An Adaptation Middleware Supporting Continuous Adaptation Of End User Driven Application Composition Framework

Sakhawat Hossain

(153-35-1335)

A thesis submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Software Engineering

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

Fall – 2019

APPROVAL

This **Thesis** titled "**An Adaptation Middleware Supporting Continuous Adaptation Of An End User Driven Application Composition Framework**", submitted by **Sakhawat Hossain**, **153-35-1335** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Software Engineering and approved as to its style and contents.

DECLARATION

I hereby, declare that I have taken this project under the supervision of Mr. K.M. Imtiaz-Ud-Din, Assistant Professor, Department of Software Engineering, Daffodil International University. I also declare that neither this thesis nor any part of this has been submitted elsewhere for award of any degree.

Sakhawat
11-12-19

………………………………………

Sakhawat Hossain

ID: 153-35-1335

Batch: 18th

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

Certified by:

[illegible] 11/12/2019

………………………………..

K.M. Imtiaz-Ud-Din

Assistant Professor

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

ACKNOWLEDGEMENT

Every good approach starts with the initiative of some person who has the vision to do something amazing. This research is a combination of some good ideas and there are many people who are behind those ideas. I hereby will be thankful forever to my supervisor K.M. Imtiaz-Ud-Din, Assistant Professor, Department of Software Engineering, Daffodil International University for the guideline he provided me from the starting of this research till today. His continuous monitoring always kept pushing me to think precisely and not to lose hope even if something not working properly. It was always motivating to work with my respected supervisor. If i had the chance or ability to work with some research works, I will definitely go along with my supervisor Mr. K.M. Imtiaz-Ud-Din. I also want to thanks my lab members who directly or indirectly helped me to complete my research work. I will say that without their help it would be so much tough to complete my research work.

I am also thankful to our honorable Professor Dr. Touhid Bhuiyan, Head of the Department of Software Engineering, Daffodil International University who always encourage us to do good works and all the other teachers in the Department of Software Engineering. I am also very grateful to those researchers who works helped me to find relevant things to my research and find the place where I can contribute more. Finally, I want to thank my parents because they are the reason behind all of my success. Without the blessings of my parents I would not able to come this long. Specially to my mother Mrs. Sabina Yasmin who is still trying her best to reach my goal.

Table of Contents

APPROVAL i

DECLARATION ii

ACKNOWLEDGEMENT iii

Table of Contents 2

List of Figures 4

ABSTRACT 5

CHAPTER I INTRODUCTION 6

1.1 Background 6

1.2. Objective 7

1.3. Research Method and Outline of Thesis 9

CHAPTER II LITERATURE REVIEW 11

2.1. Taken from Existing Works 12

2.2 Related Technologies and Frameworks 12

CHAPTER III PROPOSED ARCHITECTURE 14

STEPS AT A GLANCE 15

3.1. Service Repositories 17

3.2. Service Registry 17

3.3. Watcher 17

3.4. Block Generator 17

3.5. Service Creator 18

3.6. Interface Creator 18

3.7. Service & Interface Connector Creator 18

3.8. Meta Blocks 19

3.9. Interactive Application Creator 19

3.10. Generated Blocks & Services 19

3.11. Other Blocks 20

3.12. Composite Composition 20

3.13. Orchestration Service 20

3.14. End-User Development Interface and Code Generation Engine 20

PROPOSED ARCHITECTURE 22

3.15. Application Manager 26

3.16. Context Manager 26

3.17. Variant Manager 27

3.18. Adaptation Engine 27

3.19. Runtime Engine 28

CHAPTER IV EVALUATION 29

CHAPTER V CONCLUSION 33

5.1. Limitation 33

5.2. Future Work 34

REFERENCES 35

APPENDIX A 36

APPENDIX B 36

APPENDIX C 37

APPENDIX D 40

APPENDIX E 43

APPENDIX F 45

List of Figures

Figure 1 . Research Methodology 9
Figure 2 . Previous Team Architecture 14
Figure 3 . Proposed Architecture 22
Figure 4 . Proposed Architecture Only Present Work 23
Figure 5 . User making composition with services 29
Figure 7 . News Result for Sunny 30
Figure 8 . News result Rainy 30
Figure 9 . Working flow of each part 31
Figure 10 . Continuous adaptation of services 32

ABSTRACT

The usage of IOT devices in our daily life is increasing at a good rate. By IOT here it means the sensor and actuators that we use to satisfy our needs. But when we think by the perspective of a user we always want things in our own ways. Where everything will be flexible like if we want to add or remove certain service we need to have the capability to do so. Precisely if a user wants to add a new sensor without having programming knowledge, he or she cannot do that because most of the services that different organizations provide are not configurable. Yet as a user we cannot to that because there are some complexities when we say configurable services. Researchers are trying to provide user these sorts of configurable services, yet we are not able to provide these sort of services right now. So, this is a huge scope or place to work on because whenever a problem arises there is a solution for it and we need to find it and utilize that solution at its best.

The past researchers of "Ambient Intelligence Lab" proposed an architecture where the end-user can plug sensors and actuators without the help of professionals. They also provide the examples to emphasize that the architecture works properly or the way it should have worked. But they had some lacking too. One of them was having no adaptive environment. By adaptive environment we meant that our system should listen to the different context of different services that user use to fulfill his/her requirements. An Adaptation Engine with the help of some additional components it provides the user an ultimate flexibility where the end user just provides his/her queries by using service composition and our system will always listen to the different contexts of those services which leads our system as a continuous adaptation system.

CHAPTER I INTRODUCTION

1.1 Background

The thing which most effects our life is IOT services. It just making our life simpler and easier because it provides the simplicities and functionalities that we need in our life. When we say IOT it just does not mean only the hardware devices rather it summarizes the relation between both the hardware and software because they are both dependent and can work simultaneously. But most of the services that we used like home automation, office managerial etc. these things are provided by some companies. They provide these solutions. Every different company has different services that they offer to their user. But all these services are rigid. By rigid we meant that these services are not configurable by the user themselves. And the main thing to point here is there no existing technology that satisfies the adaptation of environment of a system. Adding with that, only the professionals who has the knowledge to alter those devices can perform the configuration. So user has this uncomfortable zone not being configure his/her own services.

1.1.1 Scenario

We will try to describe our scope of this thesis by providing a scenario based problem and later we will try to collect requirements from that scenario.

Suppose Mr. David wants to view the email based on the current weather. And he wants a composition where he will attach web service and other service like reading newspaper. So he uses our previous system to make service composition along with our web service. What he wants is if the weather is rainy he wants to know the news headlines. And if the weather is not rainy he doesn't want to do any sort of staff.

But Mr. David wants to make his composition as an interactive application which will always observe the weather like there will be times when the weather turns rainy to sunny or sunny to rainy etc. But there are industrial or research works which can satisfy this adaptation along with IoT sensor and actuator. But we know that everything is possible if we try. So to deliver an adaptive environment along with IoT sensor and actuator we need bunch of web services and their respective context values. And to do this we also need an architecture which will provide the desired output.

1.2. Objective

The purpose of this thesis work is to provide an end-user an adaptive environment where the system can listen or observe any kind of environment changes. We tried to build an architecture which will adopt the changes that occurs into an environment. An adaptive application provides user the desired output by looking up the context values or what's the current context of the services that user uses. To understand the goal of this thesis more clearly let's dive in the scenarios problem described in section 1.1. Following are the requirements that being collected from that scenario:

1.2.1 Requirements

Firstly, we need an environment where we can manage the service compositions that users are using. Because there might be several users using our system and they have their respective requirements. In order to fulfill their requirements, they will use different service composition and interactive application. The environment needs continuous adaptation so we also need a separate environment where the continuous adaptation will occur. Because we always tend to divide services as loosely coupled as we can. And there will be a co-relation between the management thing and adaptation environment. And last of all we will be required a place or engine which will be responsible for the execution of user interactive applications. Which means system should be able to break the barrier between physical and virtual world.

Overall our objective is to crossing the boundary between the virtual and physical world. Providing users flexibility to create application on the basis of his/her needs without having programming knowledge and our system will have the ability to observe the environment changes. Before this work our previous lab members worked on making or giving the user to config his/her services. But the main intention of this research work it to provide an end-user an adaptive environment where user does not need to interact with the system whenever the environment changes.

1.3. Research Method and Outline of Thesis

In order to emphasize our goal as efficient as possible we needed to follow a research lifecycle which can lead our work in the right way.

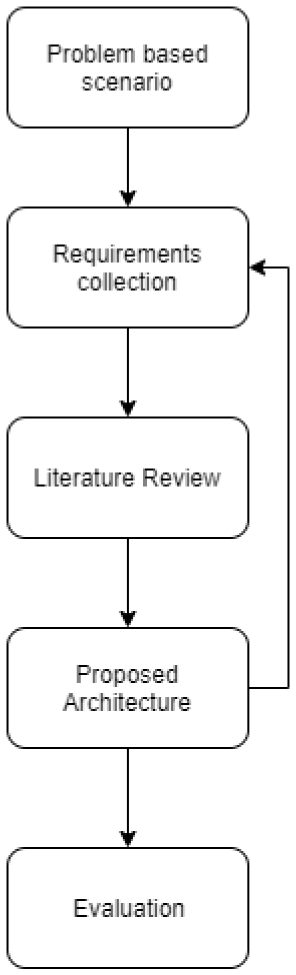


Figure 1 . Research Methodology

Every thesis or research has made or carried out to solve a solve that was not solves or even though before. We found this thesis interesting while we faced the scenario that we described in Section 1.1.1. And whenever there are some problems there might be some solutions too.

By the time we were able to figure out the issues that arises we break down the problems from the largest to the smallest one which will later be helpful for us to understand and design the problem statement properly. The problems and requirements statements were covered properly in the Section 1.2.1.

When we figured out the problem statement properly then we spent some time to find some better solutions which possibly could solve the problems and the solution must have the capability to engage more the end user with their IoT sensors and actuators and other services. And then we have created research lifecycle which have been followed thought out the research.

We all know that every whenever there are some problem that raised and needs to provide solution we intend to divide those or that problem into separate segments so that we can work more efficiently. That is why we follow the methodology that we have attached in Figure1.1. We also need to follow the work that might be previously done by other researchers which will help us to find our solution and let us compare our solution to theirs. The best help that we got is from out lab “Ambient Intelligent Lab”, the work of all previous groups summarizes a good amount of work and their help was emerging while implementing our thought to solve the problem.

We have prepared an architecture in which our system lies on. Every system has its own working flow and in order to maintain a perfect working procedure we prepared a well-furnished system architecture which is attached into the next section. We followed that model throughout our whole system. Every research work has theoretical segment and there must have an implementation whether the proposed solution effects the user or not. For this reason, we attached out proof of concept in Chapter 4, where we tried to show that our work actually satisfied the users need. After that section we tried to carry out the Appendix part which is attached into the section 5.

CHAPTER II LITERATURE REVIEW

To start the work of our thesis we tried to take out some observation on the work that has been done previously which goes along with our working area. Although there are some scopes in which some good work been done before. So we tried to review and use those work both of research and implemented based, in a sense we can use them as per our requirements. Let's point some of the work like as visual computer programing language like as Blockly, micro service architecture along with the loosely coupled scenarios, IoT sensors and actuators service along with web service like as third party api, multi-agent environment, distributed architecture etc. We had to take some of these work in order to establish a proper architecture which made our system more efficient. And we always believe in a product which is fully functional so we provide our architecture first then we tried to implement our architecture by the help of existing technologies.

The most recent works regarding on context aware or interactive application is the paper titled as *Context-Awareness for Self-Adaptive Applications in Ubiquitous Computing Environments* by Kurt Geihs and Michael Wagner. They tried to explain the how context aware application can affect the user mobility at a lower level to the highest. Another paper titled as *Supporting Adaptive Application Mobility* by Francis M. David, Bill Donkervoet, Jeffrey C. Carlyle, Ellick M. Chan, Roy H. Campbell. They also tried to establish an architecture to provide the opportunity for applications to better adapt their user interface to the new environment. Although they tried to follow two more approaches, one is Model-View-Controller design pattern which provides more efficiency into the system and another one is JADE mobile agent platform. Its worthy to mention another important paper titled as *A Survey of Approaches to Adaptive Application Security*, although their main focus was to provide a clearer view about how we can provide more security in adaptive application.

2.1. Taken from Existing Works

As, this is an extended work on our existing lab members, we had to inherit some things that had been done before by our previous groups. We needed to use Blocks [Made along with visual programming language Blockly], Compositions, Services [sensor service, actuator service and web services] on which our previous lab members worked on. We will add some new things along with previous groups, like as Application Manager, Context Manager, Variant manager, Adaptation Engine and Runtime Engine which will make the work more valuable. I believe, this work can change the problem that an end-user faced in the scenario and at the end of this thesis work an end-user can have an adaptive environment along with IoT sensor and actuator and other web services.

2.2 Related Technologies and Frameworks

In order to fulfill the vision, we need to use some technologies that are already available. And they are,

- Blocky: Blocky is a client-side JavaScript library for creating block. Worth mentioning that it is a visual programming language which is widely used by other developers and researchers out there.
- Django: Django is a Python-based free and open-source web framework, which follows the model-view-template architectural pattern.

- Ajax: we will need this to create asynchronous web applications and it will also be helpful for communicating data between our web application and database.
- HTML/CSS/Bootstrap: To design a better user interface.
- jQuery: jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax

CHAPTER III PREVIOUS AND PROPOSED ARCHITECTURE

Before proposing our architecture to provide solution on the scenario problems. Let's see what we have already in our hand. We have attached the previous team architecture in the below and later section we discussed about each component. This part of this document is a very important one because we will try to dive in what our previous groups did. Then we can find out what can be done to fulfill the problem that we collected from the above scenario. We will provide a latest solution for the scenario problem later of this section.

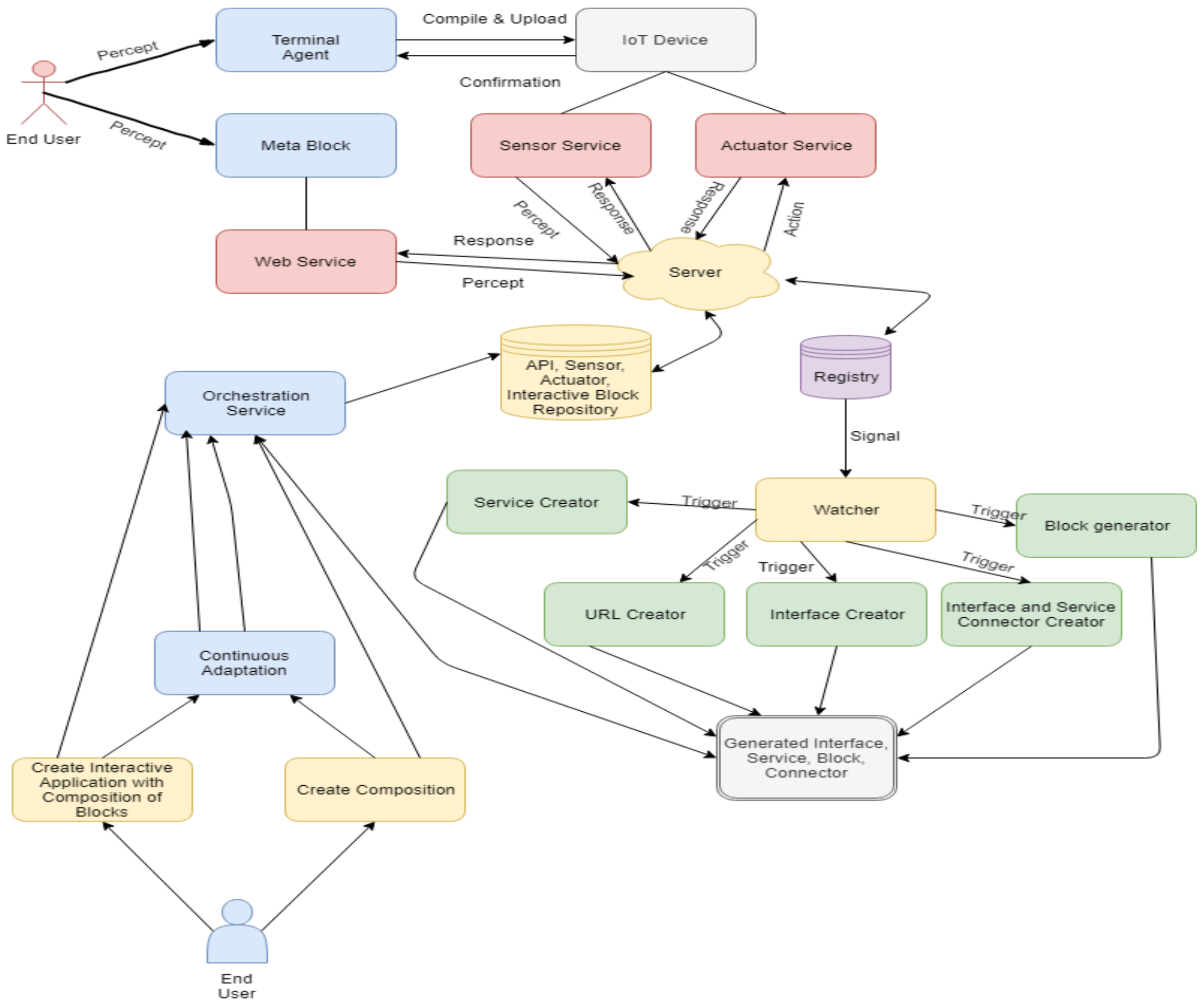
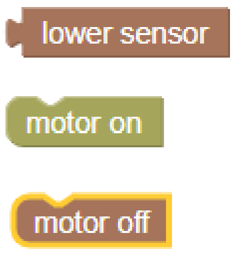


Figure 2 . Previous Team Architecture

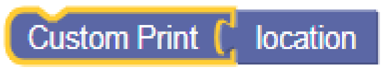
STEPS AT A GLANCE

Let’s see how the previous architecture act when a user wants to run service composition.

- Firstly, End-user create composition with the help of our existing services.
- Then the service composition gets segmented through Orchestration service. It is a sort of navigator which lies in between our Service Repository and Generated Interface, Service and Block creator.
- Our service repository holds the information related to the components i.e. Api, sensor, actuator etc. Following are some of the services for sensor and actuator.



- Our services have interfaces to interact with user. Suppose an end-user wants to use location service. There is an interface in our previously implemented system where user can use it. Following is a simple demonstration of it, which will print the location information.



- All of the points mentioned above is for those service which are registers in our system. Whenever a new service is attached like and end-user beings a sensor to control his/her door. So he/she plugs the sensor into our system. At that point it is a new sensor and we need to create services against that sensor. So that is when watcher comes to play. Watcher always watches over our service repository to check if there is any new service. As a new sensor been plugged into our system watcher instantly informed different service components. It will call block generator to generate block, interface creator to create interface, service creator which will execute against the generated interface.

In the following pages we have elaborate all this components.

3.1. Service Repositories

It is basically the repository or storage where the sensor, actuator, interactive block, web services and user preferable composition will be stored. Later we will use their references to work on. It

3.2. Service Registry

Whenever a user wants to use a service with blocks to generate services, those services need to registered from our Service Repositories first and later the watcher watches on those services.

3.3. Watcher

Watcher does the obvious thing related to its name. It always watches the Service Registry table to check whether there is any new service or not. If there is a new service arrives then it calls some other pointers like Block Generator, Service Creator, Interface Creator, Service Middleware etc. We can call it a simple intelligent agent which just watches over a component and if it finds any changes on the components then it notifies other respective components to avail all the services.

Other's additionally call their own respective block engenderer, accommodation engenderer and etc.

3.4. Block Generator

Block Generator is responsible for generating blocks for different kinds of services like api, mailbox, sensor, actuator or web services. These blocks also have the interactions with user whereas in api service the api blocks will have some extra input field. And the other like web, sensor, actuator or mailbox has their own respective inputs because we gave the user ultimate flexibility. These all are part of the previous groups works and I didn't alter those work.

3.5. Service Creator

Service Generator will be responsible for generating services against the blocks we created like api blocks will have api services or sensor blocks will have its own sensor service and so as web blocks will have web services. And all these services constructed with the help of micro service architecture.

3.6. Interface Creator

Here by only interface we tried to meant the User Interface. Two types of interfaces are available in our system. One is general blocks which needs inputs and interface will help to take inputs from the end user and another is conditional block which don't need any inputs so we don't need interface for those blocks and conditional blocks just displays the output message or result.

3.7. Service & Interface Connector Creator

Service and interface connector creator will only be called whenever it gets positive result from watcher like a new service has arrived. After that it generates some functions against those service and those functions will later be used for data communication.

3.8. Meta Blocks

Meta Blocks are responsible for creating a block which will have additional blocks into it. Like combining several blocks into a block. Our system has two meta block as default. One is API Creator which creates api by taking necessary inputs and another one is interactive Block Generator which will try to generate Interactive Application.

3.9. Interactive Application Creator

Interactive application can only work with several services. We provided some place where user can drag and place their services to make an interactive application. User can create several interactive applications there is boundary on it but at a time they can run or use only one interactive application. After making an interactive application with several services user just need to press create app button which will automatically create the necessary interfaces and connectors to run the interactive application.

3.10. Generated Blocks & Services

By Generated Blocks and Services, we meant that there will be some blocks related to API, sensor, actuators and Mailbox. And the input of these blocks and services depends on the end user. End user can use single or multiple services to make a composition.

3.11. Other Blocks

There are some other blocks like conditional blocks, mathematical term blocks, blocks related to email like [email read, label, search]. But most of these will be the default block provided by Blockly Library and these block has their own structure which cannot be changed. And user may have the permission to interact with them not to write them.

3.12. Composite Composition

A composite (Imtiaz-Ud-Din, K. M., 2011) composition is created by adjoining multiple tasks or services. If we want to elaborate Service Composition Concepts and Notation then we can say it is a composite combination can be made of triggers, services, conditions or/and queries.

3.13. Orchestration Service

This component helps to navigate between sensor or actuator and user. Whenever the value of a sensor or actuator changes based on the end users need our Orchestration Service will fire and it will change the value for sensor or actuator.

3.14. End-User Development Interface and Code Generation Engine

End-user development interface refers to the term (Graphical Interface). Which will be helpful for an end user to interact with our system. And which will be much easier to understand because all the things will be clearer when there is a good development interface. End-user can customize their own services as per their needs and can use i.e. used visual programming language Blockly which we find more catchable by the end-user.

PROPOSED ARCHITECTURE

Above we describe what our previous groups did and what they offered the end-user. But there was a place where the previous architecture cannot function properly. We are talking here about a continuously adaptive environment. Which will always react in accordance with the latest information from the environment. A continuous adaptive environment will always respond to the environment changes. Our architecture looks like,

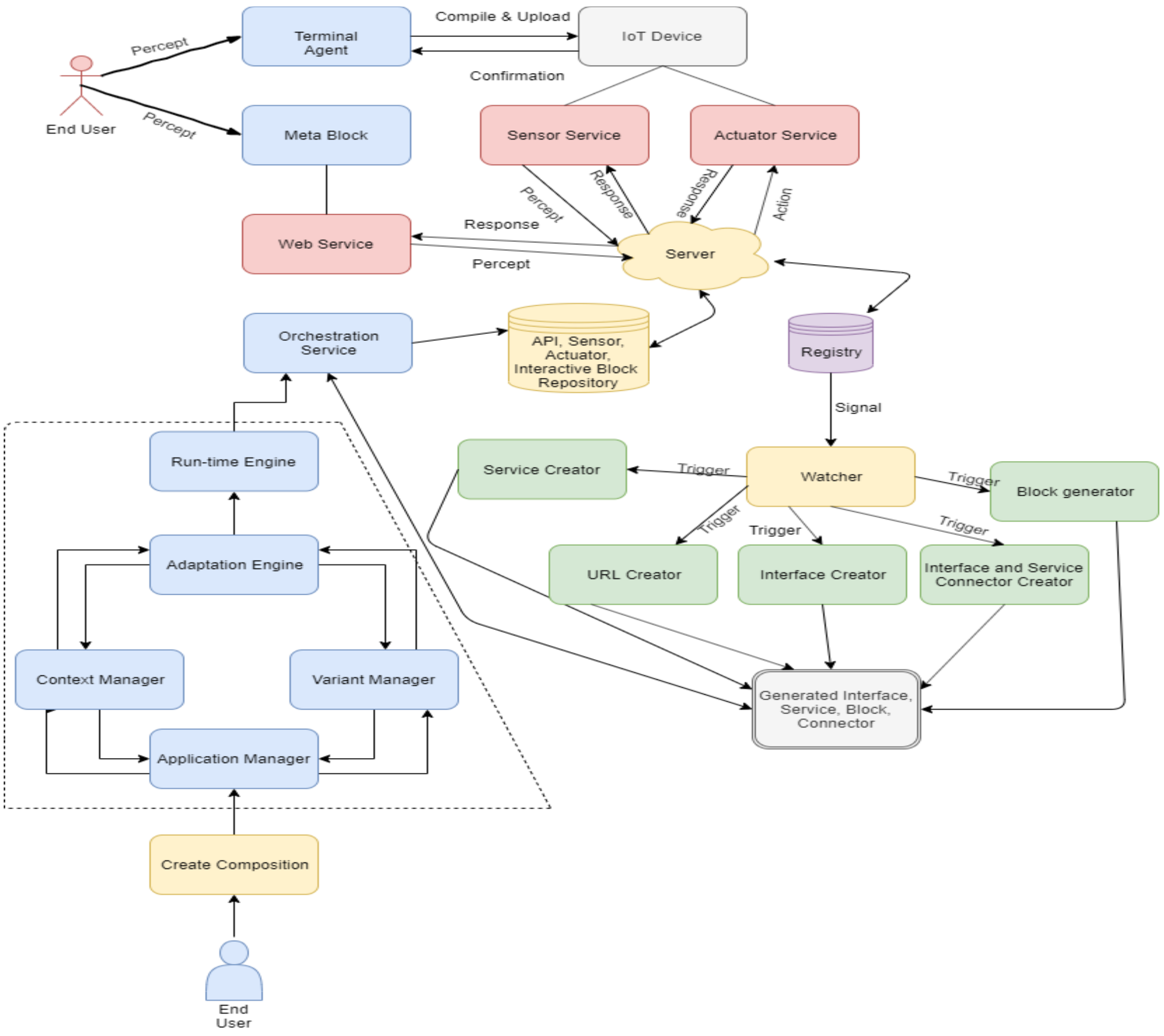


Figure 3 . Proposed Architecture

Figure 3 shows a complete picture about our proposed architecture. In Figure 4 we will try to demonstrate a workflow about how our proposed architecture works within their own components.

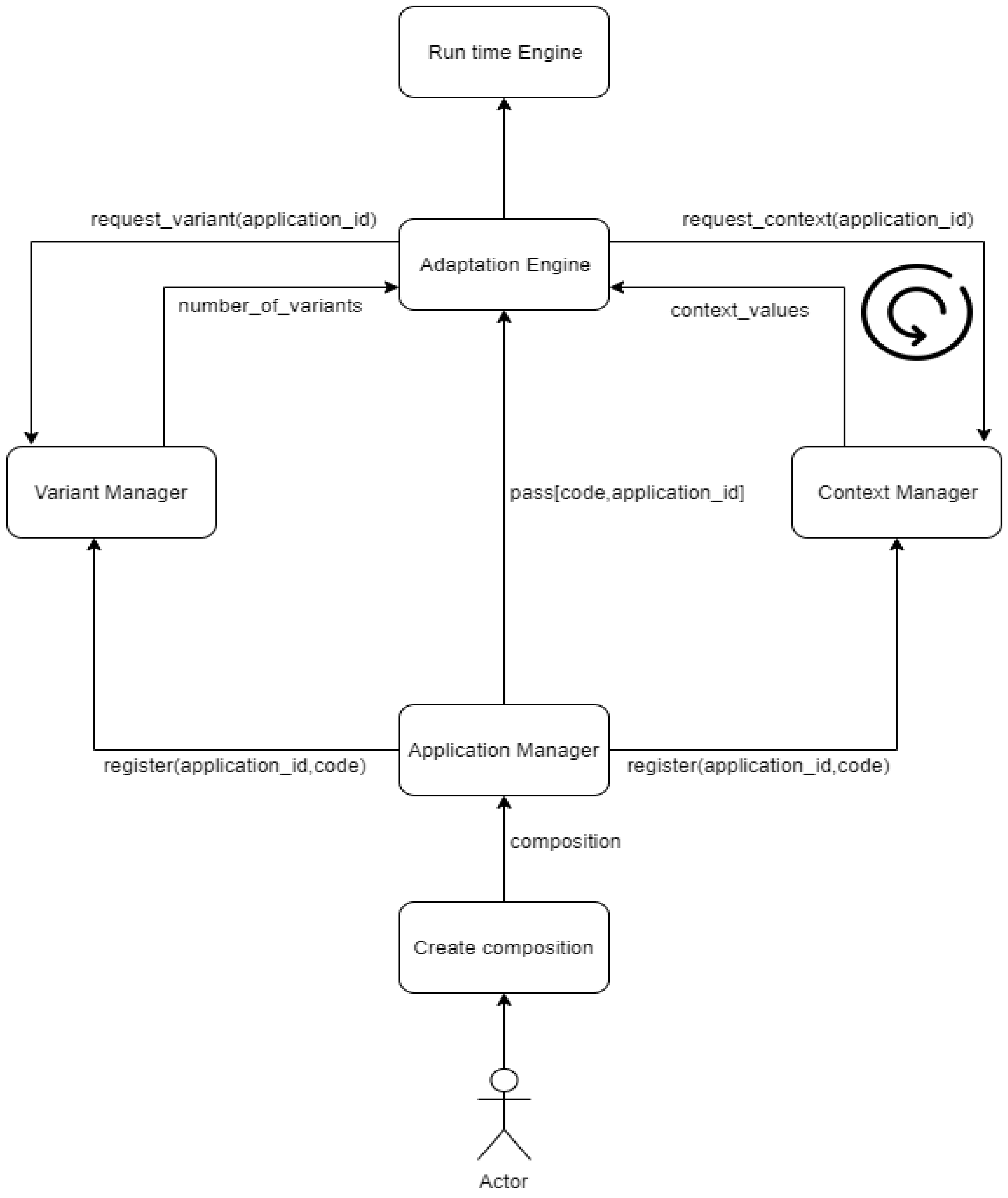


Figure 4 . Proposed Architecture Only Present Work

Following are some of the steps that our architecture follows to fulfill the scenario problem.

Steps here are:

- At the very first step Application Manager receives the user composition. After that Application Manager generates a unique application id for that composition. And finally it sends two request, one is to Context Manager to register the application along with application id and code and another one is to Variant Manager along with same application id and code.
- Secondly Context Manager does four important things. First one is to register an application with application id and code. Second one is it can unregister an application with further request. Third one and most important one is it reads the code and finds out the different contexts and store those contexts. Finally, it can watch over those contexts to receive required context values only if Adaptation Engine requests for.
- Third step is into Variant Manager. Variant Manager has also four important jobs. The first one is, it can register an application with application id and code. Second one is can unregister an application. Third one and which is an important one and that is, it separates conditional service and normal services by reading the code. Lastly it provides all the conditional services and non-conditional services to Adaptation Engine as per request.
- In fourth step Adaptation Engine receives an application id along with the code. It creates a mapped value where [context, required context and conditional service] these three components are mapped against each other. Later it requests context manager to get context values and variant manager to get all the variants. The request between

Adaptation Engine and Context Manager remains in a continuous process. Finally, Adaptation Engine picks the right variant from the mapped value based on the context and passes that variant to Runtime Engine for further processing.

- Runtime Engine receives the variant from Adaptation Engine. Runtime engine then extracts that variant and make that variant executable. And finally it executes that variant and gets the result.

In the following page we have added more elaborate description for each component. And in addition we have added this working flow in Figure7.

3.15. Application Manager

Application manager does things as its name suggest. It receives the composition that user created. It generates some unique id for each application that it receives in order to track the application. After id generation it does two more things. One is it request the Context Manager to register the application with its id and services.

Another one is it request another registration to Variant manager with application id and some code. Which will later use by Adaptation Engine. Finally, it passes application id and code to adaptation engine so that adaptation engine may perform its query. We have attached how our Application Manager looks like in Appendix A.

3.16. Context Manager

Context Manager always done two major things. One is to register the services along with the application id that it gets from Application Manager. And the other thing is, it will watch over that registered service always and will return the current context value to Adaptation Engine. And in addition it can unregister some applications contexts. There is a code snippet in Appendix C where we showed how Context Manager will work.

3.17. Variant Manager

Variant Manager does two important things. The first one is, it registers with the user service composition and application id that it gets from Application Manager. And the second things it does, is it calculate how many compositions can be occurred based on the number of Conditional Step and Step and returns those variant to Adaptation Engine. It also has the unregister functionality to unregister an application and Variant manager can unregister a service only when end-user request to do so. The role of variant manager along with application manager and adaptation engine is mentions in Appendix D.

3.18. Adaptation Engine

Adaptation Engine receives application id and user used composition from Application Manager. Later it calls Context Manager and Variant Manger parallels with the application id. Later Context Manager and Variant Manager provides the respective context values and variants. After the context and variant retrieving process Application Manager selects the right variant based on the context and pass it to Runtime Engine. So it has relation with all the components like as Application Manager, Context Manager, Variant Manager and Runtime Engine. Finally, adaptation engine can unregister an application if the user wants to stop i.e. have added the necessary steps that adaptation engine does in Appendix E.

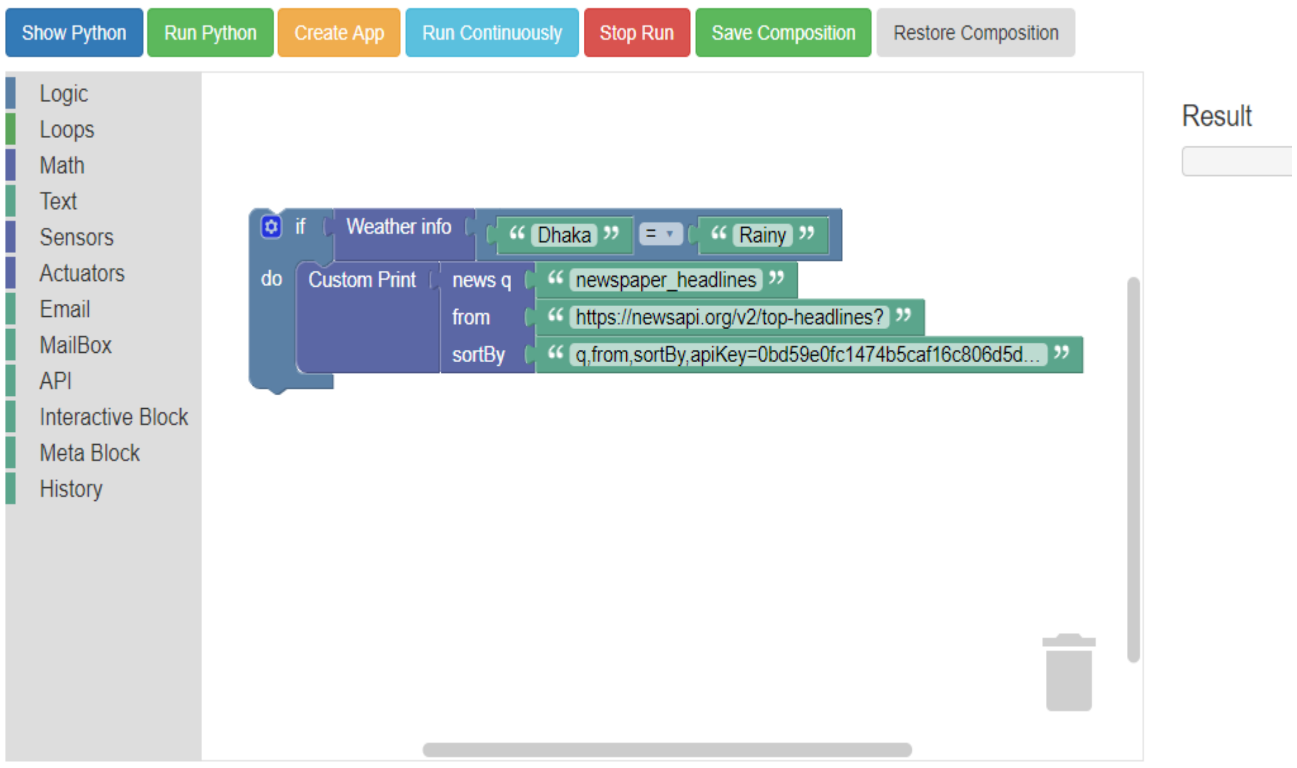
3.19. Runtime Engine

Runtime engine has only connections with adaptation engine. It receives the variant from adaptation engine. And runtime engine has two tasks. One is make the variant executable and second one is show the result to the end-user with the help of console. We have attached the code snippet of runtime engine in Appendix F.

CHAPTER IV EVALUATION

In this chapter we will show that our work satisfies the end-user needs to observe and adopt an environment. According to the scenario user wants to creates a service which will have weather and some other services and we will shoe here that the composition that he or she used will adopt while the environment changes.

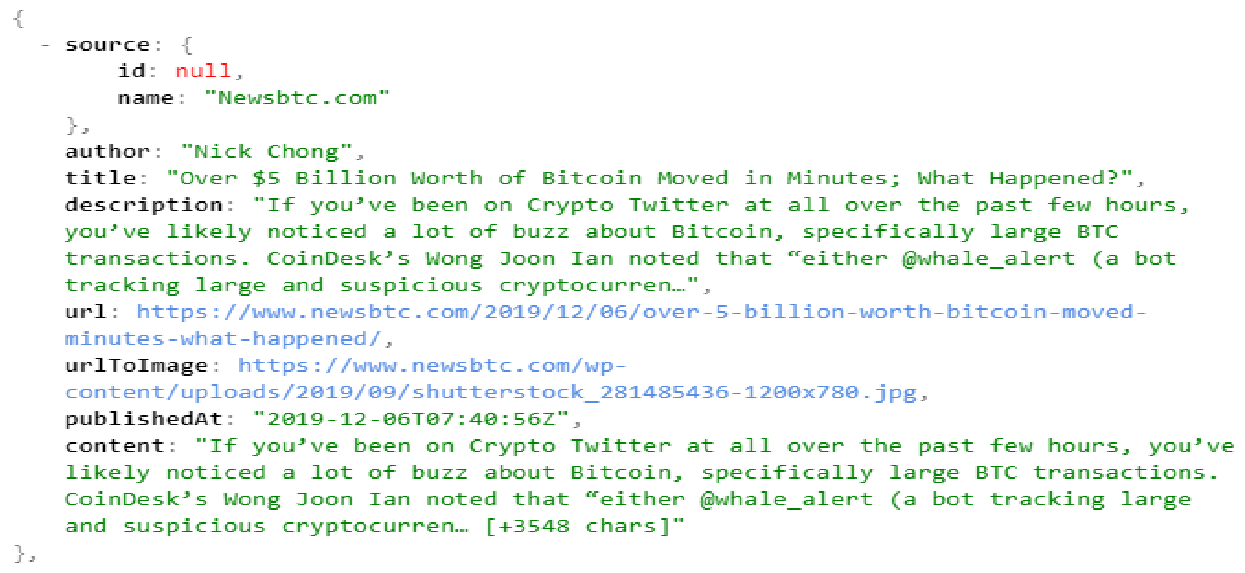
End-User can user multiple services that are currently available to engage his/her needs.



if Weather info "Dhaka" = "Rainy"
do Custom Print news q "newspaper_headlines"
from "https://newsapi.org/v2/top-headlines?"
sortBy "q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5d..."

Figure 5 . User making composition with services

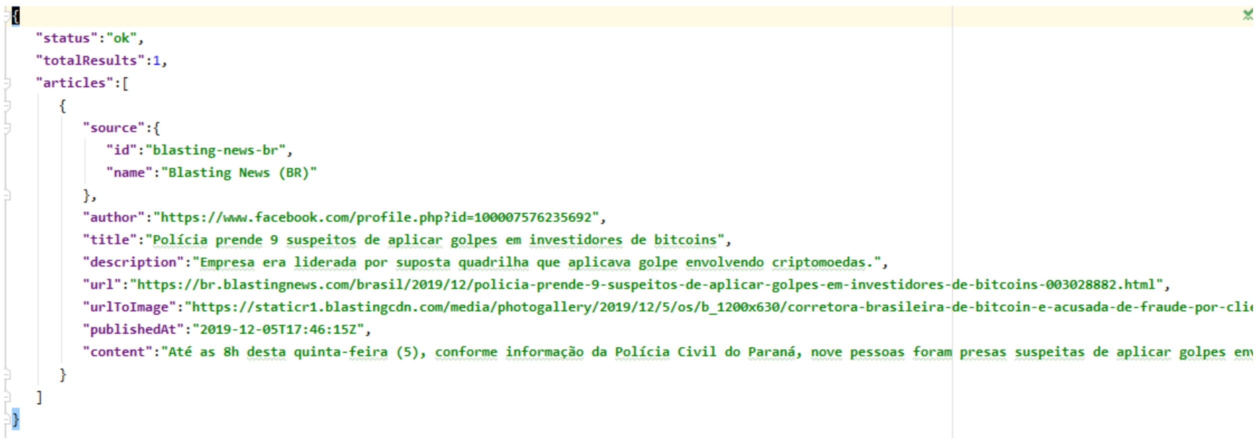
User provides his/her composition based on the needs and system provides the output on the result section.Following are the two different result for two different context.If the weather is sunny then we will get the following result.



```
{
  - source: {
        id: null,
        name: "Newsbtc.com"
    },
    author: "Nick Chong",
    title: "Over $5 Billion Worth of Bitcoin Moved in Minutes; What Happened?",
    description: "If you’ve been on Crypto Twitter at all over the past few hours,
    you’ve likely noticed a lot of buzz about Bitcoin, specifically large BTC
    transactions. CoinDesk’s Wong Joon Ian noted that “either @whale_alert (a bot
    tracking large and suspicious cryptocurren…",
    url: https://www.newsbtc.com/2019/12/06/over-5-billion-worth-bitcoin-moved-
    minutes-what-happened/,
    urlToImage: https://www.newsbtc.com/wp-
    content/uploads/2019/09/shutterstock_281485436-1200x780.jpg,
    publishedAt: "2019-12-06T07:40:56Z",
    content: "If you’ve been on Crypto Twitter at all over the past few hours, you’ve
    likely noticed a lot of buzz about Bitcoin, specifically large BTC transactions.
    CoinDesk’s Wong Joon Ian noted that “either @whale_alert (a bot tracking large
    and suspicious cryptocurren… [+3548 chars]"
},
```

Figure 6 . News Result for Sunny

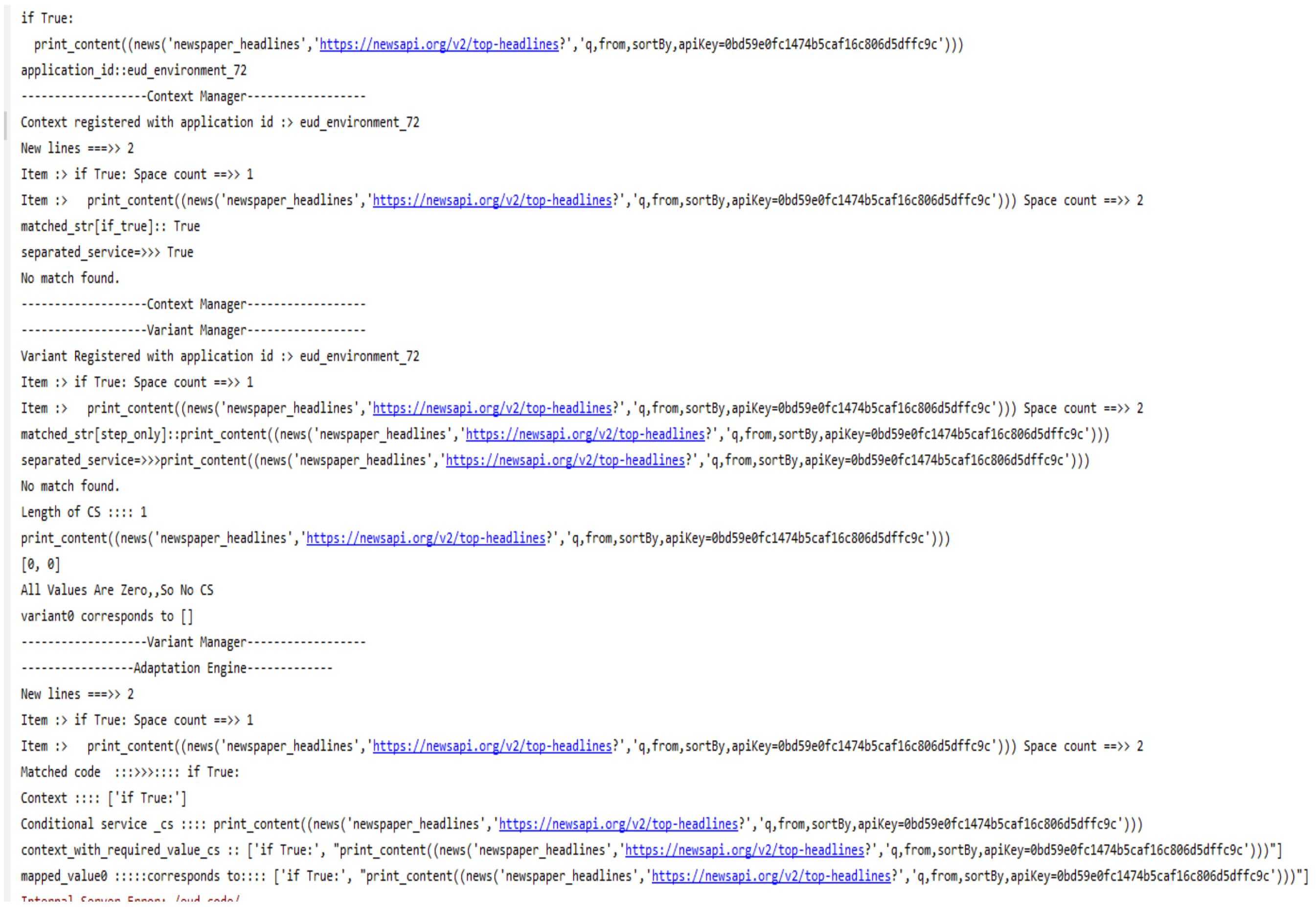
And here is the news result for rainy weather.



```
{
   "status":"ok",
   "totalResults":1,
   "articles":[
      {
         "source":{
            "id":"blasting-news-br",
            "name":"Blasting News (BR)"
         },
         "author":"https://www.facebook.com/profile.php?id=100007576235692",
         "title":"Polícia prende 9 suspeitos de aplicar golpes em investidores de bitcoins",
         "description":"Empresa era liderada por suposta quadrilha que aplicava golpe envolvendo criptomoedas.",
         "url":"https://br.blastingnews.com/brasil/2019/12/policia-prende-9-suspeitos-de-aplicar-golpes-em-investidores-de-bitcoins-003028882.html",
         "urlToImage":"https://staticr1.blastingcdn.com/media/photogallery/2019/12/5/os/b_1200x630/corretora-brasileira-de-bitcoin-e-acusada-de-fraude-por-cli
         "publishedAt":"2019-12-05T17:46:15Z",
         "content":"Até as 8h desta quinta-feira (5), conforme informação da Polícia Civil do Paraná, nove pessoas foram presas suspeitas de aplicar golpes em
      }
   ]
}
```

Figure 7 . News result Rainy

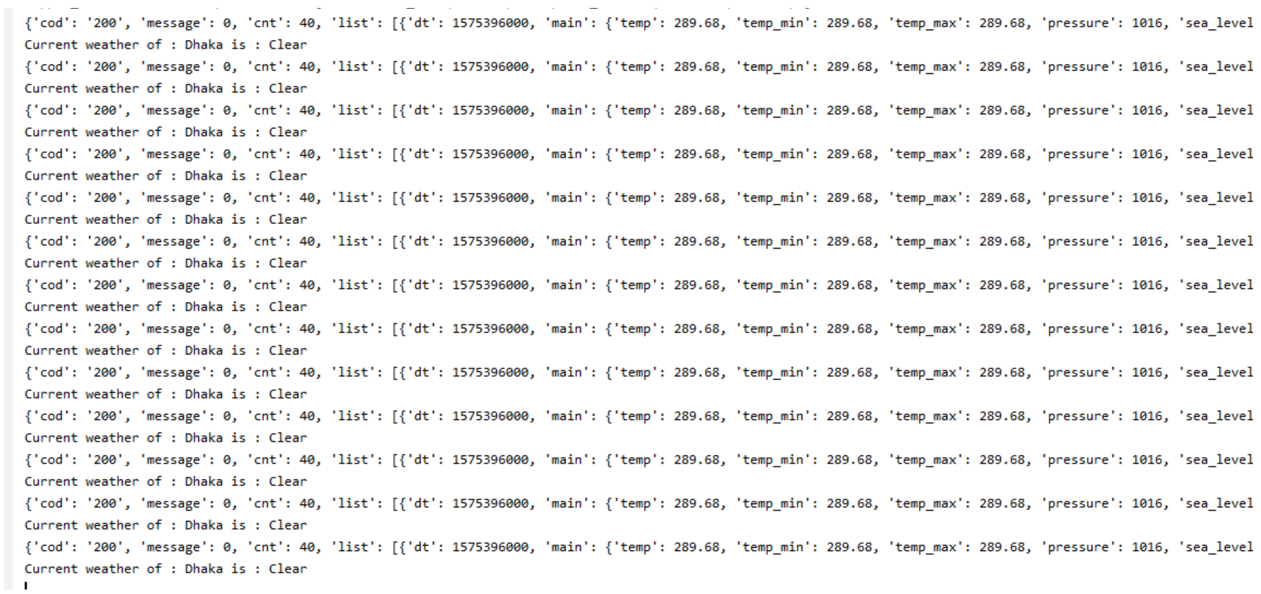
Following figure is the visual representation about what we mentioned and discussed in our architectures each step thing.You can have a visual look on it that provides how everything goes through starting from Application Manager to Runtime Engine.



```
if True:
  print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c')))
application_id::eud_environment_72
------------------Context Manager-----------------
Context registered with application id :> eud_environment_72
New lines ===>> 2
Item :> if True: Space count ==>> 1
Item :>   print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c'))) Space count ==>> 2
matched_str[if_true]:: True
separated_service=>>> True
No match found.
------------------Context Manager-----------------
------------------Variant Manager-----------------
Variant Registered with application id :> eud_environment_72
Item :> if True: Space count ==>> 1
Item :>   print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c'))) Space count ==>> 2
matched_str[step_only]::print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c')))
separated_service=>>>print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c')))
No match found.
Length of CS :::: 1
print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c')))
[0, 0]
All Values Are Zero,,So No CS
variant0 corresponds to []
------------------Variant Manager-----------------
----------------Adaptation Engine------------
New lines ===>> 2
Item :> if True: Space count ==>> 1
Item :>   print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c'))) Space count ==>> 2
Matched code  :::>>>:::: if True:
Context :::: ['if True:']
Conditional service _cs :::: print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c')))
context_with_required_value_cs :: ['if True:', "print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c')))"]
mapped_value0 :::::corresponds to:::: ['if True:', "print_content((news('newspaper_headlines','https://newsapi.org/v2/top-headlines?','q,from,sortBy,apiKey=0bd59e0fc1474b5caf16c806d5dffc9c')))"]
Internal Server Error: /eud_code/
```

Figure 8 . Working flow of each part

Following attached image shows the output of continuous adaptation of a service. We used here weather service as our context and we used a newspaper service which has that condition that in order to read newspaper the weather should be rainy but in here we get the weather status as Clear so newspaper service does not execute.



```
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
{'cod': '200', 'message': 0, 'cnt': 40, 'list': [{'dt': 1575396000, 'main': {'temp': 289.68, 'temp_min': 289.68, 'temp_max': 289.68, 'pressure': 1016, 'sea_level
Current weather of : Dhaka is : Clear
```

Figure 9 . Continuous adaptation of services

CHAPTER V CONCLUSION

In this work we tried to build an environment where user can satisfy his/her needs based on his comfort zone. All the previous work that we have analyzed none of them provided the user an adaptive environment. User has less comfort on those systems because every time the environment changes the user need to again customize his service. But we tried to build an architecture and we showed the evaluation of that architecture on the user scenario problems. By adaptive environment we meant if any changes happen on the environment the system will adopt it and provide the actual result.

And the other thing is there are no system which has the capability to adopt environment changes with IoT sensor and actuators. Our architecture provides that service too. User can plug and play with his/her sensors and actuators and have them adopt on the environment changes.

5.1. Limitation

We have done some work on the purpose that we have started with. Yet there are some limitations in this work. Our system yet cannot get the user input while adoption the environment changes. So a user interaction engine will be needed to fulfill this limitations, though we had less time to establish our work.

5.2. Future Work

In future there will be some good scope to work on. One is to execute multiple application and adopt those application in the environment and another scope will be the ultimate user integration engine.

REFERENCES

Imtiaz-Ud-Din, K.M. (2011). Collaboration based intelligent service composition at runtime by end users.

Kurt Geihs, Michael Wagner (ICCASA 2012) Context-Awareness for Self-Adaptive Applications in Ubiquitous Computing Environments.

Francis M. David, Bill Donkervoet, Jeffrey C. Carlyle, Ellick M. Chan, Roy H. Campbell (OTM 2007). Supporting Adaptive Application Mobility.

Ahmed Elkhodary, Jon Whittle (2007). A Survey of Approaches to Adaptive Application Security

Ortin, F. and O'Shea, D. (2018). Towards an Easily Programmable IoT Framework Based on Micro services. *Journal of Software*, 13(1), pp.90-102.

Tammy R. Fuller, Gerald E. Deane (FTC-2016). IoT applications in an adaptive intelligent system with responsive anomaly detection.

Salman Taherizadeh, Andrew C. Jones, Ian Taylor, Zhiming Zhao, Vlado Stankovskia (2018). Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review.

Yousef Abuseta ((IJCSSE-2018). Towards an MDD Based Framework for Self Adaptive IoT Applications Development.

Young-Joo Kim, Jong-Soo Seok, YungJoon Jung, and Ok-Kyoon Ha (2016). Light-Weight and Versatile Monitor for a Self-Adaptive Software Framework for IoT System.

APPENDIX A

Application Manager which will manage all the incoming and outgoing communication.

```
class ApplicationManager:
    code = ""
    app_id = ""

    def __init__(self):
        print("in init")

    def get_app(self, code):
        code = code
        print(code)
        app_id = generate_id()
        ContextManager.register(ContextManager, app_id, code)
        VariantManager.register(VariantManager, app_id, code)
        AdaptationEngine.set_details(AdaptationEngine, app_id, code)
    def generate_id():
        app_id = app_name + "" + str(random.randint(1, 100))
        print('application_id::'+app_id)
        return app_id
```

APPENDIX B

Regular expressions which will be used to find out conditional step and non-conditional step from a code snippet.

```
regex_if_true = re.compile(r"^if\strue:|^if\sTrue:$", re.MULTILINE)
regex_if_false = re.compile(r"^if\sfalse|^if\sFalse:$", re.MULTILINE)
regex_service = re.compile(r"(if).*?:$", re.MULTILINE)
regex_step_only = re.compile(r".*?", re.MULTILINE)
```

APPENDIX C

Context Manager which has the responsibility to find out the context and later watch those contexts required values and pass those values to Adaptation Engine as per request.

```
class ContextManager:
    code = ""
    app_id = ""
    context_list = []

    def __init__(self):
        print("in init")

    def register(self, app_id, code):
        self.code = code
        self.app_id = app_id
        print("-------------------Context Manager------------------")
        print("Context registered with application id :> " + app_id)
        self.find_context(self)
        print("-------------------Context Manager------------------")
        return

    def unregister(self):
        self.code = ""
        self.app_id = ""
        self.context_list = []
        print("Unregistered >> Context Manager")

    def find_context(self):
        new_lines = len(self.code.split('\n'))
        print("New lines ===>> " + str(new_lines))

        # Split string into segments based on new line
        code_list = self.code.splitlines()
        for item in code_list:
            print("Item :> " + item + " Space count ==>> " + str(item.count(' ')))
        if len(code_list) > 1:
            for i in range(len(code_list)):
                item_code = code_list[i].strip()
                if item_code.startswith('if'):
                    if regex_if_true.match(item_code):
                        match_str = regex_if_true.match(item_code.strip())
                        print('matched_str[if_true]::' + match_str.group()[2:-1])
                        context = match_str.group()[2:-1]
                        self.context_list.append(context)
                        print('separated_service=>>>' + context)
                    elif regex_if_false.match(item_code):
                        match_str = regex_if_false.match(item_code.strip())
                        print('matched_str[if_false]::' + match_str.group()[2:-1])
                        context = match_str.group()[2:-1]
                        self.context_list.append(context)
                        print('separated_service=>>>' + context)

                    elif regex_service.match(item_code):
                        match_str = regex_service.match(item_code.strip())
                        print('matched_str[service]::' + match_str.group()[2:-1])
                        context = match_str.group()[2:-1]
                        self.context_list.append(context)
                        print('separated_service=>>>' + context)
                    elif regex_step_only.match(item_code):
                        match_str = regex_step_only.match(item_code.strip())
                        print('matched_str[step_only]::' + match_str.group())
                        context = match_str.group()[2:-1]
                        self.context_list.append(context)
                        print('separated_service=>>>' + context)
                    else:
                        print("Does not match")
                else:
                    print("No match found.")
        elif len(code_list) == 1:
            print("Only One Statement")
        return

    def watch_context(self):
        while True:
            time.sleep(2)
            for context in self.context_list:
                print("Watching ::::: "+context)
        return
```

APPENDIX D

Variant Manager which calculates and separates conditional service and non-conditional service and then pass it to Adaptation Engine as per its request.

```python
class VariantManager:
    code = ""
    app_id = ""
    conditional_services = []
    services = []
    variants = {}

    def __init__(self):
        print("in init")

    def register(self, app_id, code):
        self.code = code
        self.app_id = app_id
        print("-------------------Variant Manager------------------")
        print("Variant Registered with application id :> "+app_id)
        self.extract_variant(self)
        self.make_variant(self)
        print("-------------------Variant Manager------------------")
        return

    def unregister(self):
        self.app_id = ""
        self.code = ""
        self.services = []
        self.conditional_services = []
        self.variants = {}
        print("Unregistered >> Variant Manager")

    def extract_variant(self):
        # Split string into segments based on new line
        code_list = self.code.splitlines()
        for item in code_list:
            print("Item :> " + item + " Space count ==>> " + str(item.count(' ')))
        if len(code_list) > 1:
            for i in range(len(code_list)):
                if code_list[i].strip().startswith('if'):
                    item_code = code_list[i+1].strip()
                    print('matched_str[step_only]::' + item_code)
                    conditional = item_code
                    self.conditional_services.append(conditional)
                    print('separated_service=>>>' + conditional)
                else:
                    print("No match found.")
        elif len(code_list) == 1:
            print("Only One Statement")
        return
    def make_variant(self):
        length_cs = len(self.conditional_services)
        print("Length of CS :::: "+str(length_cs))

        for index in range(length_cs - 1, -1, -1):
            print(""+self.conditional_services[index])
            list_of_1_0 = list(map(int, str(format(index, '02b'))))
            print(str(list_of_1_0))
            if all(value == 0 for value in list_of_1_0):
                print("All Values Are Zero,,So No CS")
                list_of_cs_and_s = self.services.copy()
                self.variants['variant' + str(index)] = list_of_cs_and_s
            elif all(value == 1 for value in list_of_1_0):
                print("ALl Values are One,, So ALl CS")
                list_of_cs_and_s = self.services.copy()
                for item in self.conditional_services:
                    list_of_cs_and_s.insert(len(list_of_cs_and_s), item)
                self.variants['variant' + str(index)] = list_of_cs_and_s
            else:
                list_of_cs_and_s = self.services.copy()
                for value in list_of_1_0:
                    if value == 1:
                        index_ = list_of_1_0.index(value)
                        cs_ = self.conditional_services[index_]
                        list_of_cs_and_s.insert(len(list_of_cs_and_s), cs_)
                        self.variants['variant' + str(index)] = list_of_cs_and_s
        for key, value in self.variants.items():
            print(key, 'corresponds to', value)
        return

    def get_variant(self):
        return self.variants
```

APPENDIX E

Adaptation Engine plays a good role with the help of context manager and variant manager. It collects variants from variant manager. And it requests context manager to watch over the context continuously. Then it also checks which variant to select based on the context and it finally passes the selected variant to runtime engine to get executed. We have attached the code snippet in the next page.

```
class AdaptationEngine:
    app_id = ""
    code = ""
    mapping_dict = {}

    def __init__(self) -> None:
        super().__init__()

    def set_details(self, app_id, code):
        self.app_id = app_id
        self.code = code
        print("-----------------Adaptation Engine-------------")
        self.mapper(self)
        #get_context()
        #request_variant()
        print("-----------------Adaptation Engine-------------")
        return

    def mapper(self):
        new_lines = len(self.code.split('\n'))
        print("New lines ===>> " + str(new_lines))

        # Split string into segments based on new line
        code_list = self.code.splitlines()
        for item in code_list:
            print("Item :> " + item + " Space count ==>> " + str(item.count(' ')))
        if len(code_list) > 1:
            for i in range(len(code_list)):
                item_code = code_list[i].strip()
                if item_code.startswith('if'):
                    if regex_mapper.match(item_code):
                        print("Matched code  :::>>>:::: "+item_code)
                        context_with_required_value_cs = item_code.split("==")
                        print("Context :::: "+str(context_with_required_value_cs))
                        cs_ = code_list[i+1].strip()
                        print("Conditional service _cs :::: "+cs_)

context_with_required_value_cs.insert(len(context_with_required_value_cs), cs_)
                        print("context_with_required_value_cs ::
"+str(context_with_required_value_cs))
                        self.mapping_dict["mapped_value"+str(i)] =
context_with_required_value_cs
        for k, v in self.mapping_dict.items():
            print(k, ':::::corresponds to::::', v)
        return

def request_variant():
    variant["variant"+str(random.randint(1,100))] = VariantManager().get_variant()
    return


def get_context():
    for index in range(len(ContextManager().watch_context())):
        context_values.insert(index, ContextManager().watch_context()[index])
    return
```

APPENDIX F

Runtime engine catch the variant that passes by Adaptation Engine and executes that variant and finally print the result.

```
class RuntimeEngine:

    variant = ""

    def __init__(self) -> None:
        super.__init__()

    def get_and_set_code(self, variant):
        self.variant = variant
        return

    def execute_code(self):
        while True:
            time.sleep(2)
            code_ast = ast.parse(self.code)

            init_ast = copy.deepcopy(code_ast)
            init_ast.body = code_ast.body[:-1]

            last_ast = copy.deepcopy(code_ast)
            last_ast.body = code_ast.body[-1:]

            exec(compile(init_ast, "<ast>", "exec"), globals())
            if type(last_ast.body[0]) == ast.Expr:
                return eval(compile(convert_expr_expression(last_ast.body[0]),
"<ast>", "eval"), globals())
            else:
                exec(compile(last_ast, "<ast>", "exec"), globals())

     def print_result(self):
        result = self.execute_code(self)
        print(result)


def convert_expr_expression(expr):
    expr.lineno = 0
    expr.col_offset = 0
    result = ast.Expression(expr.value, lineno=0, col_offset=0)

    return result
```