Software Design Specification

Version 1.0.0

Project Title : Self Adaptive Event Generation in Mobiles

College : Amrita School of Engineering, Ettimadai
Coimbatore (T.N)

Group Members :

- Ravi Diwakar
- Shreyas Dhoke
- Sumit Kamra

1. Introduction

1.1 Purpose of this document

This SDS describes functions of our project, Self Adaptive Event Generation. Design and various modules of this project are defined in this document which will cover the overall functionality of this project.

1.2 Scope of the development project

The Self Adaptive Event Generation is a mobile software which makes mobile more intelligent compare to other similar devices. This project shows how mobile takes user's decisions based on user conversation. Mobile itself decides which actions to take and when, no need to tell it manually. User has its full control; this system can be activated or deactivated anytime whenever user wants. This software makes user not to remember all his/her meetings or other important things which he/she mentioned to other user on his/her mobile. It will take care of user's conversation and notify to user whenever there is any meeting or any other important thing like any call or message etc.

This is a new idea and not yet developed by any organization so it's shown in our project that how it works in real time. The architecture of the proposed system is given below.

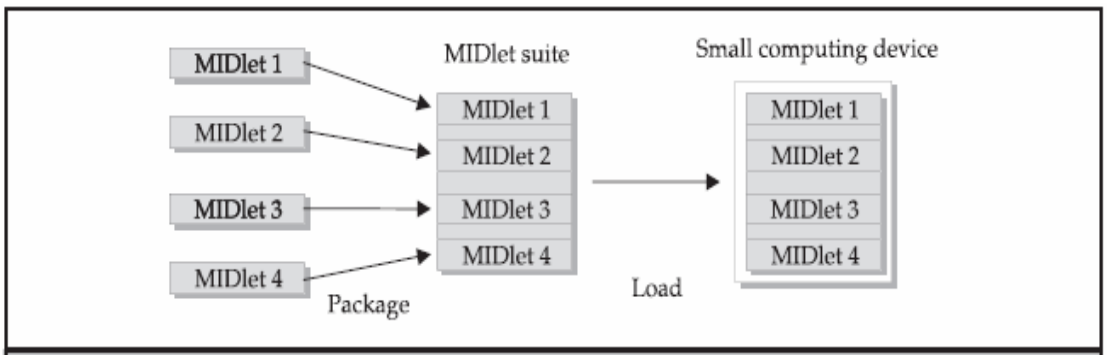


Figure *MIDlets are packaged into MIDlet suites, which are loaded in a small computing device.*

As shown in the figure project contains different MIDlets. Each MIDlet has its own functionality which is different from others. After completing all the

MIDlets they are packaged to make a single unit. Then this unit is loaded into mobile. When this MIDlet unit is launched manually system asked for which MIDlet to launch but when it is connected to mobile operating system it decides itself which MIDlet is to launch ad when. We hope that the users act in a responsible manner and talk in simple English language.

1.3 Definitions, acronyms and abbreviations

EVENT
An event in context of mobile application is the response to the user from the system by invoking certain applications. It can be of the form like a reminder, create message box, contact list with a selected contact number etc.

REMINDER
A reminder is an event triggered only at a particular time. The time may be set by the user or by the system. Reminders can be preloaded or user defined as per the requirement.

ADAPTIVE SYSTEM
A smart system which can adapt to the requirements of the user based on the analysis over a particular time of its usage. The system may behave differently for different users.

EVENT PRIORITY
The priority of an event to be considered while selecting an event when more than one event is recognized and to be notified to the user by the system. The priority can be static or dynamic.

TEXT PATTERN
A sequence of text which is defined for some special meaning and is used to make out a decision by another entity in the system.

3G
It stands for third generation wireless standard. It is a multiplex transmission method. The 3G multiplexing technique uses either CDMA or TDMA to increase the throughput

USER BEHAVIOR

It is the behavior of the user to the system. It is decided based on the analysis that what all applications the user prefers most and least, the time of usage and way of communication i.e. incoming or outgoing.

MIDLET

Application development for mobile devices is done by using an application programming interface (API) known as the Mobile Information Device Profile (MIDP). Applications written for this API are affectionately referred to as MIDlets.

CLDC

The CLDC is designed for 16-bit or 32-bit small computing devices with limited amounts of memory. CLDC devices usually have between 160KB and 512KB of available memory and are battery powered. CLDC devices include pagers, personal digital assistants, cell phones, dedicated terminals, and handheld consumer devices with between 128KB and 512KB of memory.

KVM

KVM is a highly optimized Java Virtual Machine for resource constrained device. The K Virtual Machine (KVM) is very small in size (about 40 – 80 K) and is suitable for devices such as pagers, cell phones and PDAs. The KVM can run on any system that has a 16 bit/ 32 bit processor and a total memory between 160 – 512K of memory.

Abbreviations:

AMS	:	Application Management Software
API	:	Application Programming Interface
CDC	:	Connected Device Configuration
CFD	:	Control Flow Diagram
CLDC	:	Connected Limited Device Configuration
DFD	:	Data Flow Diagram
IDE	:	Integrated Development Environment
JAD	:	Java Application Descriptor
JAR	:	Java Archive
JCP	:	Java Communication Process
JSR	:	Java Specification Request
JVM	:	Java Virtual Machine

J2EE	:	Java 2 Enterprise Edition
J2ME	:	Java 2 Micro Edition
J2SE	:	Java 2 Standard Edition
KVM	:	K Java Virtual Machine
MIDP	:	Mobile Information Device Profile
PDA	:	Personal Digital Assistant
OTA	:	Over-the-air Provisioning
SAEG	:	Self Adaptive Event Generation

1.4 References

J2ME in NUTSHELL by O`Reilly

The Complete Reference J2ME

The Complete Reference JAVA

Wireless Tech Tips, J2ME Tech Tips: February 26, 2002.
http://java.sun.com/developer/J2METechTips/2002/tt0226.html.

Sun Microsystems, Inc, Java(TM) 2 Platform Micro Edition (J2ME(TM)) Technology for Creating Mobile Devices, White Paper, 19.05.2000
http://java.sun.com/products/cldc/wp/KVMwp.pdf

Sun Microsystems, Inc, Mobile Information Device Profile, JCP Specification, 1.9.2000
http://java.sun.com/aboutJava/communityprocess/final/jsr037/MIDPSpec_1_0.zip

www.netbeans.org

http://www-106.ibm.com/developerworks/library/wi-tip21.html

http://www.symbian.com/developer/development/javadev.html

http://java.sun.com/j2se/1.4.2/docs/guide/misc/threadPrimitiveDeprecation.html

http://jal.sun.com/webapps/device/device

1.5 Overview of document

- The first section describes the introduction of document.
- The second section describes the system architecture; in this section we mention the modules of this project and their structures.
- The third section consists of detailed description of the modules described in the previous section modules.
- The fourth section states how our project can be reused and its relationship to other products.
- The fifth section gives the complete idea about design decision and tradeoffs which we had to do in order to simplify the design.
- The sixth section consists of pseudo codes for the components.

2. System Architecture Description

This section gives the exact view of the system as per its requirements.

2.1 Overview of modules / components

The overall SAEG system development can be viewed as following modules:

- Conversation storage and processing
  Process the conversation to identify for any type of event.
- Event Generation
  Generate events to user to get user's response.

- User behavior analysis
  Events will be generated only after analyzing user behavior means which events to be occur.

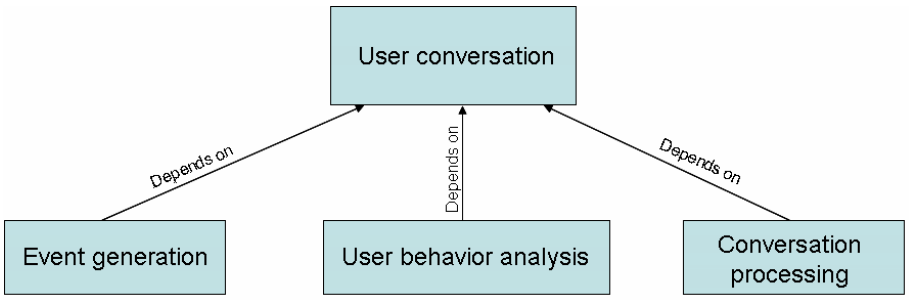
- User response Execution
  Execute the system according to user's response and take the next step.

- System Configuration Utility
  Different options to control the system.

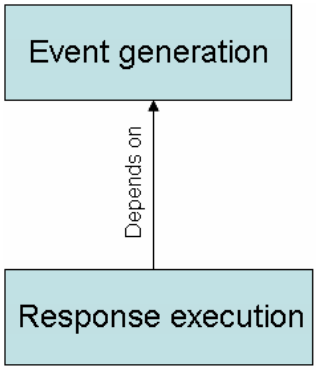
2.2 Structure and relationship

The dependencies of various modules are as follows.

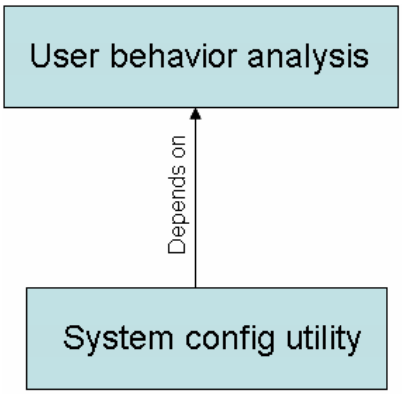
1. Event generation, User behavior analysis and conversation processing depend on the user conversation.



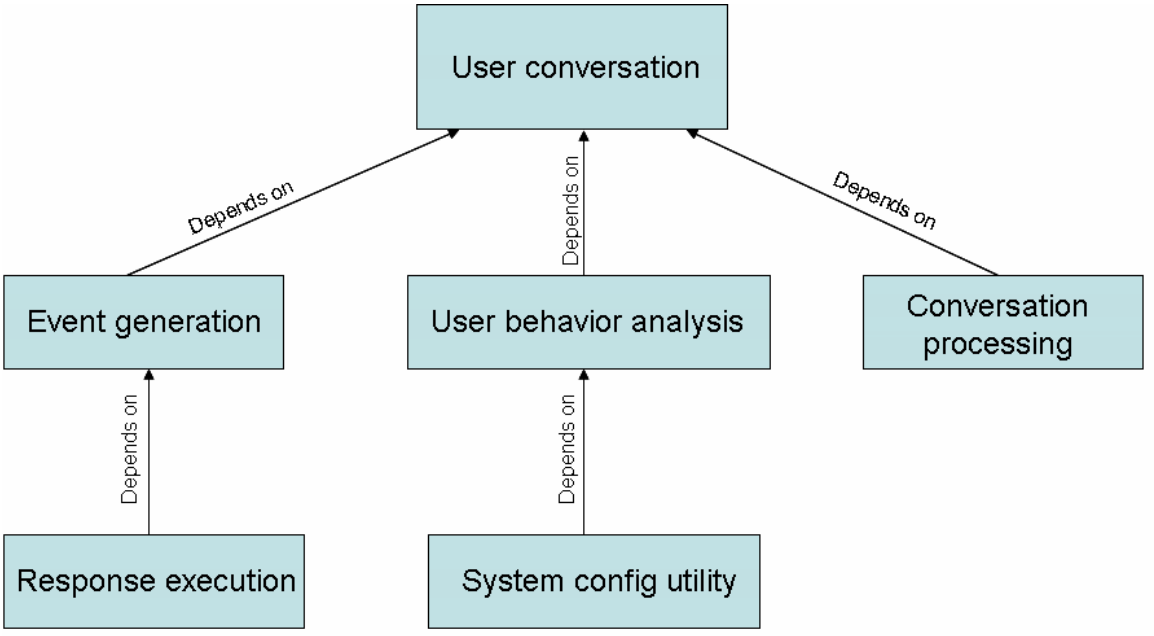
2. Response execution depends on type of event generated.



3. System configuration utility depends on user behavior. According to user behavior, system's options will be activated or deactivated.



So, the overall dependency diagram is as below



2.3 User interface issues

The GUI can be divided into two parts. One is the event generated for the user after processing the user conversation and other is the options given for the event like whether event should activated or deactivated or should be saved.

3. Detailed description of components

3.1 Component template description

Identification	:	The unique name for the component and the location of the component in the system.
Type	:	A module, a subprogram, a data file, a control or procedure, a class, etc
Purpose	:	Function and performance requirements implemented by the design component, including derived requirements.
Function	:	What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.
Subordinates	:	The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.
Dependencies	:	How the component's function and performance relate to other components. How this component is used by other components. The other components those use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.
Interfaces	:	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats,

interactive messages and other user interface components (originally defined in the SRS) should be given here.

Resources : A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are internet connection, databases, memory, CPU requirements if any, I/O channels, printers, cdwriters, libraries, and system services etc.

Processing : The full description of the functions presented in the Function subsection. Pseudo code can be used to document algorithms, equations, and logic.

Data : For the data internal to the component, describe the representation method, initial values, use, semantics, and format.

3.2 Conversation Storage and Processing Description

- Identification : Conversation Storage and Processing

- Type : Module

- Purpose :

The main purpose of this module is to store the user conversation and check for events in the conversation. If any event is found then it is given to other modules and then it is decided whether it should be generated or not.

- Function :

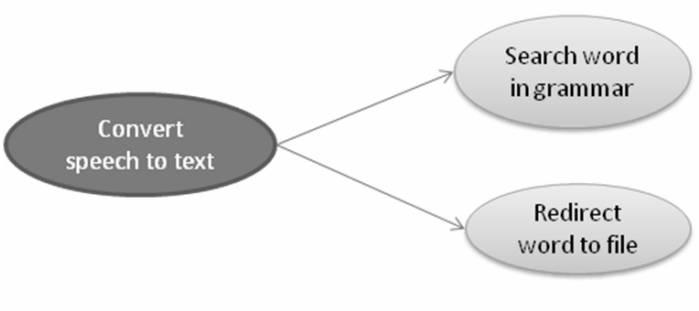
The main function of this module is to scan the user conversation and match with already defined patterns. If it found any event then it is notified after its full processing by other modules

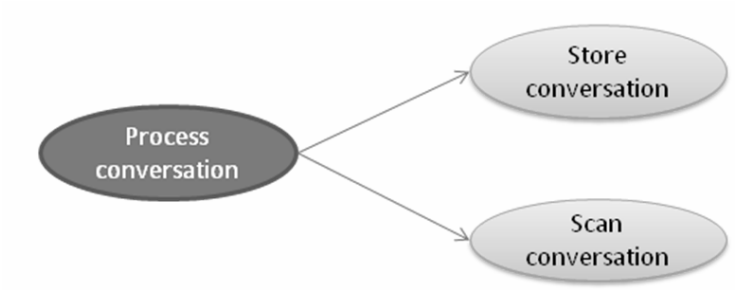
- Subordinates :

This module consists of two components. First component stores users conversation in a text file and the other component scan the full conversation and match it with already defined patterns. First module requires a speech to text converter tool with a better quality. And the other module depends on first sub-module. Events can be generated only if conversation is stored in a right manner with a specific format.

- Dependencies :

This component doesn't depend on other components because it's the first step of the system. Its performance depends only on the hardware tool's quality. Events generated by this component are checked by other components for their time, type and their probability of occurrence. If any event is not to be generated at the same time then it is saved and is generated at its particular time. If the event not to be generated then it is discarded by the system itself after analyzing user behavior.





- Interfaces :

This component doesn't have any interface because it has only background processing of user's conversation.

- Resources :

A speech to text converter tool is required for converting user's speech into a text file. Already defined patterns decide how the conversation will be scanned and events will be generated. It uses KVM and its libraries for internal processing of the functions. It uses mobile's internal database system which is RecordStore. Mobile's operating system does all the processing.

- Processing :

Processing can be divided into various steps.

1. Scan the conversation and break the full string into words.
2. Match with pre-defined patterns.
3. Check if any event occurs in the conversation.
4. Store events in specific format.

- Data :

Only pre-defined patterns are required in this component to recognize the events in the conversation. No other data is required for this component.

3.3 Event Generation Description

- Identification : Event Generation

- Type : Module

- Purpose :

Main purpose of this module is to display various events on mobile's screen generated by the previous module. It also shows different options related with that event whether to accept or deny or save the event. User has full control on this module to activate or deactivate it.

- Function :

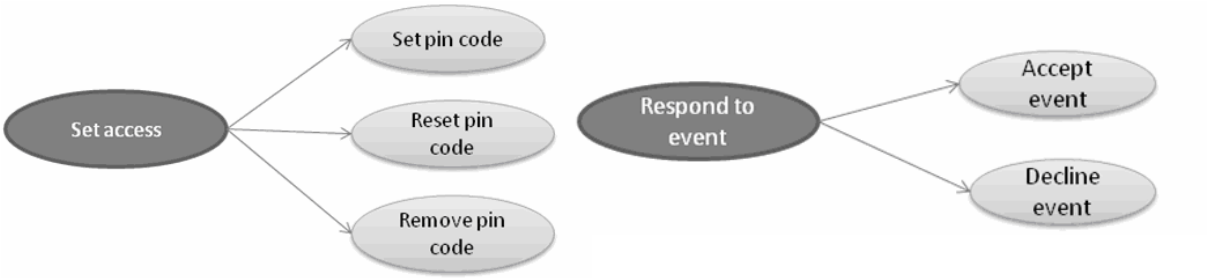
Main function of this module is to notify user with the event generated. Actually it's a GUI in a specific format with different options according to event. When all the processes are completed on any event then those events are shown to users by this module. This module can't be change by user itself. Its appearance depends on the mobile.

- Subordinate :

This module consists of two sub module. First is the event generated which are processed by other modules and another part is the options which are given with that event. Each type of event has different type of options which differ in their execution.

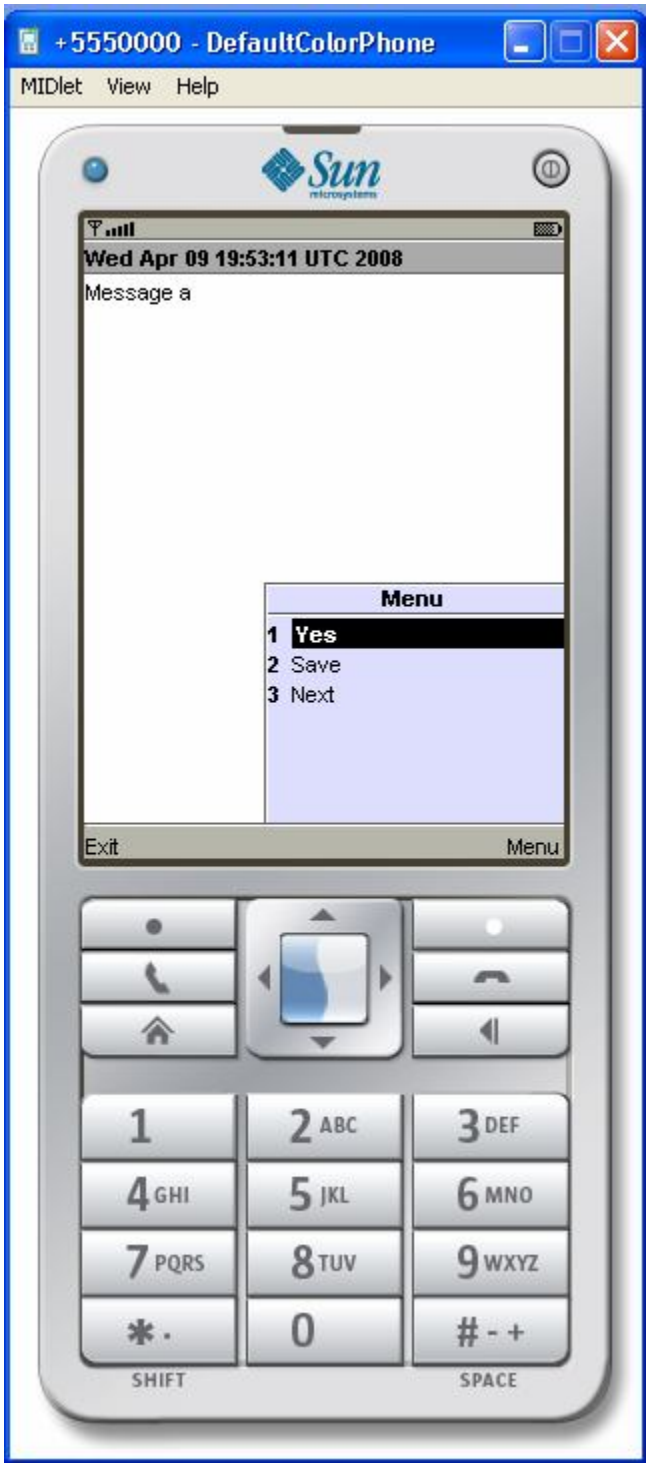
- Dependencies :

This module is totally dependent on background processing. Its main work is to display the event and signal to processor that which step is taken by the user on that event. If no events are given to this module this won't display any notification. All other processes use this module to get user response and to analyze user behavior. This module takes user input and sends it to other modules.



- Interfaces :

This is the main part of this module. Here events are shown to user and user response is send to other modules. Error message is shown when any wrong step is taken by the user of if data is not present for current event. This is the communication medium between user and mobile. According to user



response, messages are passed to other modules. Normal java functions are used to communicate from one module to other module. This diagram shown how an event is generated and what are the options related to it. An event can be accepted or can be denied or can be saved for future references.

- Resources :

This module is developed on J2ME platform so only JAVA enabled mobile phone are the hardware for this system. This module needs events to be shown to the user. This system is flexible with any type of mobile but that should support JAVA platform. Input is taken by mobile's keys which are processed by mobile's processor. It doesn't need any other resources for its functionality.

- Processing :

One function is used to display the events and other function selects options for that type of events. Then whichever option user selects its information is passed to other function for further processing. It takes very less time for this whole processing. Mobile's processor is used for all the processing which is quite enough for this because it doesn't take any heavy work to process.

- Data :

There are some options which are already defined and there working is also fixed. These options are taken by the system according to type of the event. There is no other data required for this module.

3.4 User Behavior Analysis Description

- Identification : User Behavior Analysis

- Type : Module

- Purpose :

Main purpose of this event is to keep information of the user response and use it for other events. How many times an event is accepted or rejected and which event, this all information is stored and according to this user

behavior is analyzed. If any event is rejected many times then that event will not be generated next time until user start its generation manually.

- Function :

The system will keeps on generating different events based on the user conversation. This event generation cycle is analyzed to get statistical information about the types of event and number of times they are generated. The acceptance of the user is the most important part here. The acceptance of the event consists of either performing that event at the time it is generated or saving the important in the To Do list for a later time. This information is used to decide over the priority of event based on user behavior. The method used to calculate the priority is known as "Acceptance Sampling" which is used in factories to decide whether the customer will accept the product or not. In a similar way, here the acceptance sampling is used to predict whether the user will accept the event or not. If the predicted probability is above 40% the event is generated otherwise it is not. Once the probabilities fall below 40%, the events will be generated only when the user resets the priority for the particular person.

- Subordinate :

This is a single module which doesn't consist of any other sub-module.

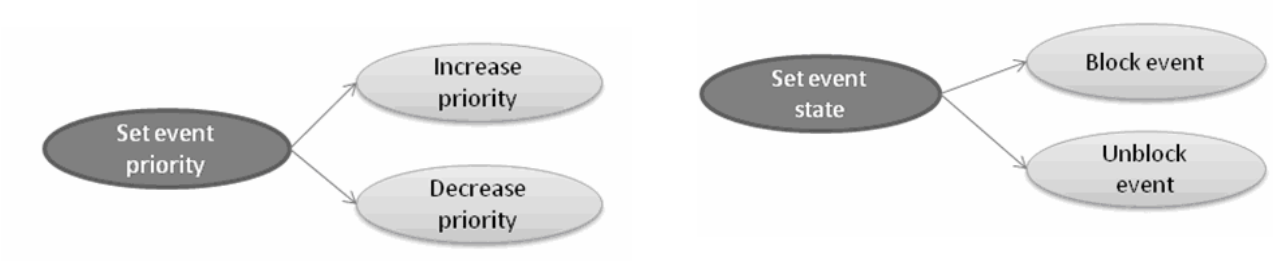
- Dependencies :

This module totally depends on the user response. If any event is rejected many time then that event is not be taken next time until user start its execution manually. Use has full control over this module. It can be activated or deactivated for any type of event and for any person. Acceptance probability is calculated whenever any event is generated and shown to user. Calculation is done based on given formula:

Acceptance sampling :

$$P(A) = p(X <= c) = \sum_{x=0}^{C} \binom{M}{x} \binom{N-M}{n-x} \binom{N}{n}$$

P (A) : probability to generate an event
C : number of times event accepted
M : number of times event rejected
N : total number of all events generated for particular person
N : total number of particular event generated for particular person



- Interfaces :

This module doesn't have any interface.

- Resources :

This module doesn't use any special resources. It only uses JAVA libraries functions which are already defined in mobile function library. It uses KVM as its java libraries functions. No special hardware resources are required for this module. It stores all the information I mobile database which is called as RecordStore.

- Processing :

When user response is passed to this module, this module compares it with previously stored information and adds this also. After this all the calculation part takes place which decides whether an event should be generated or not next time. User can reset this information for any particular type of event and for any particular person also. Before any event is generated, first this information is checked to generate or to delete the event.

- Data :

All values are initialized by a fixed value that is 60% acceptance of any event. When it reaches the value less then 40% then that event is discarded. Whenever user resets the value initial values are taken. User can manually stops the events also. All the calculation is done only after user response and saved for future.

3.5 User Response Execution Description

- Identification : User Response Execution

- Type : Control Procedure

- Purpose :

Main purpose of this part is to execute the user response. When user press any key, mobile process gets the signal and it recognize which key is pressed. Then control is passed to other function related to that key. According to user response necessary steps are taken and command is executed. If there is any event related to call and it is accepted by the user then a call is made to that particular number. If event is denied the control goes to next event. Same event can be saved for future reference.

- Function :

Each event is notified to the user over the mobile screen. Now the user has to respond to the events. User will be given the choice in the form of accept/deny, ok/cancel options. Different events have different choices to perform. The user can also save some events in the To Do List for further notifications. In case the target person of the event is not known, the user is given the option to go to the Address Book and perform the event (call/message) from there.

- Subordinates :

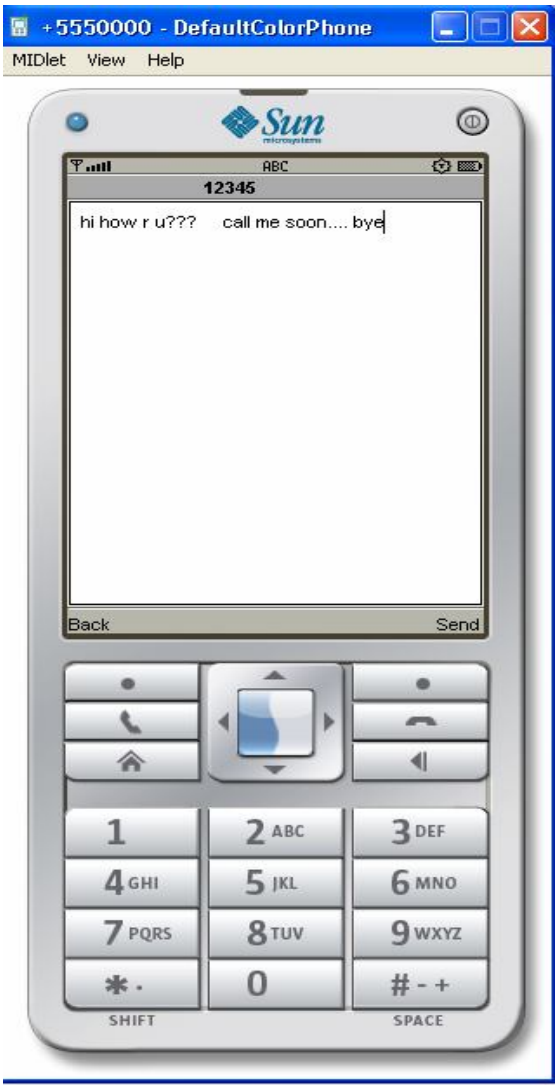
This is a single module which doesn't consist of any other sub-module. According to user response it selects which function to execute and how.

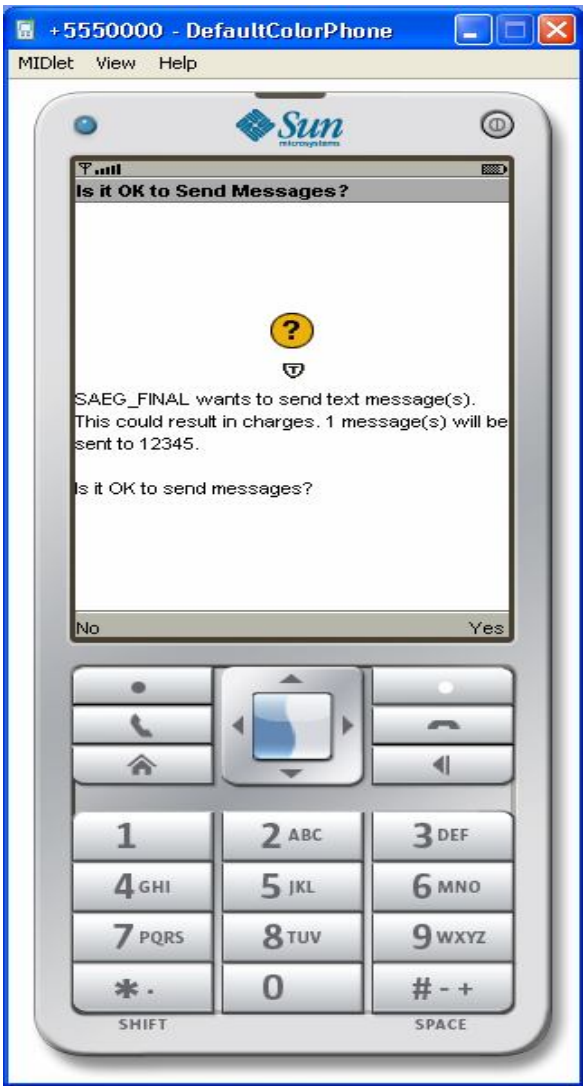
- Dependencies :

This module depends on type of event and response of the user. Because there are different options for each type of event and each option has different result on its execution. So if user denies any event system goes to next event and if user accepts any event it takes necessary steps for its acceptance.

- Interfaces :

Especially this module doesn't have any interface all the interfaces are related to event generation only. But warning/error messages are shown if something goes wrong or if event is accepted then calling screen is shown to user. If event is related to message then control goes so system's write message screen.





These diagrams show how user response is executed. In the first diagram message is written and when send option is selected then message is sent to specific number. Second screen is optionally not necessary to show to users. When the background processing is in progress till then second screen is shown, after this control goes to next event. If there are no more events then control goes to main screen.

- Resources :

This module doesn't use any special resources. It only uses JAVA libraries functions which are already defined in mobile function library. It uses KVM

as its java libraries functions. This module is developed on J2ME platform so only JAVA enabled mobile phone are the hardware for this system. This module needs events to be shown to the user. This system is flexible with any type of mobile but that should support JAVA platform. Input is taken by mobile's keys which are processed by mobile's processor. It doesn't need any other resources for its functionality.

- Processing :

Whenever an event is generated, user selects one of the given options for that event. When he/she presses a key for a particular option, mobile recognizes what user wants to do, and it executes that specific function. Then it gives control to a function which is responsible for a particular action to be happened. Then user gets its result in the form of a GUI. Whole processing is done on J2ME platform. It doesn't need a heavy processing power so mobile processor it quite enough for it.

- Data :

There is no need for any initial data or values to be set. Whatever happen, fully depends on user response.

3.6 System Configuration Utility description

- Identification : System Configuration Utility

- Type : Module

- Purpose :

Main purpose of this module is to provide various facilities to user related to this system like password protection, and other privileges to activate or deactivate the activities. User has his/her full control on the functionality of

the system. When the system is started first time all the activities are activated automatically.

- Function :

This module deals with system configuration changes made by the user. The system user is given some privileges for some basic controls over the system functionalities. The system configuration utility module is consisted of sub modules which are described in the next point.

- Subordinates :

1. Set access

The user can set some access controls to limit the access to the system. The set access sub module includes following tasks:

Task1: Set pin code
By default the system will not be password protected for configuration changes but if the user wants, he can set a pin code of four digit number.

Task2: Reset pin code
The user can reset the pin, changing the old pin when needed.

Task3: Remove pin
Pin can be removed also. In this case the pin will not be asked whenever the configuration changes are made.

2. Set system state

The user is provided with an option to turn on/off the SAEG system feature in his mobile. The set system state sub module includes following tasks:

Task1: Activate system
The deactivated system can be activated whenever user wants.

Task2: Deactivate system
The system can be turned off to avoid all automatically generated events.

3. Set event state

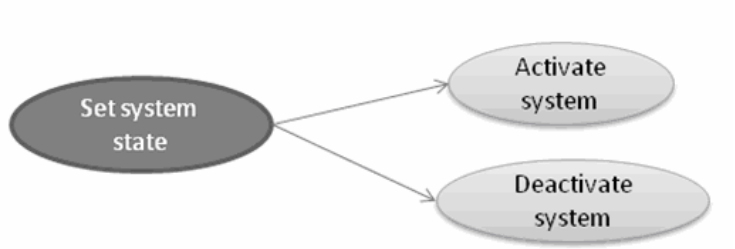
Not all events are relevant to all type of users. This module handles such a situation and facilitates user to have control over the occurrence of events. All events are associated with a current state based on which they are notified to the user. User can set these states. The main tasks include:

Task1: Block event
A particular event can be blocked by removing it from the active event list and adding it to blocked event list. The event state is set to block.

Task2: Unblock event
If the user wants to unblock a blocked event he can simply set the state to unblock. The event is removed from the blocked event list and added to the active event list.

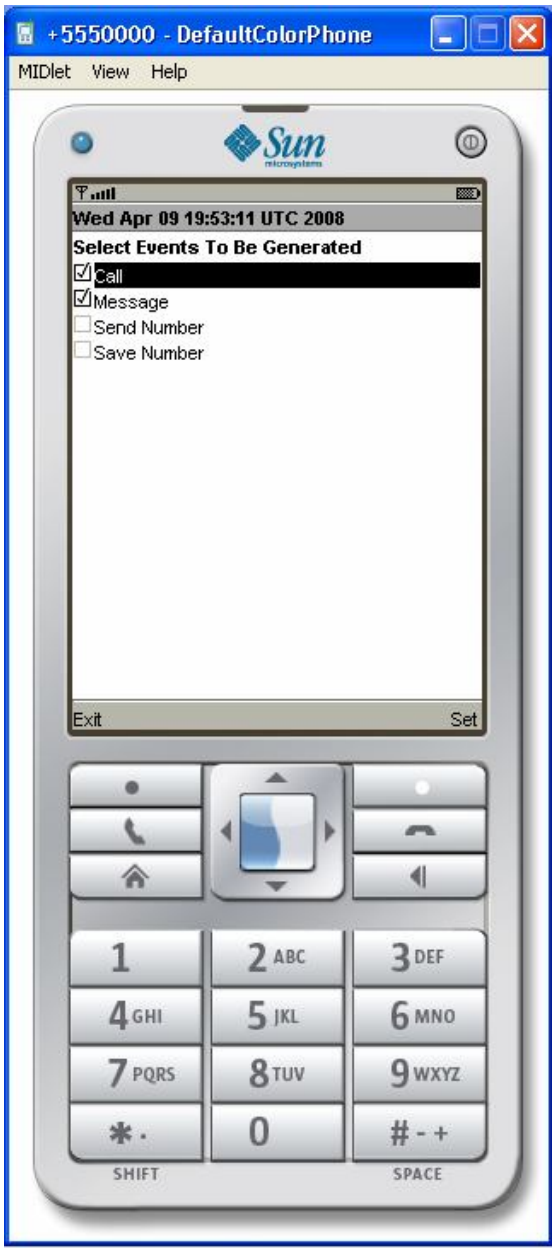
- Dependencies :

This whole module depends on user itself. This all activities can be activated or deactivated by user manually or after analyzing user behavior they can be activated or deactivated. So everything depends only on user hoe he/she behaves.



- Interfaces :

This module has some interfaces. First is login GUI. Here password is entered for login into system for authorization. An unauthorized person can not change system state.



In the second figure various types of events are shown where "call" and "message" are selected it means only for call and message events will be generated. If any event is recognized for other type of events then that event will be discarded by the system itself.

- Resources :

The whole system is developed on J2ME platform so it can be run on any mobile which support JAVA. This module doesn't require any special

hardware resources. When the system is started initial values are set which are changed by user or use by user behavior analysis.

- Processing :

As this module doesn't need any heavy processing so mobile processor is quite enough for it.

- Data :

All the initial values are set to true as system starts which are modifiable. There is a default password, when the whole system is reset, this password is set for the system. Its code is reusable so new features can be added very easily by a developer. User can change the state of the already provided features

4. Reuse and relationship to other products

SAEG has a very high scope of reuse. The whole system is divided into MIDlets and each MIDlet is developed using functions which are completely modifiable and reusable.

This software can be incorporated into any mobile which supports JAVA language. It doesn't require any heavy processing and also needs very less memory. More features can be added into this.

As it is a new idea and not yet developed by any other organization, this software shows how this can be developed. This can be made much better with many new features. This is also a boon for disabled persons by using text to speech tool. More graphics can be added and new looks can be given. This system is developed for English language which can be further generalized to other languages.

This system doesn't need any special type of hardware resources so it is compatible for all types of 3G mobiles.

This software can be incorporated into mobile in two ways. It can be integrated with mobile OS which is the right way of its working or it can be run as a separate service for demo. To work properly as a separate service it has its own phone book, calendar etc.

5. Design decisions and tradeoffs

This software is working only for 4 types of events which are:

1. Call
2. Message
3. Send number
4. Save number

There are some predefined verbs and negative words which are used to recognize the events. More verbs can be added by the user to recognize the event. As KVM supports only a few JAVA libraries so, processing is done very easy manner which is easy to understand. Tree structure was used to recognize the event but due to less power of CLDC 1.1 it was not possible to come up with any solution so pattern matching is used here.

6. Pseudocode for components

N . A.

7. Appendices

N. A.