IoTWC: Analytic Hierarchy Process Based Internet of Things Workflow Composition System

Yinhao Li
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
y.li119@newcastle.ac.uk

Devki Nandan Jha
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
d.n.jha2@newcastle.ac.uk

Gagangeet Singh Aujla
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
gagi_aujla82@yahoo.com

Graham Morgan
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
graham.morgan@ncl.ac.uk

Albert Y. Zomaya
School of Information Technologies
University of Sydney
Sydney, Australia
albert.zomaya@sydney.edu.au

Rajiv Ranjan
School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom
raj.ranjan@ncl.ac.uk

***Abstract*—Internet of Things (IoT) allows the creation of virtually endless connections into a global array of distributed intelligence. However, the design, development, and deployment of IoT applications are complex and complicated due to various unwarranted challenges. For instance, addressing the IoT application users' subjective and objective opinions with IoT workflow instances remains a challenge for the design of a more holistic approach. Moreover, the complexity of IoT applications increased exponentially due to the heterogeneous nature of the Edge/Cloud services, utilised with the aim of lowering latency in data transformation and increase re-usability. Hence, in this paper, we present an *IoT workflow composition system (IoTWC)* to allow IoT users to pipeline their workflows with proposed IoT workflow activity abstract patterns. IoTWC leverages the analytic hierarchy process (AHP) to compose the multi-level IoT workflow that satisfies the requirements of any IoT application. Moreover, the users are befitted with recommended IoT workflow configurations using an AHP based multi-level composition framework. The proposed IoTWC is validated on a user case study to evaluate the coverage of IoT workflow activity abstract patterns and a real-world scenario for smart buildings. The comprehensive analysis shows the effectiveness of IoTWC in terms of IoT workflow abstraction and composition.**

***Index Terms*—Internet of Things, Workflow Composition, Multi-criteria Decision Making, Analytic Hierarchy Process**

I. Introduction

The Internet of Things (IoT) has become an important component of leading software development and an integral part of our daily life. IoT is visible in many diverse communities such as smart buildings, smart agriculture, and smart industry [1]. With increasing demand and domain diversity, IoT applications become more complex in design, development, and deployment due to their multi-party Cloud/Edge/IoT resource makeup and the service level of agreement (SLA) requirements of their inter-connections [2].

IoT applications can be modelled as a directed acyclic graph (DAG) with data transformation tasks as its nodes and data flow dependencies (or control flow dependencies) as its vertices. To be most useful in a modelling sense, we need an IoT workflow pattern that can be general enough to define any IoT application. Representing a generic IoT workflow pattern is intricate because of the heterogeneity of IoT infrastructure (IoT device, Edge device, and Cloud), coupled with application dependency on infrastructure and variability of data.

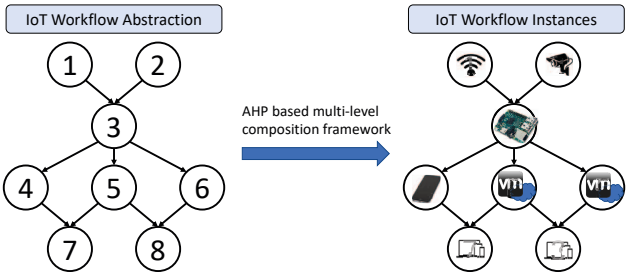

Fig. 1. IoT Workflow Abstractions and Instances Mapping

To narrow the accuracy of our modelling approach we provide a mechanism that converts the abstract workflow patterns into specific application workflow instances (see Fig. 1) (e.g., a Smart Home based on different QoS requirements such as IoT device mobility, data privacy, and latency).

Identifying a set of reasonable IoT workflow activity patterns (provisioning an abstraction understandable to the engineer yet maintaining its usefulness for composition purposes), which covers IoT data transformation tasks and workflow activities, is the primary challenge in IoT workflow representation and composition. The aforementioned research challenges can be summarised to the following research questions:

- What are the abstract IoT patterns required to create generalized IoT applications across any domain?
- How to convert IoT workflow abstractions into IoT workflow instances specific for an IoT application based on the heterogeneous QoS requirements?

978-1-7281-1099-8/20/$31.00 ©2020 IEEE
DOI 10.1109/IC2E48712.2020.00007

Traditional composition engines such as BPMN (Business Process Model and Notation) are specific only for business processes [3]. Workflow composition frameworks from academia [4]–[7] and industry (Calvin [8], Kubernetes [9], Amazon cloudFormation [10]) are also available but all of them are specific to a certain infrastructure or domain. Addressing IoT application users' subjective and objective opinions with abstract understanding of IoT workflow instances remains a challenge for a more holistic (cross-domain) approach. None of the existing frameworks are able to perform a generic IoT application composition.

To address the above challenges, we proposed and developed a novel composition framework, *IoTWC (IoT Workflow Composition System)*. Based on an extensive literature study and interviewing multiple people (domain experts/users), we provide basic activity patterns that are abstracted within IoTWC. IoTWC leverages the analytic hierarchy process (AHP) [11] to compose the multi-level IoT workflow that satisfy the requirements of any IoT application. The proposed IoTWC is validated on a user case study to evaluate the coverage of IoT workflow activity abstract patterns and a real-world scenario for smart buildings which show the effectiveness of IoTWC in terms of IoT workflow abstraction and composition.

The rest of this paper is organised as follows. Section II discuss the related work. A detailed discussion about IoT workflow activity abstract patterns are presented in Section III. Section IV describes the IoT AHP model and multi-level composition framework in IoTWC while system design and implementation is discussed in Section V. Section VI presents the system evaluation with a user case study and a real world scenario validation followed by conclusion in Section VII.

II. Related Work

Application specification and composition is well-studied problem for general purpose applications. BPMN (Business Process Model and Notation) is a graph-oriented specification language representing business processes. As an implementation of BPMN for the web service domain, WS-BPEL (Web Services Business Process Execution Language) is an XML-based specification language that leverages web services to interact with each other in any web-based business approach. [12] uses the BPEL for IoT application modelling to realise business processes. [13] defines IoT-aware business processes with the BPEL language to avoid increases in process complexity. XPDL (XMP Process Definition Language) is standardised by the Workflow Management Coalition (WfMC) to interchange the graphical business process workflow models to an XML-based model. Although these business workflow modelling approaches can present the flow of information, they do not model the specific data transformation tasks in IoT which are key to a successful IoT deployment and requires managing the correct configuration of devices and their usages.

[14] proposed a RESTful web service to encapsulate heterogeneous IoT devices, in order to manage and compose IoT services for social network deployments. [15] presented a context-aware web service description language based on adaptive service composition frameworks to support dynamic reasoning in IoT-based smart city applications. [16] proposed trust management to support service composition applications in service-oriented architecture based IoT systems. [17] developed a multi-cloud IoT service composition algorithm to create an energy-aware composition plan to fulfill user requirements. These works have addressed many IoT workflow and service composition research problems in a variety of application domains. However, there is no general approach to abstract the IoT workflow composition problems that can be useful across multiple IoT domains.

Many industry sectors also propose composition frameworks for supporting IoT application deployment. [18] applied TOSCA (Topology and Orchestration Specification for Cloud Application) in IoT to automate the deployment process of IoT applications based on the mosquito message broker. [8] modelled IoT applications using four well-defined aspects: describe, connect, deploy and manage. This eased IoT application development and deployment processes for engineers. [9], [10] provided open-source approaches to benefit the design and deployment of IoT application workflows. However, these frameworks are designed for expert users having detailed background knowledge. This is not suitable for general users having little technical knowledge. Additionally, these frameworks do not focus on data transformation aspects of the IoT workflow composition process.

III. IoT Workflow Activity Abstract Pattern

Due to the heterogeneity of IoT applications, including their supporting Cloud/Edge data processing tasks, the traditional approach of presenting IoT workflow activities as a set of abstract patterns is difficult. Representation of IoT applications as a directed acyclic graph (DAG) simplifies this process by modelling each node as an IoT data transformation task. Thus, data is considered as a meta component in an IoT application. Considering a typical IoT application, the fundamental process consists of gathering and dissemination of data (e.g., capturing raw data from temperature sensors and storing this processed data into the database). In this typical example, we abstract IoT workflow activities as the following data transformation patterns: *Data Capture*, *Data Store*, *Data Inference*, *Data Filter*, *Data Aggregate*, *Data Visualisation*, *Data Translate*, and *Actuation*. Furthermore, we also consider the main abstract connection patterns in IoT: *Things Connect* and *Data Transfer* that consist of four sub-categories (Edge to Edge, Edge to Cloud, Cloud to Edge and Cloud to Cloud) in terms of sender and receiver of data. An example IoT workflow with proposed abstract patterns is shown in Fig. 2.

After comprehensive literature reviews and interviews with IoT domain experts, we believe that our proposed IoT workflow activity abstract patterns cover the majority of the IoT data transformation tasks and associated data flows together with control flow requirements across multiple domains. The detailed evaluation will be discussed in section VI.

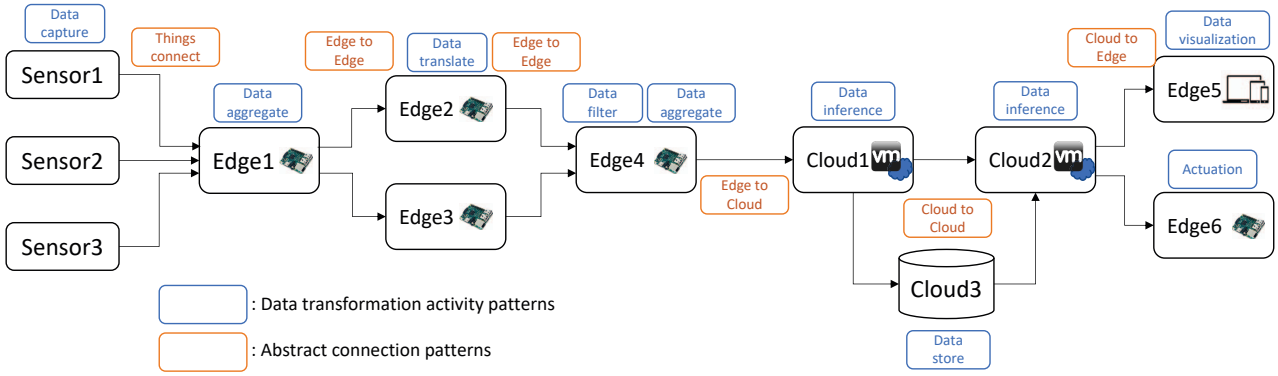

Fig. 2. Example IoT workflow with abstract patterns

IV. IoTWC: AHP-based Model and Multi-level Composition Framework

In this section, we describe the IoT AHP model and multi-level composition framework.

A. IoT Analytic Hierarchy Process Based Model

The complexity of IoT applications and their QoS requirements bring huge challenges for converting IoT workflow abstractions into IoT workflow instances. The fundamental aspect of this problem can be formalised as a Cloud/Edge resource Configuration Knowledge Representation (CKR) selection task. This selection task can be translated into a multi-criteria decision analysis problem with multiple alternative choices.

Analytic Hierarchy Process (AHP) [11] is an effective method for solving complicated multi-criteria decision making problems. In addition, AHP encourages decision makers to prioritise and identify how and when an appropriate decision is considered. The AHP can capture not only subjective, but also objective aspects of a decision by decreasing complicated decisions to a list of pairwise comparisons and then merge the results. Moreover, AHP adopts a technique to evaluate the consistency of the decision made by decision makers, resulting in the lowering of bias in the overall process under consideration.

In the AHP algorithm, a set of evaluation criteria and alternative choices need to be considered and identified by users. In addition, users also need to specify a weight between each two of the evaluation criteria based on their pairwise comparisons. Next, it will assign a score to each alternative choice based on users' pairwise comparisons. Finally, the algorithm will combine the criteria weights and the choice scores to generate a global ranking for alternatives. The ranking represents the sequence of each choice.

Fig. 3 illustrates the hierarchical representation of the CKR selection problem. Here, CKR selection is the primary goal and is based on three main criteria: Resource Cost, Resource QoS, Data. For each criterion, there is a set of sub-criteria prescribed (e.g., Hardware Cost, Hosting Cost and Network Cost are sub-criteria for Resource Cost; Reliability, Mobility, Heterogeneity, Scalability, Capability and Resource Availability are sub-criteria of Resource QoS). Based on our selection goal, we have a list of alternative choices among edge and cloud resources, such as CKR1 (high-performance edge resource, Raspberry Pi 4 model), CKR2 (low-performance edge resource, Raspberry Pi Zero model), CKR3 (high-performance cloud resource, AWS EC2 t2.xlarge) and CKR4 (low-performance cloud resource, AWS EC2 t2.micro). When the IoT AHP model is enacted, IoTWC ranks all alternatives based on criteria weights between each two of the evaluation criteria. Users can retrieve the score and ranking, and the decision of CKR selection.

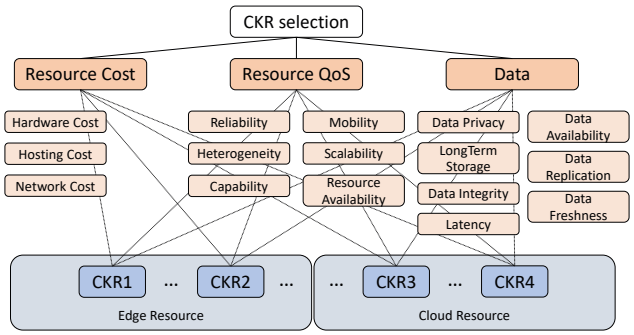

Fig. 3. CKR Selection Hierarchy

B. Criteria Definition

In this section, we discuss our criteria in more detail.

1) Resource Cost: Resource cost represents all financial commitments in the IoT application life cycle, like hardware, hosting, and network costs.

- *Hardware Cost*: Hardware cost includes all costs associated with hardware purchasing, such as IoT devices (e.g., sensor), computational devices (e.g., Raspberry Pi) and network devices (e.g., router) [19]. For each workflow activity, hardware costs depend on which resource is

targeted. For example, users who plan to deploy their workflow activity in Edge resources may need to pay for Edge devices, such as a Raspberry pi; in contrast, workflow tasks located in Cloud resources can save this cost in hardware comparatively.

- *Hosting Cost*: Hosting cost represents the cost of provisioning the IoT application on hardware/software services enabling them available for use [19]. For Edge resources, hosting cost primarily include the maintenance cost, such as power consumption cost. Meanwhile, hosting cost in Cloud includes maintenance in various enabling services such as compute, storage, database, and network. Therefore, we consider a workflow activity deployed in Edge will be cheaper than Cloud in terms of hosting.
- *Network Cost*: Network Cost relates to the money spent on managing the networking and resource connections in Cloud and Edge applications [20]. For Cloud resources, the cloud providers offer networking services, like virtual networks, load balancing, application gateways, network monitoring infrastructures, and traffic managers. At the Edge, network cost mainly represents the bandwidth paid for. A major difference is that in Cloud based services such costs are paid on a pay-for-use basis where as in Edge scenarios pre-pay for services in advance is utilised.

2) Resource QoS: Resource QoS refers to the quality of service that both Edge and Cloud resources can provide. It consists of reliability, mobility, heterogeneity, scalability, capability and resource availability.

- *Reliability*: The probability that a service or a system can perform without any failures within a time interval is considered as reliable [21]. In workflow activity deployment scenarios, reliability also represents the high availability of Cloud and Edge resources. Cloud resources can provide full-stack solutions and comprehensive troubleshooting and debugging services (increasing costs).
- *Mobility*: Mobility refers to the ability to migrate and transfer data, services and applications across Edge devices and Cloud servers [22]. Mobility also represents the ability for physical movement of Edge devices and Cloud datacenters.
- *Heterogeneity*: Heterogeneity represents the difference of hardware, software, infrastructures, architectures and technologies of both Cloud and Edge resources [23]. Numerous cloud providers offer services with different technologies and infrastructures. Many IoT device manufacturers together with their propriety solutions result in the Edge infrastructures that are highly heterogeneous.
- *Scalability*: Scalability is provided in two dimensions: Horizontal and Vertical [24]. Horizontal scalability indicates the ability to increase the same type of resources to satisfy load. For example, increasing the number of virtual machines and containers is a type of horizontal scalability. Vertical scalability indicates increasing the capability of an existing service, such as increasing CPU, memory and bandwidth of a virtual machine.
- *Capability*: Capability represents the ability to achieve a requirement [25]. Capability indicates the ability to integrate Edge or Cloud resources and the technologies to align with the users' strategic requirements.
- *Resource Availability*: The percentage of time that a user can access and operate a specific service is considered as resource availability [26]. Additionally, we can calculate the resource availability percentage by using total service time minus the time for which service is not available, then divided by total service time.

3) Data: Data indicates all data related criteria considered in our proposed model that affects the decision making, such as data privacy, data availability, long-term storage, data replication, data integrity, data freshness, and latency.

- *Data Privacy*: Privacy indicates the access control of data maintained by devices and services [27]. They need to remain in charge of their data in spite of third party data.
- *Data Availability*: Availability represents the ability to ensure data can be accessed when required [28]. In both Edge and Cloud resources, users expect to have complete access to their data at all times.
- *LongTerm Storage*: LongTerm storage describes the data that will be stored for a long period, usually in the data centre [29]. In the Cloud, the providers offer various data storage services helping users to store data safely and continuously. For Edge devices, they may have their own storage for temporary data storage.
- *Data Replication*: Data replication provides the ability to store data in more than one database or network node [30]. This promotes data availability and reduces risks of data loss and, ultimately, failure.
- *Data Integrity*: Data integrity represents the accuracy, validity, and consistency of data over the whole lifecycle [28].
- *Data Freshness*: Data freshness indicates the recent nature of data in terms of generation and collection [31]. This helps reduce out-of-order date messaging. In IoT, freshness is a serious concern when dealing with in-stream analysis and management.
- *Latency*: Latency indicates the delay to message response time and the time for data transformation tasks [32]. Latency is influenced by the following factors: geographical location, bandwidth, computational power. For Cloud resources, higher bandwidth along with increased computational power can be employed to minimise latency. However, cloud data centers are always located in a specific geographic location which may be far from users' server (indicating there is little to be done regarding this aspect of latency). However, the geographic positioning of Edge devices can have a significant impact on latency (but they have limited ability to alter bandwidth or processing capabilities).

C. Multi-level Composition Framework

The composition framework of IoTWC works in twofold manner. It first applies AHP-based algorithm to combine both

Algorithm 1: Multi-level Composition

Input: $Pref_{kl}$ – preference of kth criteria over lth criteria, $\mathcal{R}$ – all criterion defined in AHP model, r_{ij} – set of i criteria in first level and j sub-criteria, $Val_{r_{ij}o_n}$ – value for r_{ij} criteria and o_n option, σ_t – a ranking list for tth instance, θ_t – best ranking for tth instance, $\mathcal{B}$ – budget, $\mathcal{C}_u$ – cost of uth instance, $\mathcal{C}_{sum}$ – sum of instance cost

Output: CKR – list of CKRs

```
construct M_kl using Pref_kl
if (!Consistent(M)) then
    Notify user to enter new values
    return -1
else
    W = normalise (principle eigen vector (M))
end
for each r_ij ∈ R do
    multiple W and Val(RO) to get σ_O
    select best ranking θ_t from σ_O for each instance
    CKR = sum of θ_t selection result
end
C_sum = Σ_1^u C_u
if C_sum > B then
    select second best ranking of σ_t for each instance
else
    return CKR
end
```

TABLE I
IOT AHP MODEL PARAMETERS

Notations	Description
$\mathcal{L}$	Level of criteria in AHP
$\mathcal{R} = \{r_{11}...r_{ij}\}$	set of criteria
$Pref = \{Pref_{11}...Pref_{kl}\}$	set of preference of kth criteria over lth criteria
$\mathcal{M} = \{\mathcal{M}_1...\mathcal{M}_s\}$	set of s comparison matrix
$\mathcal{O} = \{o_1...o_n\}$	set of n options in AHP
$Val(\mathcal{RO}) = \{Val_{r_{11}o_1}...Val_{r_{ij}o_n}\}$	set of value for each criteria and options
$\mathcal{W} = \{w_1...w_q\}$	set of weight for q criteria
$\sigma_{\mathcal{O}}$	ranking list for option $\mathcal{O}$
θ_t	best ranking for tth instance
$\mathcal{B}$	budget for IoT application
$\mathcal{C} = \{c_1...c_u\}$	set of u instance cost
$\mathcal{C}_{sum}$	sum of instance cost
CKR	list of CKRs

subjective and objective criteria functions for the selection of CKR and then take into account the financial budget to build an IoT application platform.

For the application of AHP-based algorithm, user need to provide the preference of various criteria (see Section IV-B). Consider there are $\mathcal{L}$ number of levels in the hierarchical representation of the problem and $\mathcal{R}$ is the set of criteria. The preference of criteria k with respect to l, $Pref_{kl}$ is provided following the Satty scale as shown in table II. Next, a comparison matrix $\mathcal{M}_s$ is constructed from the user's preferences following the rule given in equation 1.

$$\mathcal{M}_s = \begin{cases} 1, & \text{when } k = l \\ Pref_{kl}, & \text{when } k > l \\ 1/\mathcal{M}_{kl}, & \text{when } k < l \end{cases} \} \quad (1)$$

Since the user enter value may not be consistent, it is important to check the consistency of each comparison matrix $\mathcal{M}$ constructed using the user's preference values. If the comparison matrix is found inconsistent, user is notified to enter new values, otherwise, the weights are calculated. The weights $w_k \in \mathcal{W}$ for criteria k is calculated by normalizing the principle eigen vector of the respective matrix $\mathcal{M}$. Finally the rank vector $\sigma_{\mathcal{O}}$ of option $\mathcal{O}$ is calculated by multiplying the weight vector $\mathcal{W}$ with the normalized option values $Val(\mathcal{RO})$. This process is repeated for each instance t which can be later combined in the next step to select one complete application instance.

TABLE II
RELATIVE IMPORTANCE VALUE

Importance	Value
Equal important	1
Moderate important	3
Strong important	5
Demonstrated important	7
Extreme important	9
Intermediate	2,4,6,8

In the second level composition, we will calculate each IoT workflow activity cost based on the recommended configurations. For example, the budget column will show the cost when deploying such Edge/Cloud resources for a given period. IoTWC will compare the sum of the IoT workflow activity cost $\mathcal{C}_{sum}$ together with the user recommended budget $\mathcal{B}$ to determine which composed IoT workflow is acceptable. If the sum is over budget, the system will first display the cost result and recommend one or more CKRs. Therefore, IoTWC can perform multi-level compositions to make appropriate decisions under a certain budget and so better reflect a real-world scenario of deployment. The pseudo code of this multi-level composition algorithm is shown in Algo 1.

D. Computational Complexity Analysis

The AHP based IoT workflow composition problem appears complex due to the computational cost of the multi-level composition framework. The computational complexity of our approach is described in equation 2:

$$O(u \times n \times (i \times j^3 + i^3)) \quad (2)$$

In this equation, u represents number of instance in an IoT application, n is the number of options in each instance, $i \times j^3$ and i^3 indicates the complexity of calculating the weight vector for all criteria. However, in our case, i, j, u are concrete number based on given IoT application. Therefore, the final computational complexity can be simplified to $O(n \times i \times j^3)$. Based on the given O, we can expect IoTWC's complexity to be proportional to n, i and j.

V. SYSTEM DESIGN AND IMPLEMENTATION

This section describes the architecture, workflow, and implementation of IoTWC.

A. System Architecture

Fig. 4 shows the architecture of IoTWC. IoTWC is developed as a Web application. The system allows a user to pipeline their IoT application workflow with IoT workflow activities and abstract patterns, along with their specific IoT application requirements for each activity. More specifically,

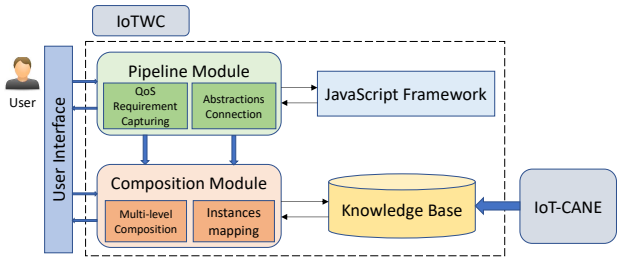

Fig. 4. Schematic Design of IoTWC System

users can easily drag and drop the desired IoT workflow activity from the provided IoT abstraction patterns. The input criteria weights for each section are selected in the similar manner. IoTWC will compose a user's IoT workflow activities together with the recommended configurations. Information entered by users relating to patterns, weights and configurations are stored in an incremental knowledge base to be recalled when required.

The incremental knowledge base is maintained by a well-established system, IoT-CANE [33], which facilitates knowledge acquisition and maintenance. In this system, an incremental method is used to automate a configuration knowledge artifact suggestion based on user requirements within IoT resource configuration management. These recommended suggestions are generated based on users' context information and domain expert edits and modifications. Details of IoT-CANE can be found in [33]. Finally, a composed IoT workflow coupled with configurations will be returned to users via the web interface.

B. System Workflow

Fig. 5 shows the workflow of IoTWC. First, a user is required to input the necessary information through the web interface (step1), such as IoT workflow activities and their associated relationships. IoTWC then initialises by retrieving the appropriate abstract patterns associated with user input (step2). The user input IoT workflow activities are then sent to a javascript based pipeline module (step3) to allow the proposed IoT workflow pipeline to be displayed via the web interface (steps 4 and 5). The user will then specify the criteria weights for each proposed IoT workflow activity (step 6), e.g., weight between Resource Cost and Resource QoS. Once weightings are complete all information is sent to the composition module to allow IoT workflow configuration composition. In this module, an SQL query is constructed from IoT workflow activities (step 7) and their criteria weights suitable for the incremental knowledge base of IoT-CANE (step 8), allowing recommended configurations to be produced (steps 9 and 10). Finally, the recommended configurations are composed into a JSON format file which will be displayed in the web interface (steps 11 and 12).

C. System Implementation

The IoT workflow composition system is implemented and programmed in the Java programming language using the

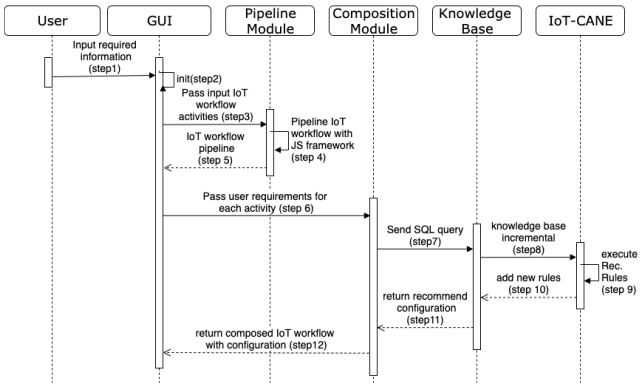

Fig. 5. IoTWC System Sequence Diagram

Spring framework 5.0 [1]The UI (user interface) is created in HTML5, CSS, and JavaScript to ensure "easy to use" website design principles. Our system is designed to help users to pipeline their IoT application workflow into the desired configuration. In addition, this also benefits mapping users' abstract IoT application requirements to real IoT activity instances by applying the AHP based multi-level composition framework. Two main modules (JavaScript (JS) based pipeline module and AHP based composition module) used in our system are now described.

JS based pipeline module. IoTWC allow users to drag and drop IoT workflow activity abstract patterns to position them as required with the aid of the JS-based pipeline module. This module is implemented with GoJS [2],a JS library for building interactive diagrams and graphs. We present a set of IoT workflow activity abstract patterns as reusable components in the system. These components can be freely dragged from the left side of the interface and dropped on the blank workspace on the right side. Links can be built between each IoT workflow activity via mouse clicks, representing connections between Cloud and Edge resources.

AHP based composition module. When the planned IoT workflow activity is clicked by a user, a list of criteria indicating available pairwise comparisons are shown. In the composition module, these pairwise comparisons capture the subjective and objective opinions from users to execute decisions by applying the IoT AHP algorithm. The composition module will calculate the ranking for the alternatives based on user supplied criteria weightings, then compose the appropriate SQL query to retrieve recommended configurations from the knowledge base (IoT-CANE). A Java library is used for vector calculations and score generation. Our implementation provides a simple and easy way for novice users to use IoTWC to display the composed results.

[1]https://spring.io/
[2]https://gojs.net/latest/index.html

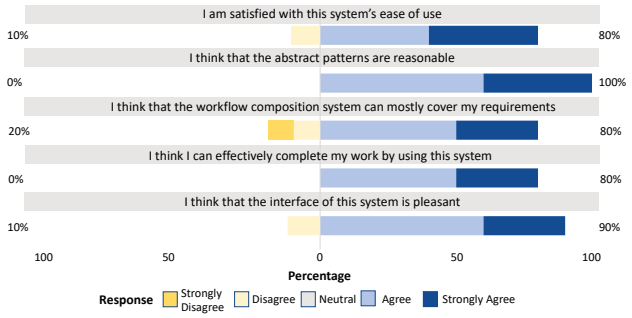

Fig. 6. User Case Study Result

VI. System Evaluation

This section describes the user case study and real world validation of our proposed IoTWC system.

A. User Case Study

This sub-section describes the experimental setup and user evaluation from our user case study.

1) Experiment Setup: As IoTWC is implemented in Java it can be composed into a JAR file to execute over a variety of environments, such as Windows, Linux, and MacOS. In this scenario, we host our workflow composition system on a MacBook Pro with MacOS operating system. The machine has the following hardware configuration: 1.4GHz Quad-Core Intel Core I5 processor, 16GB memory, Intel Iris Plus 1536MB Graphics and 512GB SSD storage. We run our tool using Visual Studio build environment Code[3] with Java and Spring Boot extensions. We use a MySQL[4] database as our data management tool. IoTWC is an open-source system and the current version of code is available on github[5].

2) User Evaluation: In order to evaluate IoTWC, we perform a use case study to ascertain acceptance and performance. We invited twelve participants, who are current Ph.D. or Masters students studying Cloud Computing and/or Internet of Things at Newcastle University. All of these participants have experience and knowledge in Cloud and Edge resource management and deployment. They do not have experience in IoT workflow composition.

After using IoTWC, the participants were asked to complete a questionnaire. Nine questions are used to investigate users' opinions regarding their experiences of IoTWC, listed below.

- How satisfied are you with this system's ease of use?
- How often does the system freeze or crash?
- To what extent do you think that the abstract patterns are reasonable?
- To what extent do the abstract patterns cover your requirements to pipeline IoT workflow?
- To what extent do you think you can effectively complete your composition work using this system?

[3]https://code.visualstudio.com/
[4]https://www.mysql.com/
[5]https://github.com/frankleesd/iot_workflow_composer

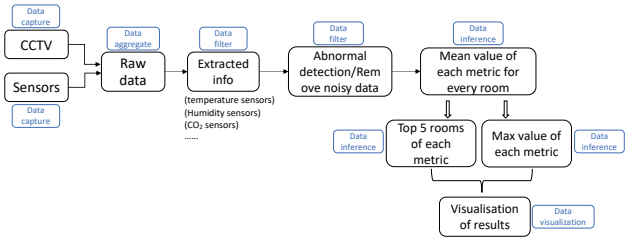

Fig. 7. Smart building scenario workflow

- Do you agree or disagree that the interface of this system is pleasant?
- How likely is it that you would recommend this software to a friend or group member?
- Overall, how satisfied or dissatisfied are you with the workflow composition system?
- How can we improve our IoT workflow composition system?

We chose five questions to show in Fig. 6. As shown in this figure, most of the users were satisfied with IoTWC in areas such as reasonable abstract patterns, and pleasant user interface. According to the feedback, we can summarise that the IoT workflow activity abstract patterns can cover the majority of IoT workflow composition requirements. However, not all of the participants were fully satisfied. Based on the feedback, we can improve IoTWC in the coverage of some alternatives and other criteria.

B. Scenario Validation

This sub-section will validate the effectiveness of the proposed IoTWC using a real world smart building IoT application scenario highlighting the typical usage in an industry style project setting.

1) Scenario Description: In a real world smart build scenario, a user plans to deploy a smart IoT application that can capture relevant data from different room sensors to provide the status of the building. Basic sensors can provide temperature level, humidity level, and CO2 level while more advanced sensors can provide images and other useful information. This smart building IoT application workflow is represented in Fig. 7. In this application, a user wishes to capture CCTV and a variety of other sensor data from rooms while including novel data management/filtering possibilities utilising Edge devices, like a Raspberry Pi. This allows the extraction of raw data into different useful data sets, such as temperature data set, humidity data set and CO2 data set. Abnormal detection and noisy data removal is performed before data inference. In this case, the user plans to calculate the top 5 rooms of each metric and the associated max value of each metric, together with provisioning visualisation of the results.

2) Scenario Validation: First, the user can pipeline the smart building application workflow in IoTWC with given abstract patterns. The pipelined workflow is shown in Fig. 8

When this smart building workflow pipeline is assessed by IoTWC, the user specifies a set of criteria weights in each IoT

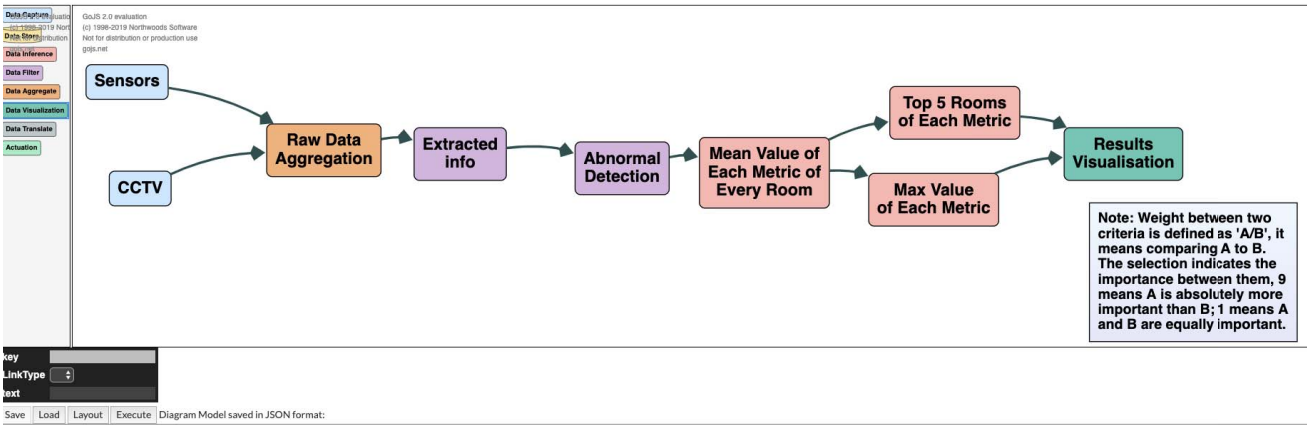

Fig. 8. IoTWC running interface and pipelined workflow

Fig. 9. Configuration Knowledge Representation Example

workflow activity which allows to capture a user's subjective and objective opinion according to his/her requirements. In addition, the user needs to specify a budget for the smart building application. For example, in Raw Data Aggregation workflow activity, a user needs to specify a comparison weights between Resource Cost, Resource QoS, and Data. When these weight information is typed in, a comparison matrix is generated as follow:

$$CKR_{Selec} = \begin{array}{c} \\ R_{Cost} \\ R_{QoS} \\ Data \end{array} \begin{array}{c} \begin{array}{ccc} R_{Cost} & R_{QoS} & Data \end{array} \\ \begin{pmatrix} 1 & 7 & 9 \\ 1/7 & 1 & 3 \\ 1/9 & 1/3 & 1 \end{pmatrix} \end{array}$$

The computation of this comparison matrix can give a ranking for Resource Cost, Resource QoS, and Data.

Additionally, each criteria has some sub-criteria for decision making. For example, a comparison matrix for Hardware, Hosting, and Network Costs is generated as follow:

$$R_{Cost} = \begin{array}{c} \\ C_{Hardware} \\ C_{Hosting} \\ C_{Network} \end{array} \begin{array}{c} \begin{array}{ccc} C_{Hardware} & C_{Hosting} & C_{Network} \end{array} \\ \begin{pmatrix} 1 & 1/3 & 9 \\ 3 & 1 & 5 \\ 1/9 & 1/5 & 1 \end{pmatrix} \end{array}$$

These two comparison matrix are generated based on the information provided by the users. In the meantime, the comparison matrix between criteria and alternatives are defined and composed by domain experts. For example, considering scalability sub-criteria under resource QoS, Cloud resource may get more weight than Edge resource due to the scalability

and services offered by cloud providers. The comparison matrix is shown below:

$$Scalability = \begin{array}{c} \\ Cloud_1 \\ Cloud_2 \\ Edge_1 \\ Edge_2 \end{array} \begin{array}{c} \begin{array}{cccc} Cloud_1 & Cloud_2 & Edge_1 & Edge_2 \end{array} \\ \begin{pmatrix} 1 & 1 & 9 & 9 \\ 1 & 1 & 9 & 9 \\ 1/9 & 1/9 & 1 & 1 \\ 1/9 & 1/9 & 1 & 1 \end{pmatrix} \end{array}$$

Other comparison matrix are constructed in the similar manner. As a result of AHP execution, a list of best ranking for each instance are calculated by previous processes. Once the 'execute' button is clicked the IoT AHP algorithm starts querying the knowledge base to gain a set of appropriate configuration knowledge representations for each IoT workflow activity. The results are displayed to the user for further consideration. An example JSON format CKR result of one IoT workflow activity is shown in Fig. 9.

VII. Conclusion and Future Work

IoT workflow composition is a complex problem due to the heterogeneity of Cloud and Edge resources and data type diversity. We proposed and developed a novel AHP based multi-level composition framework and IoTWC system. With IoTWC, IoT application users can easily pipeline and compose IoT workflow application with recommended activity configuration knowledge representations under a certain budget. The results are investigated and validated in a real world smart building scenario. The results can be further utilized by user to deploy the workflow instance while making a balance between the performance and budget for each workflow activity. In the future, we plan to design and develop a deployment module to integrate with IoTWC, in order to automate IoT application development life cycle. We will provide an extension of IoTWC to cover deployment and orchestration processes in IoT applications.

Acknowledgment

This research is supported by the following projects: Osmotic MindSphere: P35792/BH192113, FloodPrep: NE/P017134/1.

References

[1] M. Kranz, P. Holleis, and A. Schmidt, "Embedded interaction: Interacting with the internet of things," *IEEE internet computing*, no. 2, pp. 46–53, 2009.

[2] D. N. Jha, P. Michalak, Z. Wen, P. Watson, and R. Ranjan, "Multi-objective deployment of data analysis operations in heterogeneous iot infrastructure," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[3] M. Dimitrov, A. Simov, S. Stein, and M. Konstantinov, "A bpmo based semantic business process modelling environment," in *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM-2007)*, vol. 251, 2007, pp. 1613–0073.

[4] D. Chiu, T. Hall, F. Kabir, and G. Agrawal, "An approach towards automatic workflow composition through information retrieval," in *Proceedings of the 15th Symposium on International Database Engineering & Applications*. ACM, 2011, pp. 170–178.

[5] P. Asghari, A. M. Rahmani, and H. Haj Seyyed Javadi, "A medical monitoring scheme and health-medical service composition model in cloud-based iot platform," *Transactions on Emerging Telecommunications Technologies*, p. e3637.

[6] P. Partheeban and V. Kavitha, "Versatile provisioning and workflow scheduling in waas under cost and deadline constraints for cloud computing," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 1, p. e3527, 2019.

[7] G. Nikol, M. Träger, S. Harrer, and G. Wirtz, "Service-oriented multi-tenancy (so-mt): enabling multi-tenancy for existing service composition engines with docker," in *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, 2016, pp. 238–243.

[8] P. Persson and O. Angelsmark, "Calvin–merging cloud and iot," *Procedia Computer Science*, vol. 52, pp. 210–217, 2015.

[9] K. Hightower, B. Burns, and J. Beda, *Kubernetes: up and running: dive into the future of infrastructure*. " O'Reilly Media, Inc.", 2017.

[10] Amazon. [Online]. Available: https://aws.amazon.com/cloudformation/

[11] T. L. Saaty, "Analytic hierarchy process," *Encyclopedia of Biostatistics*, vol. 1, 2005.

[12] N. Glombitza, S. Ebers, D. Pfisterer, and S. Fischer, "Using bpel to realize business processes for an internet of things," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2011, pp. 294–307.

[13] D. Domingos, F. Martins, C. Cândido, and R. Martinho, "Internet of things aware ws-bpel business processes context variables and expected exceptions." *J. UCS*, vol. 20, no. 8, pp. 1109–1129, 2014.

[14] G. Chen, J. Huang, B. Cheng, and J. Chen, "A social network based approach for iot device management and service composition," in *2015 IEEE World Congress on Services*. IEEE, 2015, pp. 1–8.

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

[16] R. Chen, J. Guo, and F. Bao, "Trust management for soa-based iot and its application to service composition," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 482–495, 2014.

[17] T. Baker, M. Asim, H. Tawfik, B. Aldawsari, and R. Buyya, "An energy-aware service composition algorithm for multiple cloud-based iot applications," *Journal of Network and Computer Applications*, vol. 89, pp. 96–108, 2017.

[18] A. C. F. da Silva, U. Breitenbücher, K. Képes, O. Kopp, and F. Leymann, "Opentosca for iot: automating the deployment of iot applications based on the mosquitto message broker," in *Proceedings of the 6th International Conference on the Internet of Things*. ACM, 2016, pp. 181–182.

[19] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE transactions on services Computing*, vol. 5, no. 2, pp. 164–177, 2011.

[20] S. A. Karthikeya, J. Vijeth, and C. S. R. Murthy, "Leveraging solution-specific gateways for cost-effective and fault-tolerant iot networking," in *2016 IEEE Wireless Communications and Networking Conference*. IEEE, 2016, pp. 1–6.

[21] H. Madsen, B. Burtschy, G. Albeanu, and F. Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable fog computing," in *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2013, pp. 43–46.

[22] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's intranet of things to a future internet of things: a wireless-and mobility-related view," *IEEE Wireless communications*, vol. 17, no. 6, pp. 44–51, 2010.

[23] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[24] D. Zhang, L. T. Yang, and H. Huang, "Searching in internet of things: Vision and challenges," in *2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications*. IEEE, 2011, pp. 201–206.

[25] X. Yu, B. Nguyen, and Y. Chen, "Internet of things capability and alliance: Entrepreneurial orientation, market orientation and product and process innovation," *Internet Research*, vol. 26, no. 2, pp. 402–434, 2016.

[26] S. C. Mukhopadhyay and N. K. Suryadevara, "Internet of things: Challenges and opportunities," in *Internet of Things*. Springer, 2014, pp. 1–17.

[27] C. Perera, R. Ranjan, L. Wang, S. U. Khan, and A. Y. Zomaya, "Big data privacy in the internet of things era," *IT Professional*, vol. 17, no. 3, pp. 32–39, 2015.
[28] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.
[29] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.
[30] A. Kumar, N. C. Narendra, and U. Bellur, "Uploading and replicating internet of things (iot) data on distributed cloud storage," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE, 2016, pp. 670–677.
[31] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless internet of things," in *2015 international conference on recent advances in internet of things (RIoT)*. IEEE, 2015, pp. 1–6.
[32] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency internet of things," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 39–45, 2018.
[33] Y. Li, A. Alqahtani, E. Solaiman, C. Perera, P. P. Jayaraman, R. Buyya, G. Morgan, and R. Ranjan, "Iot-cane: A unified knowledge management system for data-centric internet of things application systems," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 161–172, 2019.