), 30–64. https://doi.org/10.1016/j.artint.2016.03.002

[13] Thomas Kirste, Thorsten Herfet, and Michael Schnaider. 2001. EMBASSI : Multimodal Assistance for Universal Access to Infotainment and Service Infrastructures. *Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing: Providing for the Elderly* (2001), 41–50. https://doi.org/10.1590/S0101-32622010000200006

[14] Frank Krüger, Gernot Ruscher, Sebastian Bader, and Thomas Kirste. 2011. A context-aware proactive controller for smart environments. *CEUR Workshop Proceedings* 747 (2011). https://doi.org/10.1524/icom.

[15] Simon Mayer, Dominic Plangger, Florian Michahelles, and Simon Rothfuss. 2016. UberManufacturing A Goal-Driven Collaborative Industrial Manufacturing Marketplace. *Proceedings of the 6th International Conference on the Internet of Things - IoT'16* (2016), 111–119. https://doi.org/10.1145/2991561.2991569

[16] Simon Mayer, Ruben Verborgh, Matthias Kovatsch, and Friedemann Mattern. 2016. Smart Configuration of Smart Environments. *IEEE Trans. Automation Science and Engineering* 13, 3 (2016), 1247–1255.

[17] Mahda Noura, Amelie Gyrard, Sebastian Heil, and Martin Gaedke. 2018. Concept extraction from the Web of Things knowledge bases. In *Proceedings of the International Conference WWW/Internet 2018.*

[18] Mahda Noura, Sebastian Heil, and Martin Gaedke. 2018. GrOWTH: Goal-Oriented End User Development for Web of Things Devices. In *International Conference on Web Engineering 2018.*

[19] Mahda Noura, Sebastian Heil, and Martin Gaedke. 2019. Webifying Heterogenous Internet of Things Devices. In *International Conference on Web Engineering 2019.*

[20] Javier Palanca, Elena del Val, Ana Garcia-Fornes, Holger Billhardt, Juan Manuel Corchado, and Vicente Julián. 2018. Designing a goal-oriented smart-home environment. *Information Systems Frontiers* 20, 1 (2018), 125–142. https://doi.org/10.1007/s10796-016-9670-x

[21] Shankar R Ponnekanti and Armando Fox. 2002. Sword: A developer toolkit for web service composition. In *Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI*, Vol. 45.

[22] Noel Nuo Wi Tay, Janos Botzheim, and Naoyuki Kubota. 2018. Human-Centric Automation and Optimization for Smart Homes. *IEEE Transactions on Automation Science and Engineering* (2018), 1–13. https://doi.org/10.1109/TASE.2018.2789658

[23] Pengwei Wang, Zhijun Ding, Changjun Jiang, Mengchu Zhou, and Yuwei Zheng. 2016. Automatic web service composition based on uncertainty execution effects. *IEEE Transactions on Services Computing* 9, 4 (2016), 551–565. https://doi.org/10.1109/TSC.2015.2412943

[24] Stephen S. Yau and Arun Balaji Buduru. 2014. Intelligent Planning for Developing Mobile IoT Applications Using Cloud Systems. *2014 IEEE International Conference on Mobile Services* (2014), 55–62. https://doi.org/10.1109/MobServ.2014.17