International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7, Issue-6, March 2019

Development of Generic Context-Aware Middleware for Pervasive Smart Environment (GCAMPSE) Based Unified Modeler (GUM TOOL)

J. Madhusudanan, S. Geetha, V. Prasanna Venkatesan, D. Saravanan, Venkata Naresh Mandhala

Abstract*: *In today's world, the needs for smart environments are growing rapidly to make the user comfortable to work and live within it. The need to create smart environment for developing pervasive services like "Anywhere, Anytime, any service" is required to serve the users. These kinds of domain services are the focus of major applications, when merging the required services to create a pervasive application. The developments of pervasive applications require many smart devices which are heterogeneous in nature. In the real world, developing a pervasive application is expensive due to its different heterogeneous smart devices. The major challenge is to integrate the heterogeneous devices and to make it work in the smart environment. The other challenge is to test the working of the pervasive applications. Due to this it is very difficult for the developers to develop a pervasive application. To overcome these challenges, a Generic Context-Aware Middleware for Pervasive Smart Environment (GCAMPSE) based Unified Modeler (GUM) simulation tool is developed. This paper presents the development phase like design, implementation, testing and evolution and maintenance of the GUM tool.

Index Terms*: *Pervasive Computing, Context-aware, Middleware, Tool for Smart environment.

I. INTRODUCTION

"Invisible Computing" and "Many devices work for single users" are the stepping stone for the origin of pervasive computing as stated by Mark Weiser, 1991. Invisible computing with the functionalities of "anytime, anywhere, any service" is partially achieved with the help of many devices working (monitoring) for a single user in a pervasive application. Despite this, invisible computing need is felt in all pervasive application with different functionalities "all situations, all environments, all time" support. Currently the devices focus on single user, despite the need today's computing trend is not only devices but also to focus on location, objects, time, etc [14].

With these requirements, it is clear that pervasive systems are moving towards a new version of Pervasive Computing era as "Many devices work for a single context" where context may be stated as information (context will be user, object, location, time, etc). Therefore context-aware computing is the primary solutions in the near future. Context awareness can be achieved in any pervasive application with the support of heterogeneous context (Ricardo Couto Antunes da rocha and Markus Endler, 2006; Bettini et al., 2010) which are inferred using heterogeneous smart devices (Zia UshShamszaman et al., 2014) [19, 4, 24].

II. EXISTING TOOL BASED DEVELOPMENT EFFORTS

In order to test the working of the application model/ architecture a tool based method is needed for the developer. The development tool acts as a test bed for most of the application developers. In recent times, pervasive application developers are using various pervasive developer tools for their applications. The Context Tool Kit (Anind K Dey et al., 2001) model is made up of different context widgets and it is placed in a distributed environment. These context widgets are used to access the context information and also hide the context sensing procedure. Context widgets are software components that are used for developing pervasive application. This model deals only with the context sensing aspect [3].

The other tool Olympus (Anand Ranganathan, 2005) is to specify the devices, locations and user in a high-level description form which in turn is converted into actual active space components that is used to build programs. Since this model specifies the context, in a high-level description it is difficult for the developer to understand. It is domain specific so there are more difficulties in defining the specification of the application [2].

Archface is a tool (Naoyasu et al., 2007), that acts as an interface between the architecture and methods and to implement the application. It has its own architecture description language to describe the specification of application and it has programming level interface to convert the specifications into the implementation.

The major drawback of the tool is that it deals only with architecture perspective and not with the various components that is used to develop the application [16].

Revised Manuscript Received on 30 March 2019.
***** Correspondence Author
J. Madhusudanan*, Department of Computer Science and Engineering, Sri Manakula Vinayagar Engineering College, Pondicherry, India,
S. Geetha, **V. Prasanna Venkatesan**, Department of Banking Technology, Pondicherry University, Pondicherry, India,
D. Saravanan, **Venkata Naresh Mandhala**, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh, India.

PervML (Carlos Cetina et al., 2007) is a domain specific language tool used to define the requirement specification in meta-model design. This meta-model is in turn mapped to convert it into program that hides the high-level complexities involved in the development of the pervasive application. This model defines the specifications of the tool in descriptive form, which makes the developer difficult for defining the specifications without predefined knowledge about the domain [5].

Diasuite (Damien Cassou et al., 2012) is a tool that allows the designer to define the taxonomy of specific application environment by using the existing application model. It has its own simulation tool to simulate the working of the application with the prewritten specification of the application [7].

From the existing tool, it is found that simulation tool is one of the best options to be used as part of designing a pervasive application to allow developers and designers to discover design alternatives such as building a smart environment in different possibilities, test theories such as verify the application in different contexts and confirm performance prior to developing a real-time application. This leads to the development of new tool for designing and development of pervasive smart applications.

III. DESIGN AND DEVELOPMENT OF GUM TOOL

Based on the design and concept of reference model for pervasive architecture (Matthias Baldauf et al., 2007, Satyanarayanana M, 2001; Rongying Zhao et al., 2011), a tool is designed encompassing the Generic Context-Aware Middleware for Pervasive Smart Environment(GCAMPSE) reference architecture (Madhusudanan et al.,2014) and device layer and service layer of reference model. It is named as GCAMPSE based Unified Modeler (GUM). GCAMPSE based Unified Modeler (GUM) tool is proposed with the intention of developing diverse domain applications [15, 21, 20, 12,13].

GUM tool development process starts with the design level separation of the application into three different design aspects as domain service aspects, context aspects and device aspects. Essential device components and service components are considered to simulate the smart environment using GUM tool. Device aspects are used to design the simulation environment with different components, such as

- Device components are used to design the devices, which are required in the pervasive applications,
- Environment components to define the location of the environment in the pervasive application and
- User components to create different users in the pervasive application.

Context aspects get the scenarios of the application as an input and parse each scenario into different contexts. Each context can be used to simulate the environment of a specific pervasive application to test the working of the pervasive application. Domain service aspect use device and context aspect to develop different domain services according to the pervasive application requirements. Different phases of UM tool development is shown in Figure 1.

The first phase of the tool is the design phase in which the design of the pervasive application is been created using the device and context aspect. In device aspect, each component is designed as pre-defined GUM component which is ready to be implemented. In context aspects, different contexts are validated with the specific environment and the different contexts information are accumulated, which are used in the pervasive applications. In implementation phase, GUM components can easily be interfaced with any programming language.

In testing phase, various scenarios are tested by simulating the different contexts of the pervasive application. In evolution and maintenance phase, developers are allowed to create or recreate any component of the pervasive environment. With GUM tool, various experimental works are carried out by developing different pervasive applications. The results are used to find the advantages and disadvantages that are involved in the specific pervasive application.

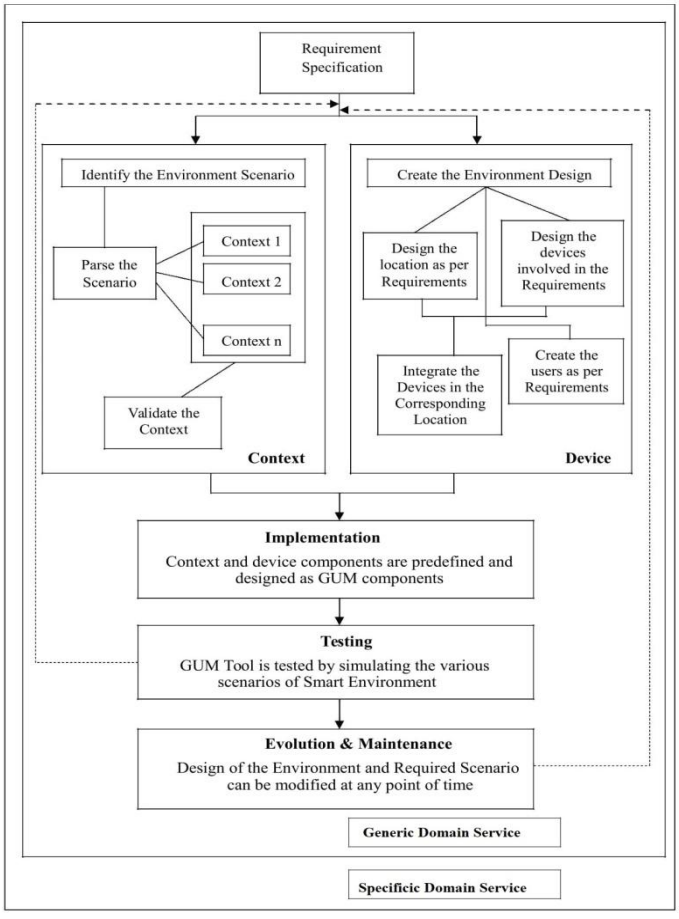

Fig. 1: Development Phase of GUM Tool

IV. DERIVED GUM ARCHITECTURE USING GCAMPSE REFERENCE ARCHITECTURE

The system architecture shown in Figure 2 depicts the three main layers involved in the GUM middleware. The first layer is the Environment creator that provides the Interface for environment creation and device creation dynamically.

This Layer also contains the language support that helps to build the complete environment. The second layer is the Scenario Handler, whose purpose is to generate the required scenario file, and the final layer is the Scenario simulator that reads the scenario file, changes the GUM components accordingly and does the simulation of various scenarios.

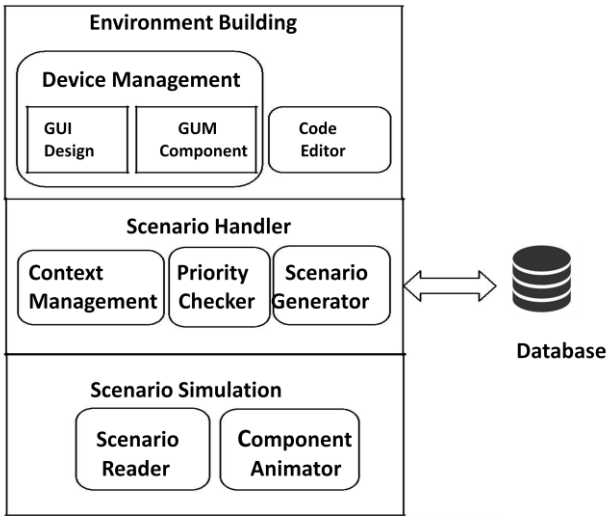

Fig. 2: GCAMPSE Based Unified Modeler Architecture

4.1 Environment Building

Environment is created as required by the user with specified dimensions, deploying devices, saving and loading the required environment whenever needed. The User interface was developed similar to Visual Studio interface, as the device list can be found on left panel, with middle panel holding the current environment, and right panel holding the properties of the selected device.

The environment builder is programmed in such a way that a double click event on the device list triggers the action of adding a new device on the environment panel. Each device created can be moved anywhere around the environment panel. Any number of devices can be added up in the environment. To change the property of a device, the required device has to be selected and the corresponding property gets displayed in the property panel. Any of the property may be changed and it gets updated in the database once the environment is saved. The color of the devices will be in black, if it is not in the active state. If the device is in active mode, then the device would be changed to a new color denoting that its status has been changed to ON.

In Figure 3, GUM tool development environment is shown, where the developer could be able to setup room(s) along with the smart devices, which is present at the Left Panel, within the Environment.

Environment Building Window comprises of following components:

1. Menu Bar
2. Tool Bar
3. Left Panel (GUM components)
4. Centre Panel (Designing work space)
5. Right Panel (GUM component properties)

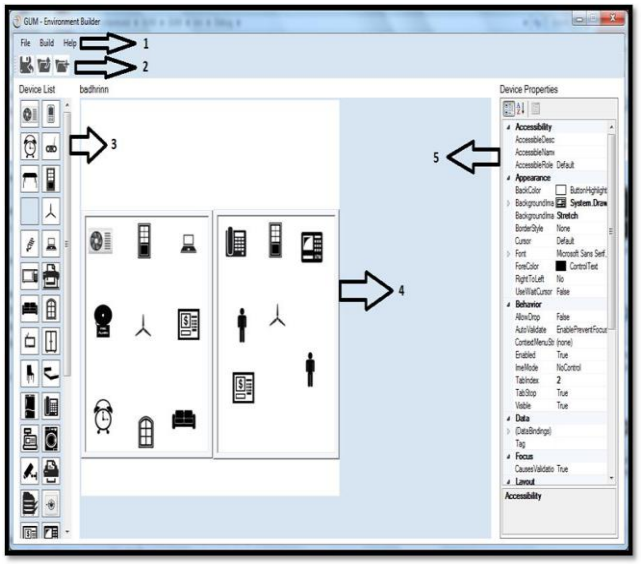

Fig. 3: GUM Tool Development Environment

4.2 Device Management

This module deals with creating the devices required for environments like Home, Hospital, Bank, etc. These devices were created as GUM Components.

4.2.1 GUM Components and GUI Design

GUM component is an Object Linking and Embedding (OLE) custom control, it supports the development of "drag-and-drop" programs that can be developed in any language and used dynamically by any pervasive application. These programs are used in GUM tool as GUM components and the application in which they are run is known as a Graphical User Interface Environment. This GUM component-based approach to pervasive application development reduces development time of the application and improves the program capability and quality. GUM component is a pre-defined component which is ready to execute.

Each GUM component is programmed such that they have their own properties. E.g. TV with properties likes volume, channel, status, location, etc. These device components are built with support for autonomy. Consider a TV moved to Bed room, the location property of TV changes automatically to Bed room from its previous value.

GUM component is classified into two categories as general GUM component and smart environment GUM component. In general, some GUM components are common to any environment creation in diverse domain applications as shown in Table I. Smart environment GUM components are again classified into many specific environment GUM components such as smart home environment GUM component as listed in Table II, smart bank environment GUM component as listed in Table III, smart hospital environment GUM component as listed in Table IV.

Table. I: Components of General Environment

GENERAL	
User	It is used to denote a person in the environment
Room	It is used to denote a location
Fan	It is an electronic device used by the user to access a fan
Tube Light	It is an electronic device which acts as a light source in an environment
Air Conditioner	It is a device which is used to connect to an AC component and made use by the user
Mobile	It is an electronic device used to simulate a mobile phone to enable mobility
Telephone	It is an electronic device used to simulate a mobile phone to enable connectivity
Computer	It is an electronic device which is used to perform every day process at ease
Chair	It can be used as a furniture in any kind of environment which can be accessed by the user
Table	It can be used as a furniture in any kind of environment which can be accessed by the user
Door	It can be used as a portal to travel between two locations
Window	It is used to achieve visual connectivity between two Locations
Fire Alarm	It is a device used to alarm the user in an environment during mishaps due to fire
Printer	It is an electronic device used to convert an electronic document into a Hard copy
Xerox Machine	It is an electronic device which is used to produce multiple copies of an electronic document
CCTV	It is an electronic device which is used to keep track of the activities at a location for security purposes
Sofa	It can be used as a furniture in any kind of environment which can be accessed by the user

Table. II: Components of Smart Home Environment

HOME	
Alarm	It is an electronic device which is used to alert/remind the user about something at a particular time
Oven	It is an electronic device used to cook device
Bed	It can be used as a furniture in any kind of environment which can be accessed by the user
Fridge	It is an electronic device used to preserve eatables at a low temperature
Bureau	It can be used as a furniture used to store materials
Television	It is an electronic device used for entertaining purpose
Washing Machine	It is an electronic machine used to wash clothes
Music Player	It is an electronic device used to play audio files for entertainment purpose

Table. III: Components of Smart Bank Environment

BANK	
Currency Counter	It is an electronic device used to count the number of currency notes
Cash Deposit Machine	It is an electronic device used to deposit money
ATM	It is an electronic device used to retrieve money from a bank account
Fake note Detector	It is an electronic device to detect counterfeit currency
Locker	It is a modern electronic device used to hold valuables and currency at a safe place

Table. IV: Components of Smart Hospital Environment

HOSPITAL	
Hematology Analyzer	It is an instrument used to analyze a person's blood chemistry
Stethoscope	It is an instrument to analyze the frequency of the heartbeats of a person
Stretcher	It is used to transport patients to various locations
Thermometer	It is an instrument used to determine the body temperature of a person
Prescription Pad	It is a notepad used to record the medical history of a Patient
Wheel Chair	It is used to transport patients to various locations
Injection	It is a mechanical device to inject medicine into a person's blood circulation
Sphygmomanometer	It is an apparatus used to determine the blood pressure of a patient
Oxygen Concentrator	It is a device used to control the oxygen concentration administered into a patient
Sugar Tester	It is a device used to find the blood sugar concentration in a person's blood
Weight Machine	It is an electronic device used to measure a person's weight

4.3 Code Editor

GUM has its own syntax for the editor which can be used to create devices to a specified location in the current environment and to set properties for each device created. A user may add and move the devices manually, but if he needs a device to be moved to an exact location by specifying the accurate location, or to set up an accurate property to it, then he needs to use the code editor. Thus, purpose of the GUM code editor is to create the environment with accurate positions.

Syntax for adding Device to current Environment:

Add(Device,Device_Name,x=x_loc,y=y_loc)

Syntax for setting up property for a particular device:

Set(Device_Name,Property_Name,Property_Value)

Property Name E.g: Width, Height, Status, Length, Breadth, etc.

4.4 Scenario Handler

This part is used to get the user defined scenarios and to generate the scenario file. This scenario builder displays the current environment name, and devices present in it along with the users associated with them. The scenario given by the user must follow the following syntax:

Format

IF user \action/ in (location) at <time> turn ON/OFF [device1] & turn ON/OFF [device2]

Here in \action/ context the user should specify the priority value for the context. The scenario gets built according to the priority of the devices and the actions. The generated scenarios can be viewed and modified if it is needed. The scenario file gets saved in a specified location.

Table. V: Different Aspects of Smart Home, Smart Hospital and Smart Bank

	Smart Home	Smart Hospital	Smart Bank
Number of Contexts Supported	5 (Location, User, Device, Activity, Time)	5 (Location, User, Device, Activity, Time)	5 (Location, User, Device, Activity, Time)
Number of Devices	4	32	18
Number of Users	2	4	3
Number of Activities	2	2	1
Number of Locations	3	5	4
Number of possible scenarios	96	2560	212
Context Management	Dynamic (At Run Time)	Dynamic (At Run Time)	Dynamic (At Run Time)
Device Creation	Static (At Design Time)	Static (At Design Time)	Static (At Design Time)
Device Management	Dynamic (At Run Time)	Dynamic (At Run Time)	Dynamic (At Run Time)
Non-Functional Qualities Achieved	Scalability, Modifiability, Integrability, Reusability and Ease of creation.	Scalability, Modifiability, Integrability, Reusability and Ease of creation.	Scalability, Modifiability, Integrability, Reusability and Ease of creation.

4.4.1 Process Steps for the Scenario Handler

Step1: Start the GUM Scenario Handler

Step2: Enter the input Scenario

- Enter the input scenario for both applications of smart home and smart banking.
- The scenario for smart home application will be like this, if \kalai/ relaxing in living room at evening turn light on
- The scenario for smart banking application will be like this, if \kalai/ depositing cash in deposit counter at morning turn computer on and turn printer on.
- These are the sample scenarios given as input to the generic context middleware. Similarly, all the possible scenarios are given as input to the generic context-aware middleware.

Step3: Check scenario exists in database

- If scenario exists in database, display scenario already exists in database and goto Step5.
- Otherwise parse the input scenarios.

Step4: Split the context from the input scenario and generate scenario file

- From the input scenario split the context e.g. if \kalai/ relaxing in living room at evening turn light ON.
- This scenario is divided into different contexts like user context as kalai, activity context as relaxing, location context as living room, time context as evening, device context as light and then generate the scenario file for representing the given scenario.
- The scenario file which is represented using XML will be like this,

Xml Representation:

```
<? xml version="1.0" encoding="utf-8" standalone="yes"?><Scenario>
<Environment>Smart Home</Environment><Scenario1>
<User>Kalai</User>
<Action>relaxing</Action><Location>living room</Location><Time>evening</Time>
<Device1>light</Device1>
<Status1>ON</Status1>
</Scenario1>
</Scenario>
```

Step5: Check the contexts (user, location and device) derived from the input scenario matches with the environment

- If context matches with the environment, Goto Step6.
- Otherwise Goto Step2.

Step6: Get the priority value for each context (user, location, device).

- Set the priority for users, locations and devices in the environment.
- Generate the contexts of the given scenario in the form of scenario file.

Step7: Save the scenario file for future reuse

- Save the scenario file which represents the given scenario, to the scenario repository.

Step 8: Open/delete the scenario files in the database

- It is provided with options of open/delete any number of scenario files in the database.

Step 9: Stop the GUM scenario handler

4.5 Simulation of Scenarios

The simulation for a scenario can be done by loading the scenario file with required context. The users can be moved

around and tested for the working of given contexts. If a new scenario should be loaded and tested, the current scenario should be unloaded. The scenarios can be loaded and unloaded any number of times. During the simulation, only user can be moved around the environment, and all devices become fixed. At initial stage, all devices will be at inactive state. If the devices become active, then it is indicated with the help of colour change. The user's activity property will be at null state initially, denoting that the user is not doing any action currently. The required action must be entered in the user's activity property before moving him, to check how the environment reacts for his action, such as learning, sleeping, entering, etc. The work flow for the simulation of scenarios is briefed below.

4.5.1 Process Steps for Simulation of Scenarios

Step 1: Start the GUM tool

Step 2: Create the Environment

• Various environments like smart home, smart hospital can be build by specifying the length and breadth.

Step 3: The required components are dragged and dropped for the Simulation of the Scenario in the Environment.

• Here components represent the contexts like users, locations and devices. Since components are created with supported format and added to the environment. It can be easily added to the environment.

Step 4: Save the Environment

• After building the environment with the devices at the specified location, it can be saved for future reuse.

Step 5: Build the Scenario

• The required scenario for the environment can be build by following the steps given in the Scenario Handler process.

Step 6: Run Scenario

• After building the Scenario, it has to be executed to simulate the scenario for the smart environment.

Step 7: Get the input for scenario simulation as a scenario file.

• Get the input from the scenario handler as scenario file and generate the scenario and perform the simulation.

• For example scenario file which is represented using XML, *<? xml version="1.0" encoding="utf-8" standalone="yes"?><Scenario>*

<Environment>Smart Home</Environment><Scenario1>

<User>Kalai</User>

<Action>relaxing</Action>

<Location>living room</Location><Time>evening</Time><Device1>light </Device1><Status1>ON</Status1></Scenario1>

</Scenario>

• The example scenario generated is "if \kalai/ relaxing in living room at evening turn light ON".

Step 8: Check the given Scenario with rule scenario file

• If both matches, Goto Step9.

• Otherwise no simulation of the scenario and Goto Step10.

Step 9: Simulate the scenario

• GUM tool simulates the specified scenario in the built environment.

Step 10: Exit the GUM tool.

V. CASE STUDIES

More and more demand arise in the development of pervasive application as it facilitates the end user or day to day activities of the human world through supporting the smart services which are invisible and available everywhere. As there are many domains coming up, the need for pervasive application development is more and also growing. Further there is a need to support quick development of pervasive application to cope up with the rapid growth in the computing field. In order to meet the needs of the pervasive application development, a number of methodologies are adapted, particularly to support heterogeneity of smart devices and different context of the smart environment. Context-aware middleware plays a big role to support heterogeneity of smart devices and different context of the environment.

There are about 110 middleware (K.E.Kjaer, 2007; Soma Bandyapadhyay et al., 2011; Paolo Bellavista et al., 2013, Vasanthi R et al., 2011) that were developed in the last ten years and they are implemented in different domains. Most of these implementations are smart environment applications like smart office, smart class, smart car, smart meeting room, smart home, smart hospital, etc. In this paper, three different domain applications are chosen to develop using GUM tool. Smart home offers smart services to facilitate the stakeholder. More demand is visible particularly to help elder people who stay alone in the home[11, 22, 17, 23].

- Smart hospital assists the workers, patients and doctors to make a comfortable working and living environment. 24 X 7 monitoring and reporting of the patient status is one of the main features of the smart hospital.
- Smart banks are intended to provide a customized service to the customer through smart gadgets through which more customers are attracted and improve the business of the bank.
- Initially the need and requirement of the three different domain applications are analyzed. There are many requirements for the smart home as (Kidd et al., 1999; Reinisch et al., 2010) [18, 10].
- Location of the user should be monitored continuously, to support the location-based service.
- Access preference of the devices should be defined, to control the intruders or children's abnormal activities.
- Activities of the users are continuously monitored, to support activity-based service.
- Predefined activities are required, to handle the abnormal situation of the environment.
- Continuous monitoring for health status of elder users are required to support medical emergency.

Many middleware is already available to support different features. There is middleware to support context features alone and there are middleware to support specific device features. CDTOM middleware (Hongbo Ni et al., 2011) is implemented to support elderly people in the home [9].

ATLAS middleware (AbdelsalamHelal et al., 2007) and HYDRA middleware (Soma Bandyopadhyay et al., 2011) are developed to interface the heterogeneous smart device in the smart home[1,22]. Smart hospital is the evolving application of pervasive computing and it requires many features to support the functionality of the hospital. The requirements are

- Smart device particularly for medical support are allocated for different functionalities of the hospital. Example ECG monitor reading are updated to the doctor regularly.
- Different locations in the hospital are classified based on the high priority. For example, ICU is given more priority than doctor room.
- Device access preferences are set to restrict the intruder usage. For example, only doctor can use operation theater equipments.
- Location access preferences are set to restrict the trespasser usage. For example, ICU can be accessed by authorized user only.
- Different smart services are developed to support the functionalities of the hospital. For example, tracking of nearby doctor to attend the critical patient.

Altas and Hydra (AbdelsalamHelal et al., 2007; Soma Bandyopadhyay et al., 2011) are the two-middleware developed to support the heterogeneous devices. Marks middleware are designed to support the location-based service in the hospital [1,22]. Pico middleware uses the hospital database and provides smart service to the doctor. In the perspective of smart hospital very minimal number of middleware are available.

Smart Bank is the demanding application of pervasive computing and it requires many services to support the working of the bank. The requirements are

- Assisting the customer to get a proper service is the primary objective of the bank.
- Smart gadgets are needed to support smart banking operation and reducing the time of customer and bank employee.
- Access preference to the location and devices are needed. For example, non-customer cannot enter locker room.
- Currency tracking provision will help to avoid money laundering problem.

There is no middleware to support smart bank application and the need for the development is mentioned in the literature survey (Chong et al., 2011).

Abstract context component of the generic context-aware middleware is realized into concrete context component through the development of three different domains of smart environment applications such as smart home, smart hospital and smart bank using GCAMPSE reference architecture. These smart environments handle different context namely user contexts, location contexts, device contexts, activity contexts and time contexts.

Each context type will have a specific context value which is dynamic as per the smart environment. Context information is continuously monitored through GUM components properties where the changes will be updated according to the status of the smart environment. Different contexts are framed as scenarios and execution of these scenarios in turn serves as a service to the smart environment. Smart environment applications are verified using different scenarios which are tested in the GUM tool simulation environment. Scenarios can be created dynamically. This enables to accomplish the requirements of the smart environments. Priority of scenarios is defined through the priorities of the contexts and it helps to manage the conflict between the scenarios. Table V represents the number of contexts considered in the design of smart home, smart hospital and smart bank and these smart applications use 5 different context such as user contexts, device contexts, location contexts, time contexts and activity context. Smart home use 4 devices, whereas smart hospital uses 32 devices and smart bank uses 18 devices in their environments. In terms of scenario generation, smart home can generate 96 scenarios, smart hospital can generate 2560 scenarios and smart bank can generate 212 scenarios based on three case studies.

Context management handles dynamic creation of context in all three applications. But in device management, the creations of new devices are static, and the creation of existing devices and their management are dynamic. Smart home, smart hospital and smart bank support non-functional qualities such as scalability, modifiability, integrability reusability and ease of creation.

Also, the different domain applications can be designed independently using the GUM tool. Now it is evident that three or more smart applications can be developed through GUM tool. Case studies are taken to exhibit the use of proposed architecture and related experience. To Propose the required metrics and study of the various functional and non-functional characteristics these case studies are considered as a future work.

VI. COMPARISON OF GUM TOOL WITH EXISTING TOOLS

Simulation tool is the best approach to design and simulate a smart environment application and test the various possible scenarios by executing in the simulated platform. Existing simulation tool are taken from (Damien Cassou et al., 2012). They are as follows [7].

Context toolkit: It is used to monitor the context of the environment with the help of context widget and each context widget is defined as pre-defined classes which support abstraction in implementation level. It supports development of smart environment application in terms of programming perspective, but not in the simulation perspective. It gives provision to add devices in design level and maintenance of heterogeneous devices in implementation level.

Olympus: It defines the smart environment application requirement as predefined specification and in turn the specification is converted to a program. User, location and environment are fixed as per the smart environment application and support the dynamic heterogeneous devices.

Arch face tool: It has its own architecture description language to describe the specification of architecture. The specification is used as static information about the smart environment application through which abstraction is supported.

PervML tool: It collects the requirement of the smart environment application and stores it in a meta-model. In turn, this meta-model is mapped to convert into program and also hiding the high-level complexities involved in the development of the pervasive application. It also supports heterogeneous devices to infer context information.

Diasuite tool: It allows the designer to define the taxonomy of specific application environment by the existing application model. It has its own simulation tool to simulate the working of the application with the prewritten specification of the application through which abstraction is supported. It has a provision to add device and new environment in device level and supports heterogeneous devices and its modification. Existing smart environment design can be reused to develop new smart environment application. In smart environment application, different scenarios are tested using its own simulation tool and predefined scenarios to handle abnormal situation. The Diasuite tool provides for modification of its device and its environment.

There are many similarities in Context toolkit, Olympus, PerML and Archface tools. They are working based on predefined specification and in turn it is converted as program, but it does not support in perspective of simulation. So, the abstraction property is achieved, despite the features like addition of users, locations and environments are not possible to attain in design level. Testing and evolution of the smart environment application are restricted as it works on predefined specification. In four different phases, GUM tool can be compared with the existing tools, they are design, implementation, testing and evolution and maintenance of the project. The GUM Tool is evaluated based on the different parameters of each phase.

6.1 Comparison Based on Design Phase

In order to evaluate GUM tool at design level, four context handler parameters are chosen as per (Eduardo Castillejo et al., 2014; Damien Cassou et al., 2012), they are user, location, environment and device [7,8]. These parameters are chosen to ensure the following features

- User parameter is used to check whether new users can be added to the environment.
- Location parameter is chosen to ensure new locations can be added in an environment.
- Environment parameter is selected to verify the environment is changeable.
- Device parameter is chosen to make sure whether new devices can be added to the environment.

The comparison of GUM tool with the existing tools is shown in Table VI. In this Diasuite supports the parameters device and environment whereas all the other tools support only device parameter in the design phase. The Archface tool does not support any of the above parameter in the design phase. In GUM tool, smart environment is designed using GUM components which provide an option to create and monitor any number of users, devices, locations and environments. From this it is evident that all parameters are achieved.

Table. VI: Comparison of GUM Tool with Existing Tools in Design Phase

Middleware	Device	Environment	Location	User
GUM	✓	✓	✓	✓
Diasuite	✓	✓	-	-
Olympus	✓	-	-	-
Context TK	✓	-	-	-
Archface	-	-	-	-
PervML	✓	-	-	-

6.2 Comparison Based on Implementation Phase

In order to evaluate GUM tool at implementation level, three parameters are chosen as per (Damien Cassou et al., 2012), they are reusability, heterogeneity and abstraction. These parameters are chosen to ensure the following features

- Reusability is to ensure that the implementations of the existing smart environments, which are developed by the same tool, are reused to create different smart environment application.
- Heterogeneity is to check whether the diverse domain devices are properly managed. It also interfaces with the smart environment.
- Abstraction is to make sure that the complexities of developing the smart environment applications are hidden from the view of developer.

In a development cycle, the implementation stage is defined as the execution of the design phase. The implementation stage of the GUM tool deals with the execution of context aspect and device aspect of the smart environment application to exhibit its qualities.

- **Reusability:** In GUM tool, implementation is based on different components such as user component, device component and location component, where components are reused effectively.
- **Heterogeneity:** Device heterogeneity is supported through interface of any smart device with the GUM component.
- **Abstraction:** Complexity of implementation is hidden through GUM development approach where developers are provided to use drag and drop of the GUM component to design smart environment application.

From the above discussion, it is obvious that GUM tool supports all the three properties and the comparison with other existing tool is shown in Table VII.

Table. VII: Comparison of GUM Tool with Existing Tools in Implementation Phase

Middleware	Abstraction	Heterogeneity	Reusability
GUM	✓	✓	✓
Diasuite	✓	✓	✓
Olympus	✓	✓	-
Context TK	✓	✓	-
Archface	✓	-	-
PervML	✓	✓	-

The table shows that only GUM and Diasuite supports all the properties of implementation phase. All the other tools support only Abstraction and Heterogeneity whereas the Archface tool supports only Abstraction property in the implementation phase.

6.3 Comparison Based on Testing Phase

Testing of pervasive application is very complex, because creating real environment involves high cost (devices, sensor, location and various contexts). In order to evaluate GUM tool at testing level, three parameters are chosen as per (Damien Cassou et al., 2012). They are concurrency, conflict handling and proactiveness. These parameters are chosen to ensure the characteristic of scenarios

- Concurrency is used to test whether the scenarios are executed completely in the simulated environment.
- Conflict handling between scenarios is used to ensure that the smart environment applications are managing the conflict situation.
- Proactiveness is to verify the provision provided to the smart environment application to manage itself.

Smart home, smart bank and smart hospital are developed using GUM tool and the smart application is tested with different scenarios and providing solution to handling scenario conflict through priority of the context.

Scenarios for the abnormal situation are predefined and it will get executed on that context.

These parameters ensure the developers to test the smart environment application with different scenarios and its attained properties are compared with the other existing tools as shown in Table VIII. Here only Diasuite supports proactiveness and concurrency in the testing phase. All the other existing tools does not support any of the properties when compared with the GUM tool.

Table. VIII: Comparison of GUM Tool with Existing Tools in Testing Phase

Middleware	Proactiveness	Concurrency	Conflict Handling
GUM	✓	✓	✓
Diasuite	✓	✓	-
Olympus	-	-	-
Context TK	-	-	-
Archface	-	-	-
PervML	-	-	-

6.4 Comparison Based on Maintenance and Evolution Phase

In smart environment development, maintenance and evolution stage plays a vital role, because modification is much needed to cope with the rapid growing smart environment applications. In order to evaluate GUM tool at maintenance and evolution level, three parameters are chosen as per (Damien Cassou et al., 2012). They are context modification, environment modification and device modification. These parameters are chosen to ensure whether

- Context modification checks the contexts of the smart environment can modify according to the application.
- Device modification ensures the smart environment can cope with dynamic devices.
- Environment modification verifies whether there is a provision to alter the smart environment.

In GUM tool, GUM components can be created or recreated at any time as per the application requirement. GUM tool supports the dynamic devices and its context of the smart environment through monitoring the properties of GUM components. Hence the modification is achieved in context and device, GUM tool compared with other existing tools as shown in Table IX. In the Evolution and Maintenance phase only Diasuite supports device and environment modification. Olympus and Context TK tool supports device modification where as Archface and PervML does not support any of the modification properties.

Table. IX: Comparison of GUM Tool with Existing Tools in Evolution and Maintenance Phase

Middleware	Device Modification	Environment Modification	Context Modification
GUM	✓	✓	✓
Diasuite	✓	✓	-
Olympus	✓	-	-
Context TK	✓	-	-
Archface	-	-	-
PervML	-	-	-

From the comparison of GUM tool with the existing tools it is found that GUM tool has various merits. The GUM tool provides a platform for the developer to create a smart environment application and to test various scenarios in the smart environment application. It provides provision for addition of any domain specific components to the GUM tool which in turn help the developer to develop the domain specific smart environment. It supports characteristics of scenarios like concurrency of scenarios, conflict handling of scenarios and proactive scenarios. It also supports reusability, which helps the developer to reuse the components as well as the smart environment applications.

VII. CONCLUSION

In this paper, design and the implementation of GUM tool is presented. The GUM tool architecture is composed of two main components. First one is GUM user interface which is used in the design and development of smart environment

application in diverse domains. Second one is the scenario simulation, where the different scenarios are tested with different cases. Finally, the GUM tool is compared with the existing tools and its merits are reported.

REFERENCES

1. (AbdelsalamHelal et al., 2007) Abdelsalam (Sumi) Helal, Hen-I Yang, Jeffrey King and Raja Bose, "Atlas - Architecture for Sensor Network Based Intelligent Environments", ACM, 1073-0516/01/03 00-0034, 2007.
2. (Anand ranganathan et al., 2005) A. Ranganathan, S. Chetan, J. Al-Muhtadi, R.H. Campbell and M.D. Mickunas, "Olympus: A High-Level Programming Model for Pervasive Computing Environments", Proceedings of IEEE Third International Conference on Pervasive Computing and Communications, pp. 7-16, ISBN: 0-7695-2299-8, 2005.
3. (Anind K Dey et al., 2001) A.K. Dey, G.D. Abowd and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Journal of Human-Computer Interaction, ACM, Vol. 16, no. 2, pp. 97-166, 2001.
4. (Bettini et al., 2010) C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D.Riboni, "A Survey of Context Modelling and Reasoning Techniques", Journal of Pervasive and Mobile Computing, Elsevier, Vol.6, no.2, pp. 161–180, 2010.
5. (Carlos cetina et al., 2007) Javier Muñoz, Vicente Pelechano and Carlos Cetina. "Software Engineering for Pervasive Systems: Applying Models, Frameworks and Transformations", IEEE International Conference on Pervasive Services (ICPS), pp. 290-294, E-ISBN: 1-4244-1326-5, Jul 2007.
6. (Chong et al., 2011) G. Chong, L. Zhihaoand, Y. Yifeng, "The Research and Implement of Smart Home System Based on Internet of Things", International Conference on Electronics, Communications and Control (ICECC), IEEE, pp. 2944 –2947, ISBN: 978-1-4577-0320-1, 2011.
7. (Damien Cassou et al., 2012) Damien Cassou, Julien Bruneau, Charles Consel and Emilie Balland, "Toward a Tool-Based Development Methodology for Pervasive Computing Applications", IEEE Transactions on Software Engineering, Vol. 38, no.6, pp.1445-1463, 2012.
8. (Eduardo Castillejo et al., 2014) Eduardo Castillejo, Aitor Almeida, Diego Lopez-de-Ipina and Liming Chen, "Modeling Users, Context and Devices for Ambient Assisted Living Environments" Sensors, Vol.14, no.3, pp. 5354-5391, 2014.
9. (Hongbo Ni et al., 2011) Hongbo Ni, BessamAbdulrazak, Daqing Zhang and Shu Wu, "CDTOM: A Context-Driven Task-Oriented Middleware for Pervasive Homecare Environment", International Journal of UbiComp (IJU), Vol.2, No.1, Jan 2011.
10. (Kidd et al.,1999) Kidd, C D., Orr, R J. Abowd, G D., Atkeson, C G, Essa, I A. MacIntyre, B., Mynatt, E, Starner T E and Newstetter W, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research", CoBuild'99, Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture, LNCS, Vol.1670, Springer-Verlag, pp. 191-198, ISBN:978-3-540-66596-0, Oct 1999.
11. (K.E.Kjaer, 2007) Kristian EllebækKjær, "A Survey of Context-Aware Middleware", Proceedings of the 25th conference on IASTED International Multi- Conference: Software Engineering, pp. 148-155, 2007.
12. (Madhusudanan et al., 2014) J. Madhusudanan, S. Hariharan, A. Manian Selvan and Dr. V. Prasanna Venkatesan, "A Generic Middleware Model for Smart Home" International Journal of Computer Network and Information Security, Vol. 8, pp. 19-25, 2014
13. (Madhusudanan et al., 2014) J. Madhusudanan, P. Anand, S. Hariharan, Dr.V. Prasanna Venkatesan "Verification of Generic Ubiquitous Middleware for Smart Home Using Coloured Petri Nets" International Journal of Information Technology and Computer Science(IJITCS), Vol. 10, pp.63-67, 2014
14. (Mark Weiser, 1991) Weiser M, "The Computer for the 21st Century", Scientific American, Sep 1991.
15. (Matthias Baldauf et al., 2007) M. Baldauf, S. Dustdar and F. Rosenberg, "A survey on context-aware systems", International Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, no. 4, pp. 263-277, 2007.
16. (Naoyasu et al., 2007) N. Ubayashi, J. Nomura and T. Tamai, "Archface: A Contract Place Where Architectural Design and Code Meet Together", Proceedings of 32nd ACM/IEEE International Conference on Software Engineering, Vol.1, pp. 75-84, ISBN: 978-1-60558-719-6, 2010.
17. (Paolo Bellavista et al., 2013) Paolo Bellavista, Antonio Corradi, Mario Fanelli and Luca Foschini, "A Survey of Context Data Distribution for Mobile Ubiquitous Systems", ACM Computing Surveys, Vol.44, no.4, Aug 2013.
18. (Reinisch, et al., 2010) Christian Reinisch, Mario J. Kofler and Wolfgang Kastner, "ThinkHome: A Smart Home as Digital Ecosystem", 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST), pp. 256-261, ISSN: 2150-4938, 2010.
19. (Ricardo CoutoAntunes da rocha and Markus Endler, 2006) Ricardo CoutoAntunes da Rocha and Markus Endler, "Context Management in Heterogeneous, Evolving Ubiquitous Environments", IEEE Distributed Systems Online, Vol. 7, no. 4, ISSN: 1541-4922, 2006.
20. (Rongying Zhao et al., 2011) Rongying Zhao and Ju Wang, "Visualizing the Research on Pervasive and Ubiquitous Computing", Scientometrics, Vol.86, no.3,pp. 593-612, ISSN:1588-2861, 2011.
21. (Satyanarayanana M, 2001) Satyanarayanana M, "Pervasive Computing: Vision and Challenges", Personal Communications, IEEE, Vol. 8, no.4, pp.10 – 17. ISSN: 1070-9916, Aug 2001.
22. (Soma Bandyapadhyay et al., 2011) Soma Bandyopadhyay, MunmunSengupta, SouvikMaiti and SubhajitDutta, "A Survey of Middleware for Internet of Things", Recent trends in wireless and mobile networks communications in computer and information science, WiMo/CoNeCo Vol.162, pp. 288–296, Springer, 2011.
23. (Vasanthi R et al., 2011) Vasanthi R and Dr. R.S.D. Wahidabanu, "A Middleware Platform ForPervasive Environment", International Journal of Computer Science and Information Security, Vol. 9, no. 4, Apr 2011.
24. (Zia UshShamszaman et al., 2014) Zia UshShamszaman, SafinaShowkatAraIlyoung Chong and YounKwaeJeong, "Web-of-Objects (WoO)-Based Context Aware Emergency Fire Management Systems for the Internet of Things", Sensors14, pp. 2944-2966, 2014