Software Requirements and Specification Document for Smart Home Notification and Calendering System

Team 2: Hojun Jaygarl, Andrew Denner, Nam Pham

December 15, 2008

Version	Date	Author	Change
0.1	12/02/08	Andrew Denner	Initial Version
0.11	12/03/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Section 2 Added
0.2	12/04/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Sequence Diagrams
0.21	12/04/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Section 3 Added
0.22	12/05/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Use Case diagrams
0.23	12/06/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Class diagrams
0.3	12/07/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Appendix
0.31	12/08/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Definitions
0.32	12/09/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Fixing gramatical errors
0.4	12/10/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Reference
0.5	12/14/08	Hojun Jaygarl, Nam Pham, Andrew Denner	Motivation/Conclusion

Contents

1 Introduction **4**
- 1.1 Purpose . . . 4
- 1.2 Scope . . . 4
- 1.3 Definitions, acronyms, abbreviations . . . 4
- 1.4 References . . . 4
- 1.5 Overview . . . 4

2 Overall Description **5**
- 2.1 Product Perspective . . . 6
  - 2.1.1 Concept of Operations . . . 7
- 2.2 Product functions . . . 8
  - 2.2.1 Scheduler . . . 9
  - 2.2.2 Medical Monitor . . . 12
  - 2.2.3 Security/Fire monitor . . . 13
  - 2.2.4 Nutrition Monitor/Planner . . . 14
  - 2.2.5 Configuration System . . . 16
  - 2.2.6 Medical Consult . . . 17
  - 2.2.7 Degraded Function Mode . . . 18
  - 2.2.8 Priortizer . . . 20
  - 2.2.9 User Level Control . . . 21
  - 2.2.10 Caretaker Control . . . 22
  - 2.2.11 Proactive Alert/Notify System . . . 24
  - 2.2.12 Reactive Alert/Notify System . . . 25
- 2.3 User characteristics . . . 26

3 Specific Requirements **27**
- 3.1 External Interface Requirements . . . 27
  - 3.1.1 User Interfaces . . . 27
  - 3.1.2 Hardware Interfaces . . . 28
  - 3.1.3 Software Interfaces . . . 29
  - 3.1.4 Communication Interfaces . . . 29
- 3.2 Classes . . . 29
- 3.3 ClassDiagram . . . 30
  - 3.3.1 Congifuration . . . 30
  - 3.3.2 Medical consult . . . 30
  - 3.3.3 Degraded function mode . . . 31
  - 3.3.4 Prioritizer . . . 32
- 3.4 Scheduler . . . 32
- 3.5 Medical Monitor . . . 33
- 3.6 Security Monitor . . . 34
- 3.7 User Level Control . . . 35
- 3.8 Caretaker Control . . . 35

3.9 Proactive Alert 35
3.10 Reactive Alert 36
3.11 Nutrition Monitor/Planner 36
3.12 Performance Requirements 36
3.13 Software System Attributes 37
3.13.1 Reliability/Dependability 37
3.13.2 Security 37
3.13.3 Availability 38
3.13.4 Maintainability 38
3.13.5 Repair-ability: 38
3.14 Design Constraints 38
3.15 Other Requirements 39

4 Appendix A **40**
4.1 Motivation: 40
4.2 BDI Logic 41
4.3 Rhapsody 43
4.4 Conclusion: 45

1 Introduction

1.1 Purpose

This document details both the functional and nonfunctional requirements for the Smart Home Notification and Calendering System (NCS).

This document serves as a contract between the team members of the Smart Home Project(SHP) at Iowa State University to ensure fulfillment of project requirements and to describe the inner workings of the NCS and it's interactions with the SHP.

1.2 Scope

This document covers the functional and non-functional requirements of the NCS including the physical description of the system as well as the behavioral and other factors necessary to provide a complete and comprehensive description of the NCS.

1.3 Definitions, acronyms, abbreviations

Term	Description
Smart Home (SH)	Home Automation system that provides Comfort, Convenience and Safety
Notification System	A combination of software and hardware that provides a means of delivering a message to a set of recipients.
Calendering System	A software that helps people to manage and plan schedules
NCS	Notification and calendering System
ADLs	Active Daily Lives
SRS	Software Requirement Specification Document
RFID	Radio Frequency Identification Device

1.4 References

ADD ME LATER

1.5 Overview

The user demographic of the Smart Home is primarily elderly and disabled people. This particular demographic is often scared by a myriad of different software applications for CNS systems. These pre-existing software solutions are often far too complex and not specifically suited to the unique needs of this user demographic. At the same time our users are faced with potential decline in cognitive function and an ever increasing set of medical appointments and other senior activities as well as new dietary needs (such as low cholesterol diets, diabetic concerns, etc.). These increasing needs coupled with a desire to continue to be independent underlay the requirements and design of the entire Smart Home as well as the CNS specifically.

2 Overall Description

This section seeks to clarify and describe the requirements of Smart Home NCS. Basically, the system provides schedule management, notification and planning. Since the target system is Smart Home and the target user is elderly or disabled people, it has design constrains and specific features different than general NCS. For example, a user interface should not to be difficult. It should be simple and easy to understand. Moreover, features need to be related to safety and health of users and provide useful information for the target user.

The NCS of the Smart Home has a number of crucial functionality. Calendaring functions give specific scheduling plan for elderly people such as appointments for a doctor, nurse, community event, and medicine and meal plan. Notification functions provides information about safety, emergency situation, health status, and security.

- Calendaring - Functional features
  - Scheduling transportation to community events
  - Scheduling a visit to doctor
  - Consulting with a nurse (virtual consultation)
  - Dispensing medications and taking pills
  - Recording exercise activities
  - Refilling prescriptions
  - Nutrition and Meal Planning
- Notification - Functional features
  - Monitoring health status (glucometer, blood pressure, spirometer, pulse oximeter, weight scales, and etc.)
  - Calling for help after a fall
  - Turning off the range automatically when it gets too hot or stays on too long
  - Looking to see who is at the front door (without opening the door)
  - Providing long-distance monitoring for safety
  - Requesting house cleaning or home maintenance services Checking security system Schedule for house work (When did I clean, when should I change a purifier filter, air filter, and so on) Making shopping list automatically Checking bank statements Paying bills via Internet
  - Requesting house cleaning or home maintenance services
  - Checking security system
  - Making shopping list automatically
  - Checking bank statements
  - Paying bills via Internet
- Non-functional features

- Universal Design for elderly people
- Safety constrains
- Reconfigurable System

2.1 Product Perspective

This section is borrowed from the vision document by Chad Kilgore, Matt Peitz, Kendra Schmid.

Assisted Living Facilities: Assisted living facilities are for persons that need assistance with ADLs but wish to live as independently as possible. Most assisted living facilities create a detailed service plan for individual residents upon admission, which is updated regularly to assure that the resident receives the appropriate care as his or her condition changes. These services include help preparing meals, bathing, dressing, performing household chores, and aid for persons confused or experience memory problems. Other common terms for assisted living are residential care, personal care, adult congregate living care and supported care.

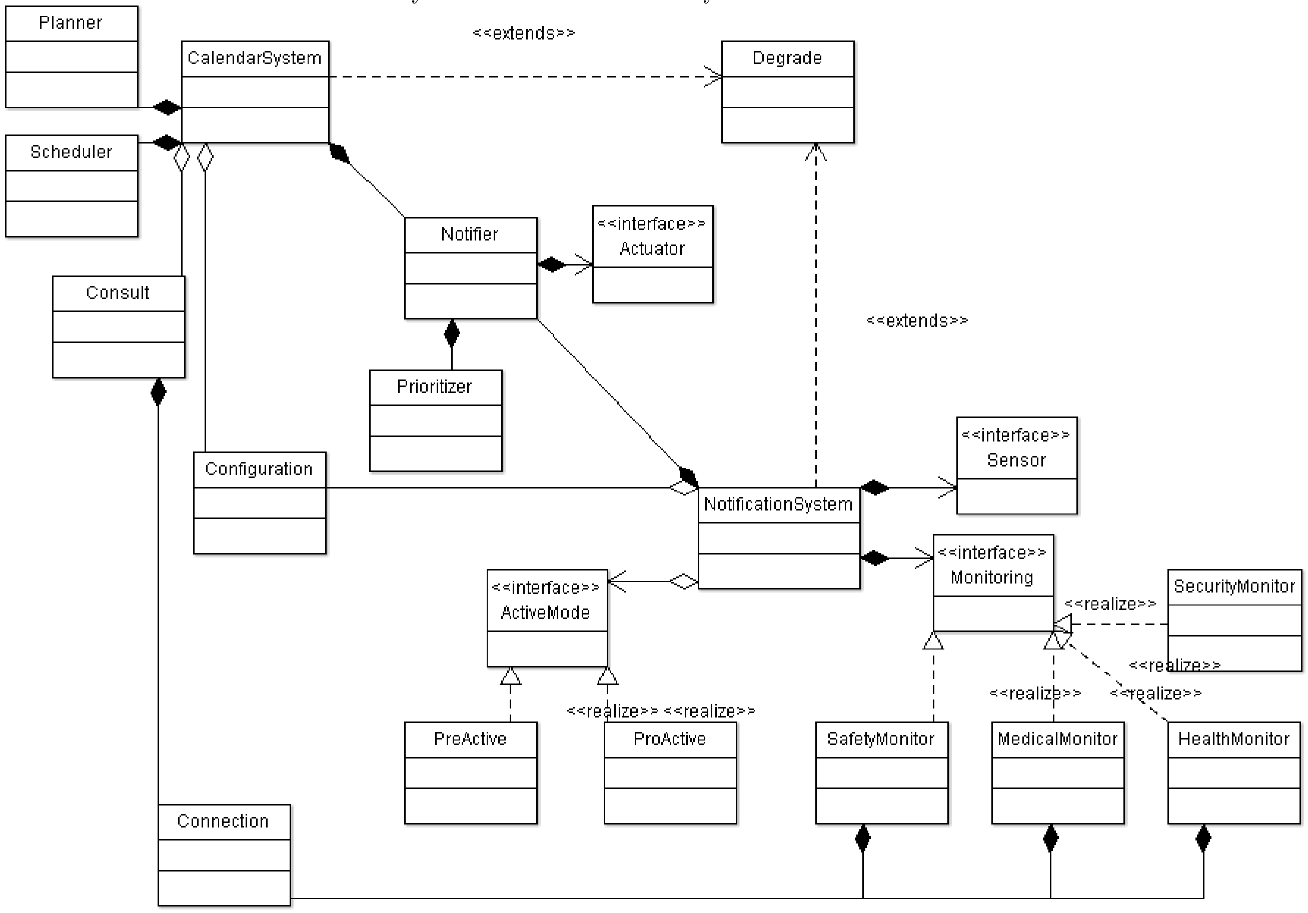
Nursing Homes: Nursing homes provide skilled nursing care and rehabilitation services to people with illnesses, injuries or functional disabilities. Even though most facilities serve the elderly, some facilities provide services to younger individuals with special needs, such as the developmentally disabled, mentally ill, and those needing drug and alcohol rehabilitation. Nursing homes are generally stand-alone facilities, focusing their attention on rehabilitation, so that their residents can return to their own homes as soon as possible. Some of the services provided by nursing homes include therapies, pharmacy services, and specialty care, such as Alzheimer's treatment, neuromuscular diseases, stroke recovery.

University of Florida: "The RERC-Tech-Aging is testing currently available home monitoring products relative to their effectiveness in relation to independence, quality of life and health related costs. The RERC-Tech-Aging is also identifying needs and barriers to home monitoring and communication technology, and addressing needs of special populations including rural-living elders, and people aging with disability. The results of this research will be relevant to health policy makers, device developers, and other investigators. The RERC-Tech-Aging works with companies on pre-product testing, including Honeywell's very promising Independent LifeStyle Assistant (ILSA). We are advancing very new consumer products such as Motorola's Smart Phone, to provide applications useful for older people with disabilities. We are also studying the requirements for, and development of a device / system for elders with cognitive impairment. We are applying the concept of pervasive computing to the needs of older persons through our work with smart phones and smart homes."

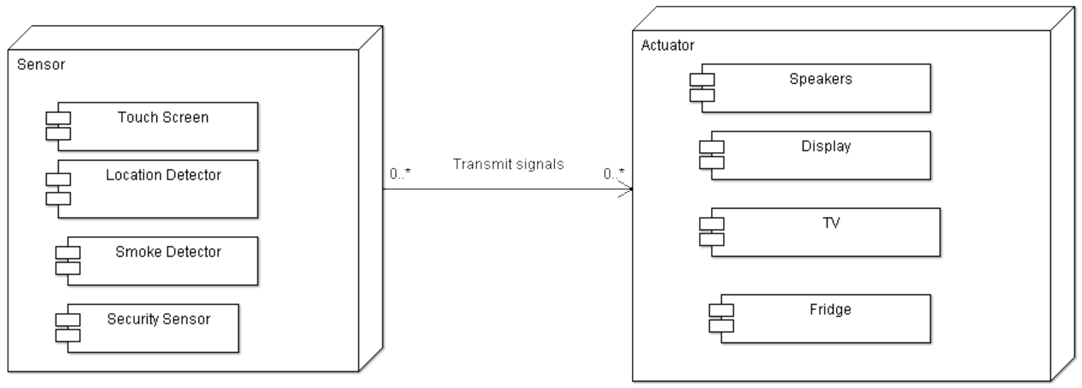
Niagara Framework: The Niagara Framework, developed by Tridium Software, is a Java-based framework that allows development of smart homes using different components from different manufactures. Niagara was developed using Java Beans that allows every component to be treated as an object. Niagara uses an adapter development pattern so that every object, regardless of manufacture, communication standard, or software can run from a standard web browser. The framework can then dictate to each device what needs to be done its own protocol, thus saving time that would be used to program all devices. The framework operates on a wide variety of hardware platforms and operating systems due to Java development. The Niagara Framework differs from the project vision since the framework is designed primarily for an electronic house, not necessarily a smart home.

2.1.1 Concept of Operations

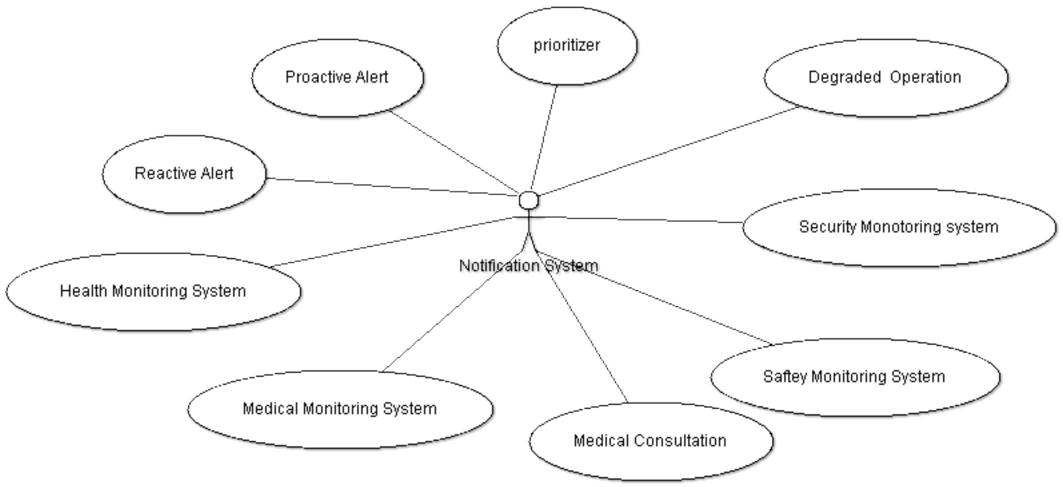
The system will not only help users in daily tasks such as scheduling and shopping, it also protects a user inside the house with a health care and nutrition system and protects the security of the house. The system has an easy-to-use interface and is configurable. The system is fault tolerant, for example, in the worst case the system does not do any harm to users. With an interactive interface, the system can visualize concepts and communicate with users in friendly and understandable ways.

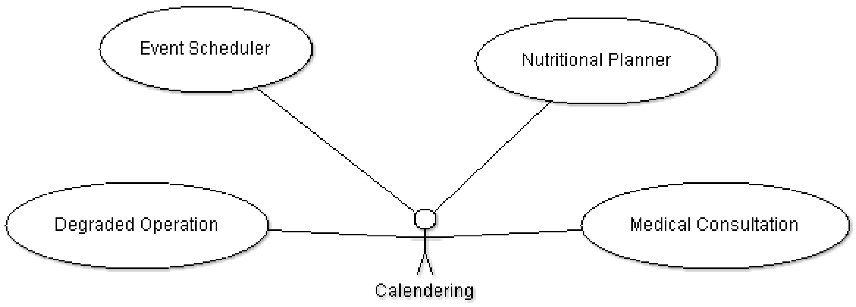


The class diagram shows basic concepts of our system in a class perspective view. There are two main classes, Notification System and Calendar System. These classes manage notification and calendaring function respectively. Both classes extend Degrade class that detects system hazards and change a mode of the system to degraded mode. Notifier class manages notifications and alarms to send out and uses Prioritizer class to give priority to notifications and alarms. The configuration class provides an interface and functions to configure the system. Both the Calendaring System and Notification System uses the Notifier and Configuration classes. The Calendaring System class uses classes such as Planner for planning schedule such as health plan, food plan and, exercise plan, the Scheduler for scheduling, and the Consult class for getting medical consultation services. The Notification System class uses classes such as Reactive and Proactive Alert and notification systems, Monitoring , and Sensors. The Notification System detects safety, medical, health, security issues and notifies users and service providers such as a hospital, police, or fire station.

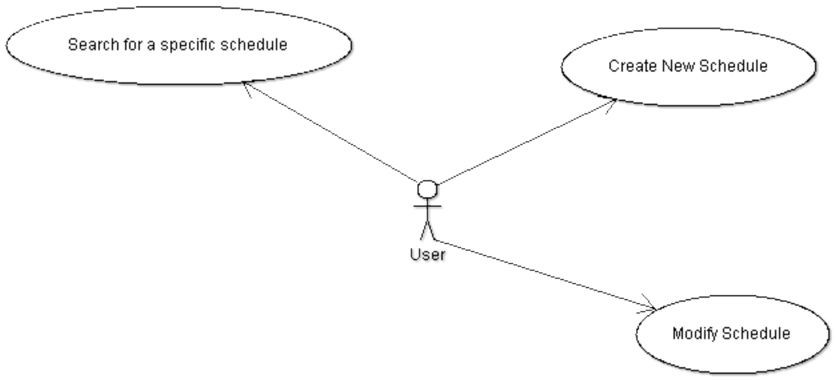


2.2 Product functions





2.2.1 Scheduler



Tittle: Scheduler

Description: User needs to make or modify a schedule

Actor: User

Preconditions: The system is on and the user wants to make/modify a schedule

Post conditions: The user has made/modified a schedule and the system records changes in scheduling database.

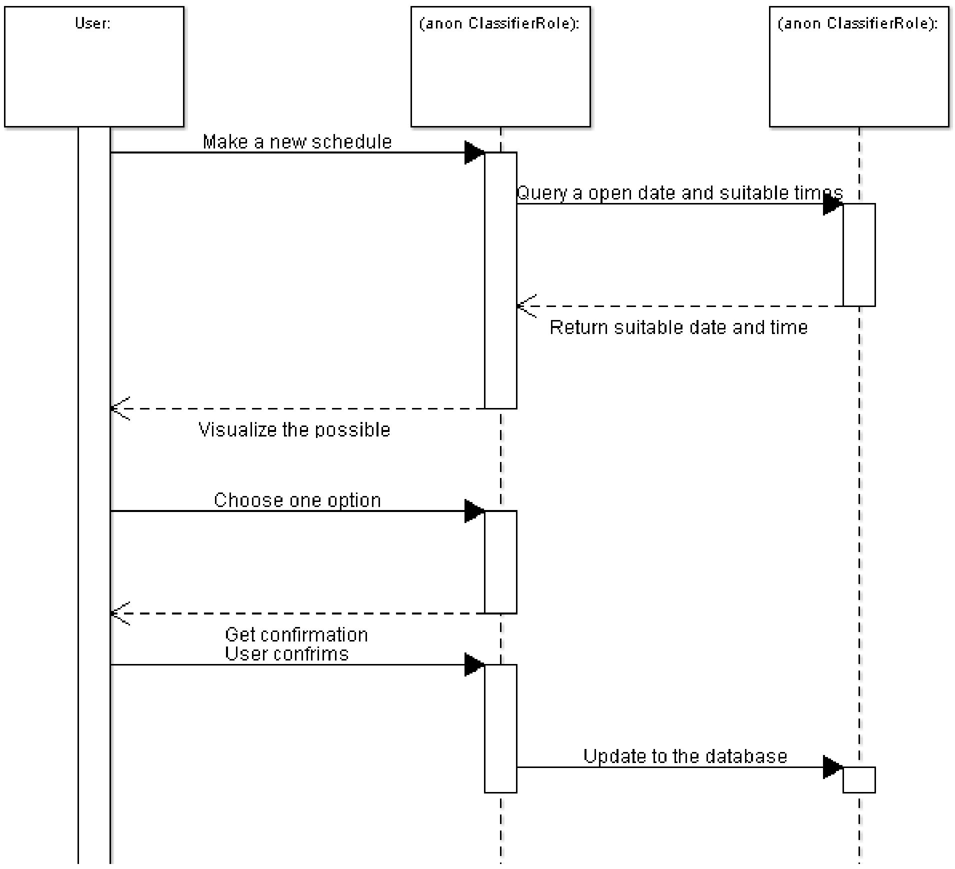
Basic flow:

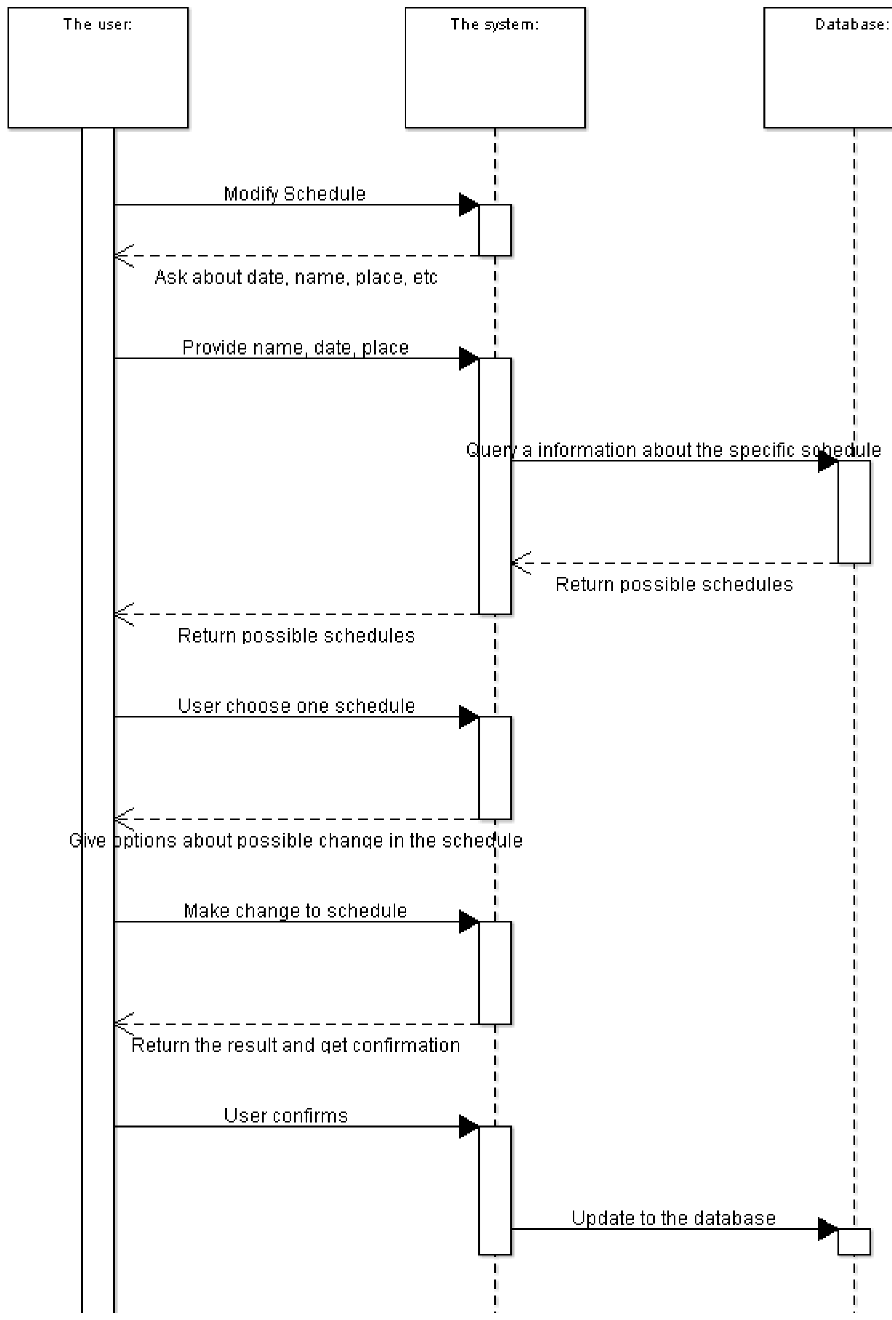
1. The user informs the system to make a schedule with given preference like date, time or place.
2. The system identifies which time and place is available and suitable (e.g weather conditions, traffic)for user's normal schedule

3. The system visualize available times and suitable places for the user to choose
4. The user decides which time is suitable
5. The system get confirmation from user and store in the database

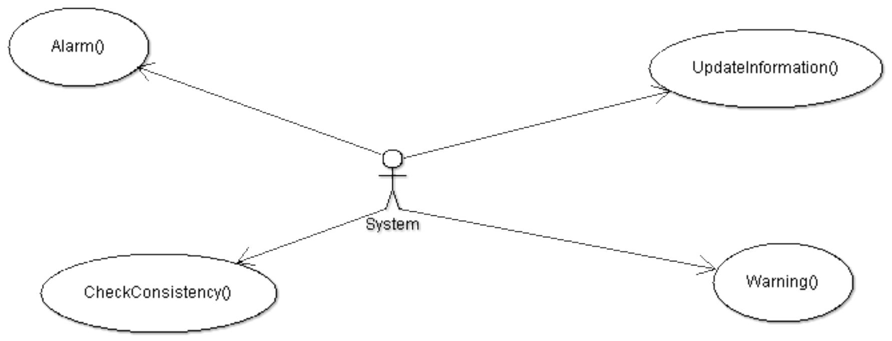
Alternate flow:

1. The user wants to modify a schedule given name,date or time of meeting
2. The system searches through the database for possible matched schedules and visualize them for user
3. The user chooses the right one
4. The system displays options: correct the meeting time, cancel an appointment, change the meeting place.
5. The User chooses one option
6. The system follows the user's choice and continue until the user finishes modification
7. The system get confirmation about the modifications and store in the database




The user:
The system:
Database:
Modify Schedule
Ask about date, name, place, etc
Provide name, date, place
Query a information about the specific schedule
Return possible schedules
Return possible schedules
User choose one schedule
Give options about possible change in the schedule
Make change to schedule
Return the result and get confirmation
User confirms
Update to the database

2.2.2 Medical Monitor



Title: Medical Monitor

Description: The system monitor the health status of the user

Actors: The system and user

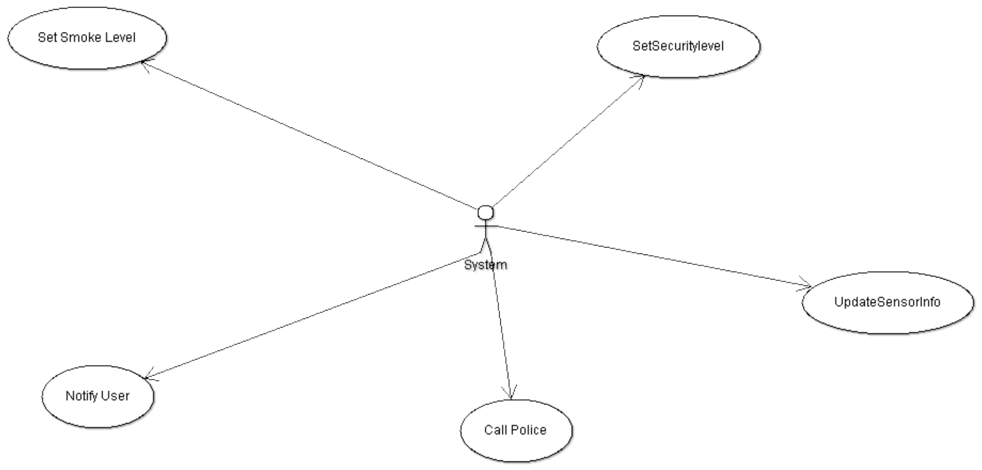
Pre conditions: The system is on and all sensors are in working condition

Post conditions: The information about health status of users is obtained.

Basic flow:

1. The system receives medical information from all sensors such as blood pressure, glucometer level, spirometer, pulse oximeter, weight scale, etc
2. The system checks the consistency with the information in previous period.
3. The system notifies the user immediately when the obtained information is abnormal.
4. If there is no response from user in specific amount of time, the system raises an alarm and calls the hospital for emergency case.

2.2.3 Security/Fire monitor



Title: Security/Fire monitor

Description: The system provides security monitoring for users and the home

Actor: The system and the user

Pre Conditions: The system is on.

Post conditions: The house and the customer are safe.

Basic flow:

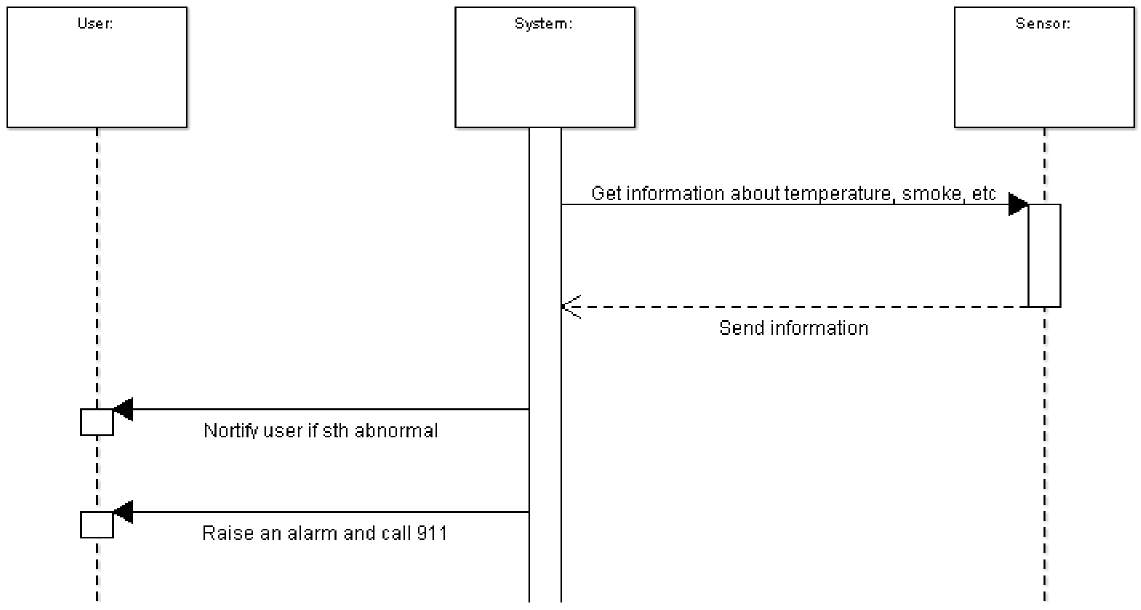
1. The system receive information from medical monitor and sensors like temperature sensor, smoke sensor, etc around the house.
2. Anything abnormal happens, the system give warnings to user via communication devices like cell phone, email, etc.
3. After a specific of time without any confirmation from user, the system raises an alarm, and calls 911.

Alternate Flow:

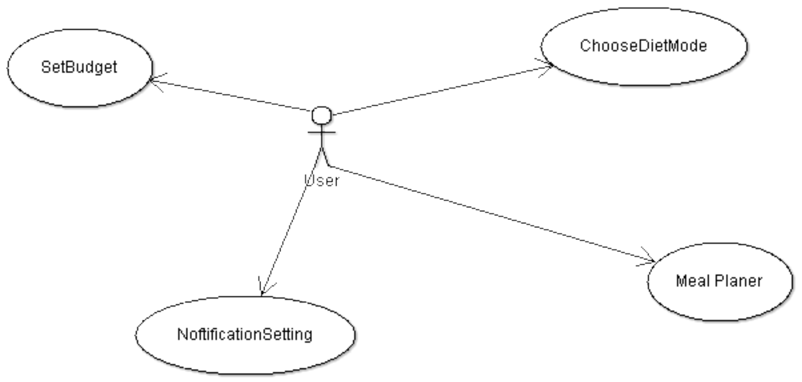
3(a). The user turns off the alarm.

4. The system verify the user's identity

5.The system turns off the alarm when the identity is clear


User:
System:
Sensor:
Get information about temperature, smoke, etc
Send information
Nortify user if sth abnormal
Raise an alarm and call 911

2.2.4 Nutrition Monitor/Planner



Title: Nutrition Monitor/Planner

Description: Monitor food usage and help the user choose right food plan

Actor: The system and the user

Preconditions: The system is in working mode and the user's privilege is clear

Post conditions: The nutrition level in user's meal is acceptable.

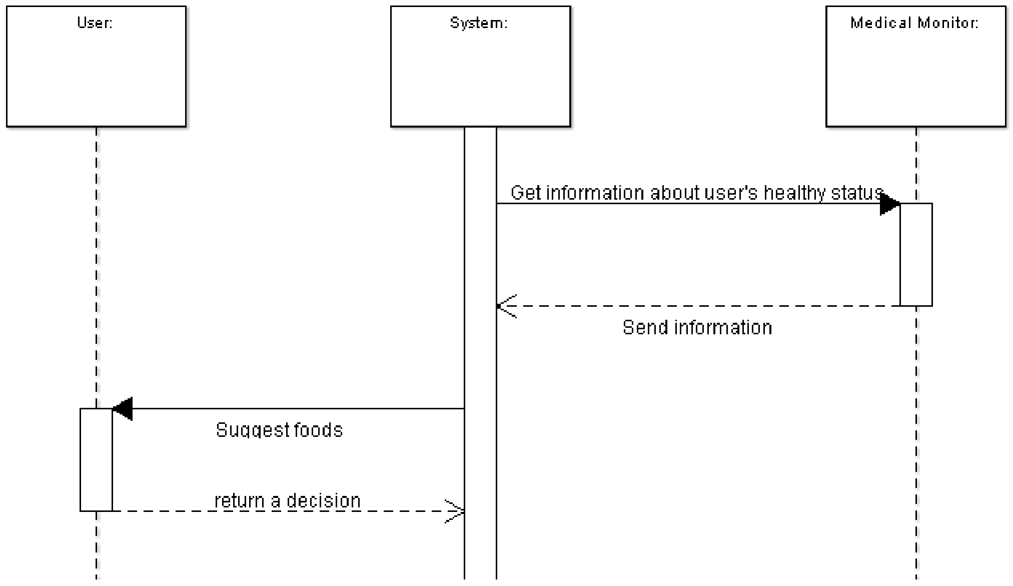
Basic flow:

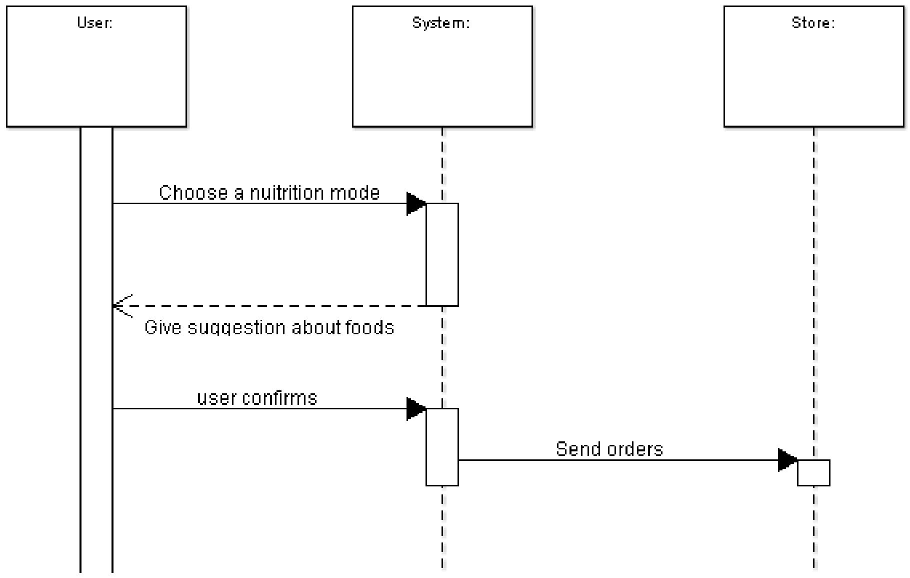
1. The system gets information from medical monitor to get health status of the user.
2. The system suggests the user which food is good for current status of user.

3. The user chooses specific food from suggestions.
4. The system send the food order to the market.

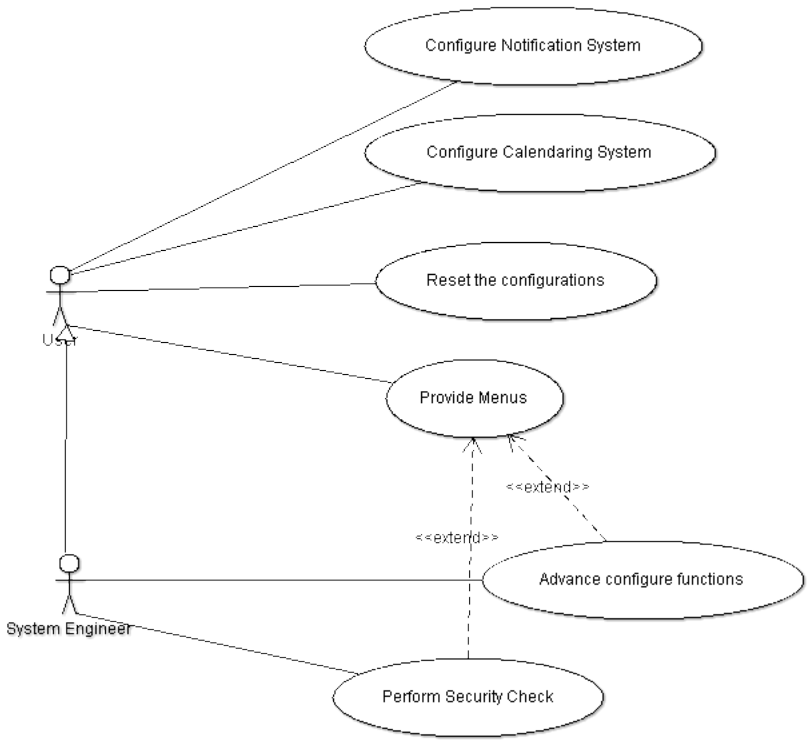
Alternate:

1. The user chooses a nutrition mode: normal or diet.
2. The system compares the health status of user and the mode.
3. If something conflicts, the system give a warning to the user.
4. If everything is fine, the system schedules and orders suitable food for user's nutrition mode.





2.2.5 Configuration System



Title: Configuration System

Description: Configuration system provides an interface to configure system.

Actor: User, system engineer, service provider

Pre Conditions: The system is on and configuration system is available for use.

Post Conditions: The system has been configured.

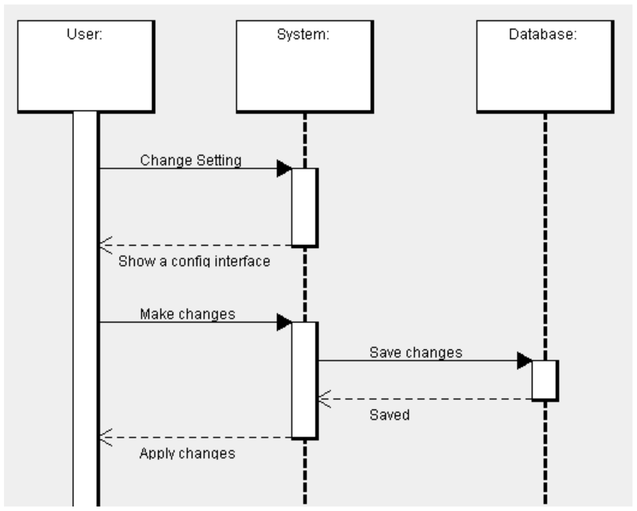
Basic Flow:

1. The customer instructs the system to be configured.
2. The system provides an interface to configure system.
3. The customer configures the system through the provided interface.
4. The system notifies the customer that the system has been configured.

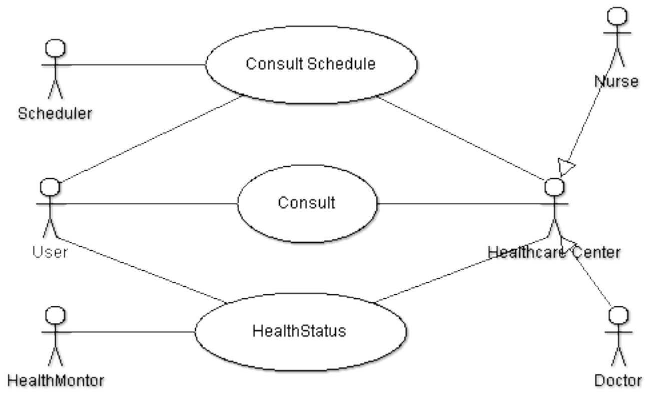
Alternate Flow:

1a) The system engineer instructs the system to be configured.

1. The system provides a richer interface to configure the system.



2.2.6 Medical Consult



Title: Medical Consult

Description: Consults with a doctor or a nurse about the customer's health condition.

Actor: Customer

Pre conditions: The System is on and phone and/or network connections is available.

Post conditions: The Medical issue has been resolved.

Basic flow:

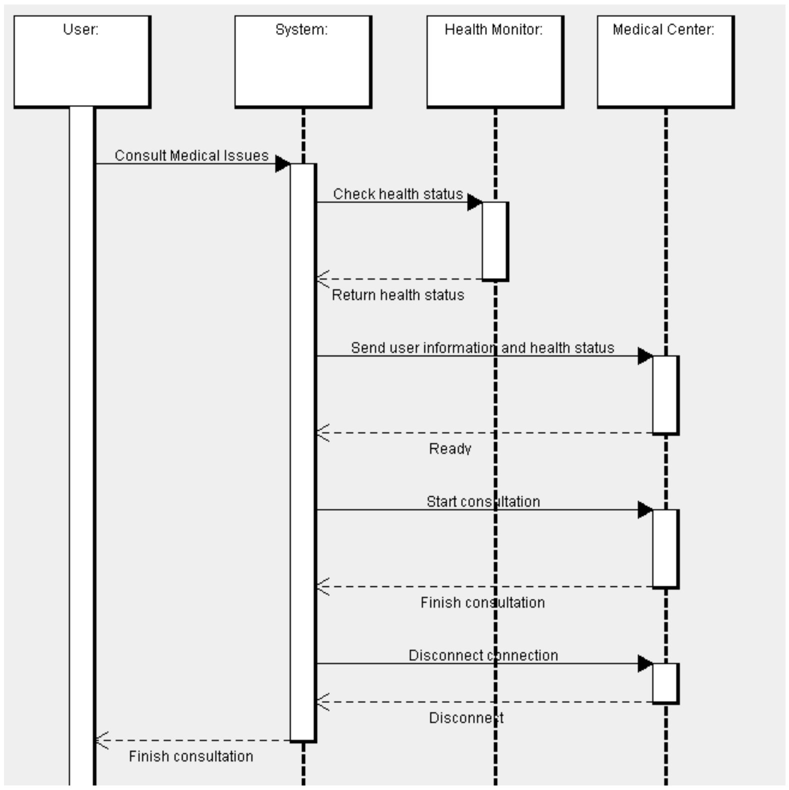
1. The customer instructs the system to connect a medical service center.
2. The system call a medical service center.
3. The customer consults with a doctor or a nurse about his/her condition.

4. The system disconnects the phone call.

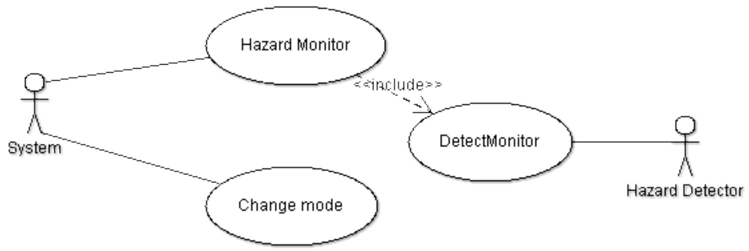
Alternate flow:

2a) If the other end of the connection as video conferencing equipment, use a video conference.

3a) If the system has a network connection, the system sends user's health status.



2.2.7 Degraded Function Mode



Title: Degraded Function Mode

Description: The Degraded mode keeps system working correctly through possible system hazard conditions. Possible hazards include blackout, network disconnection, sensor or device malfunctions and etc.

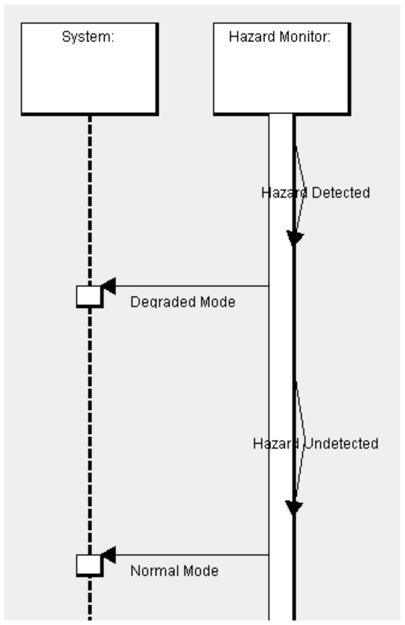
Actor: System

Pre Conditions: The system detects a hazard.

Post Conditions: The system keeps working.

Basic Flow:

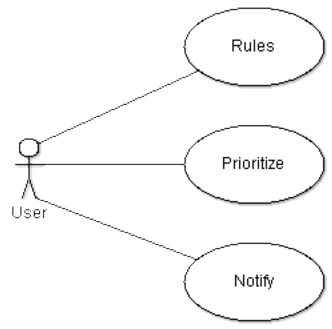
1. The system detects a hazard.
2. The system change its mode to degraded mode.
3. Depending on the type of hazard, the system performs the proper actions to keep working.



Alternate Flow:

3a) If the system can not continue operation, the service center detects failure through a heart-beat technique.

2.2.8 Priortizer



Title: Priortizer

Description: The prioritizer gives a function that prioritize notifications and alarms.

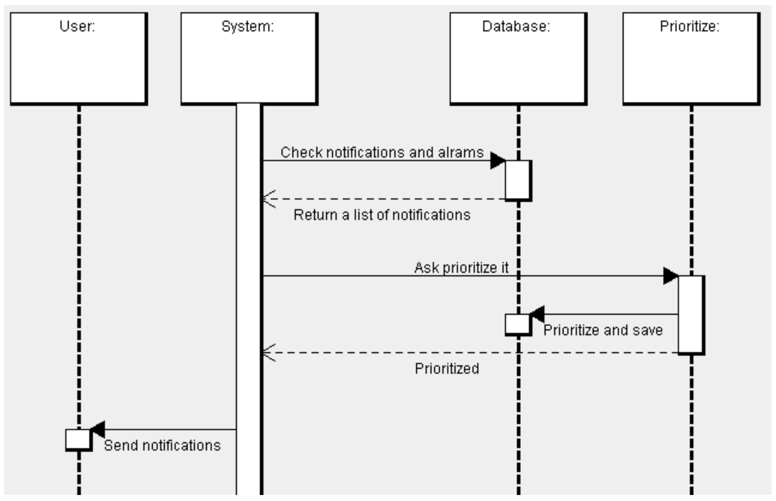
Actor: System

Pre Conditions: System has notifications and alarms in queue to be processed.

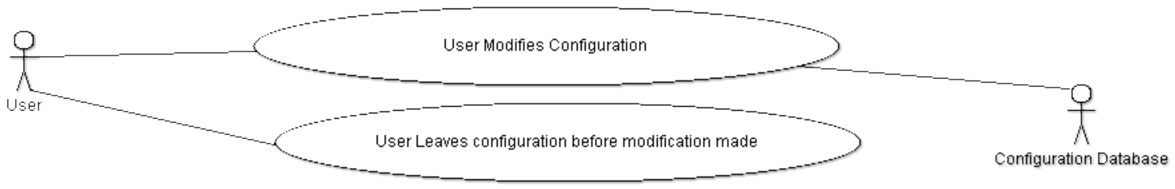
Post Conditions: Process the notifications and alarms based on priority.

Basic Flow:

1. The system detects notifications and alarms to be sent.
2. The system checks the rules to prioritize notifications and alarms.
3. The system reorders the notifications and alarms based on priority.
4. The system sends the notifications and alarms based on priority.



2.2.9 User Level Control


User
User Modifies Configuration
User Leaves configuration before modification made
Configuration Database

Title: User Level Control

Description: The user desires to make a modification to a setting, or create a new event or modify some other setting.

Actor: Customer

Pre Condition: The user is a properly authenticated user, and the system is on

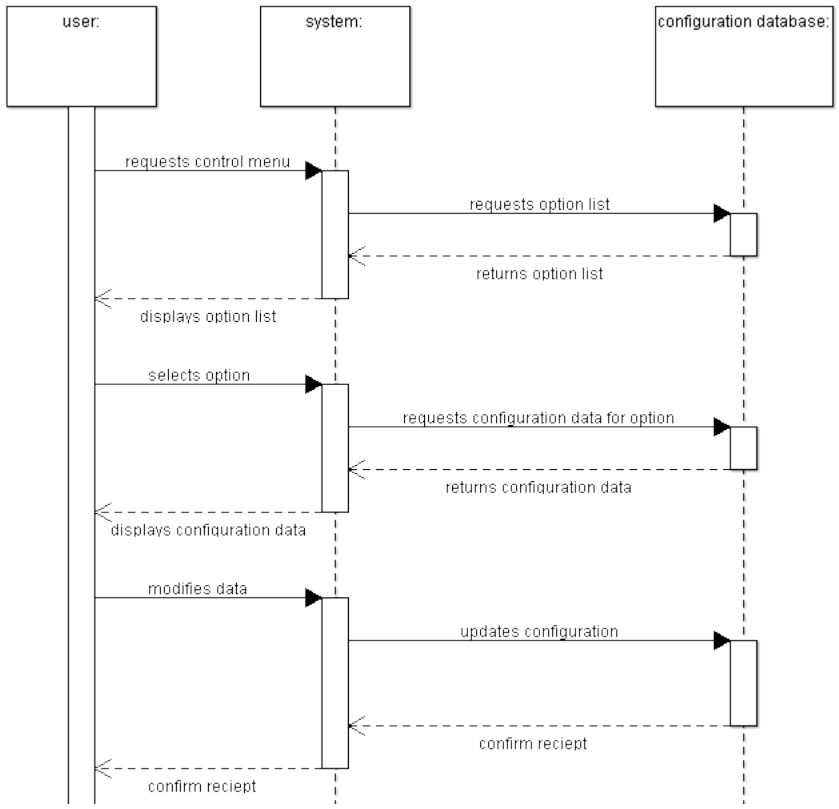
Post Condition: The system is updated with the change to the user's preference

Basic Flow:

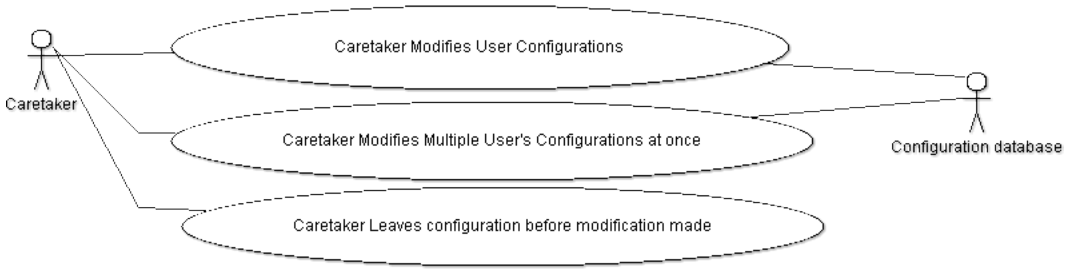
1. User selects the Control menu from listed options
2. System displays Control options for the User
3. User selects option to be changed
4. System displays data associated with option
5. User provides updated data
6. System saves configuration

Alternate Flow:

1. User leaves control before saving configuration
2. System disregards changes and returns to previous state



2.2.10 Caretaker Control


Caretaker Modifies User Configurations
Caretaker
Caretaker Modifies Multiple User's Configurations at once
Configuration database
Caretaker Leaves configuration before modification made

Title: Caretaker Control

Description: The caretaker desires to make a modification to a user

Actor: Caretaker

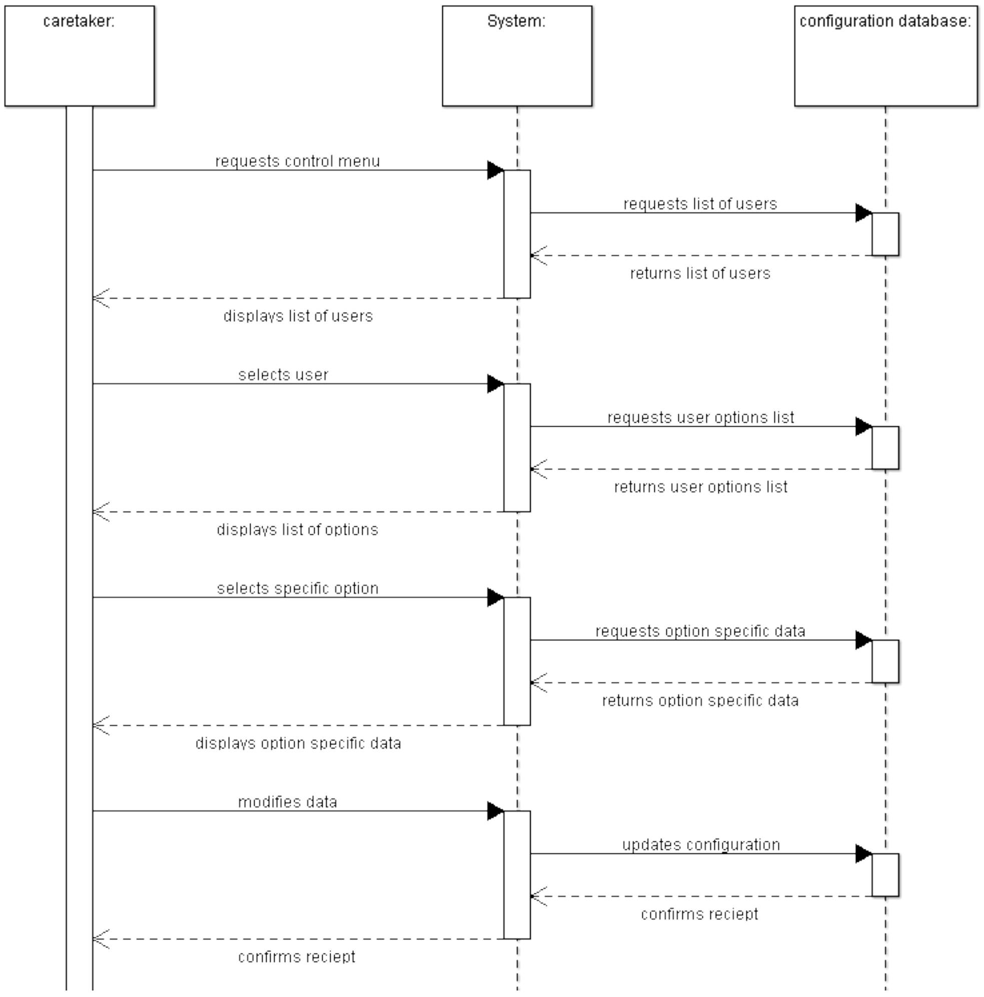
Pre Condition: The user is a properly authenticated caretaker and the system is in caretaker mode

Post Condition: The system is updated with the change to the user's preference

Basic Flow:

1. Caretaker selects the control menu from listed options
2. System displays list of users for the caretaker

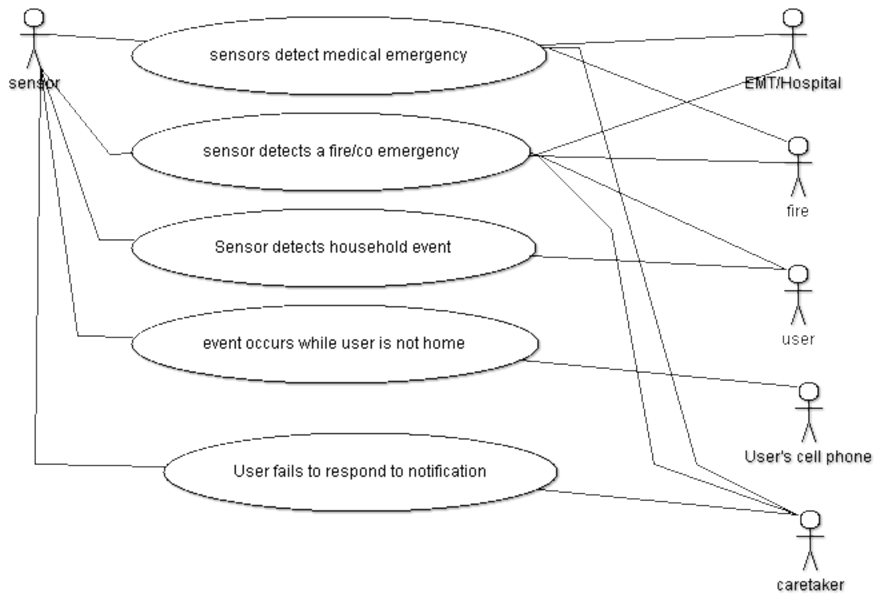
3. Caretaker selects specific user to preform actions on
4. System displays a list of options associated with user
5. Caretaker selects specific option to modify
6. System displays data associated with option
7. Caretaker provides updated data
8. System saves configuration



Alternate Flow:

1. User leaves control before saving configuration
    (a) System disregards changes and returns to previous state
2. a. Caretaker selects multiple users to modify at the same time

2.2.11 Proactive Alert/Notify System


sensor
sensors detect medical emergency
sensor detects a fire/co emergency
Sensor detects household event
event occurs while user is not home
User fails to respond to notification
EMT/Hospital
fire
user
User's cell phone
caretaker

Title: Proactive Alert/Notify System

Description: The CNS notifies/alerts user/caretaker of a pre-scheduled event. An example of this would be notification of the need to take medicine or of an appointment

Actor: Customer, Caretaker

Pre Condition: A pre scheduled event is about to occur and the system is in a ready state

Post Condition: The user/caretaker has been notified and acknowledges the event or alert

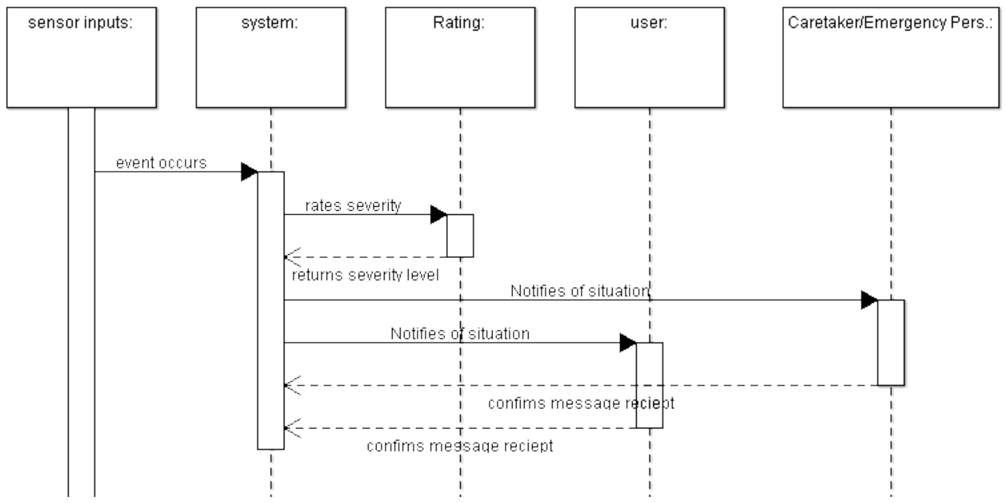
Basic Flow:

1. An event has been previously scheduled with the CNS system and is about to occur (as defined in user preferences, i.e. user specified that they desire notification 20 minutes before scheduled Medical Consultations)
2. System determines location of user through proximity sensors and RFID tagging
3. System sends notification to device near user
4. User acknowledges alert

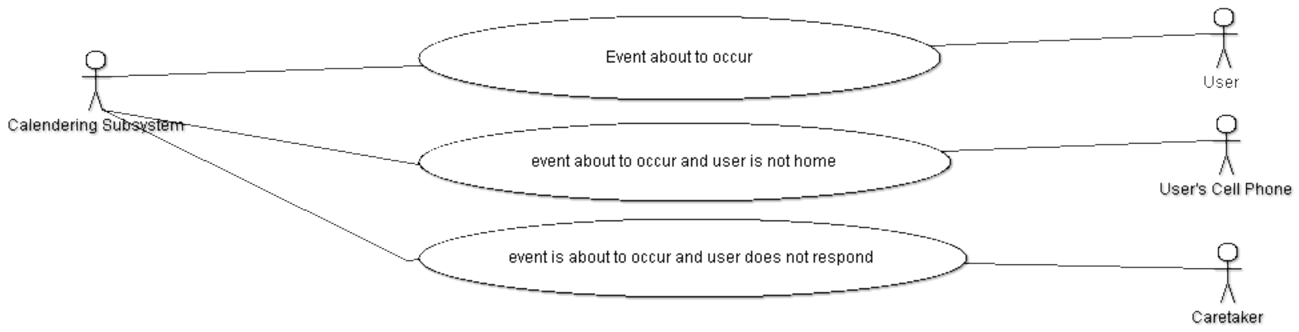
Alternate Flow:

1. System can not find user
   (a) System sends notification via Cellphone Text Message (SMS) to user's phone
   (b) User acknowledges via a text message
2. User does not respond to alert

   (a) System re-sends alert to a wider display set near user (as determined by RFID and proximity sensors)
   (b) User acknowledges request
3. User still does not respond
   (a) Caretaker is notified of potential issue



2.2.12 Reactive Alert/Notify System


Calendering Subsystem
Event about to occur
event about to occur and user is not home
event is about to occur and user does not respond
User
User's Cell Phone
Caretaker

Title: Reactive Alert/Notify System

Description: The Reactive Alert system notifies user/caretaker/emergency personnel to an event that has just occurred

Actor: User, Caretaker, Emergency Personnel

Pre Condition: An event has occurred that requires intervention from the user, caretaker, or emergency personnel depending on level of severity

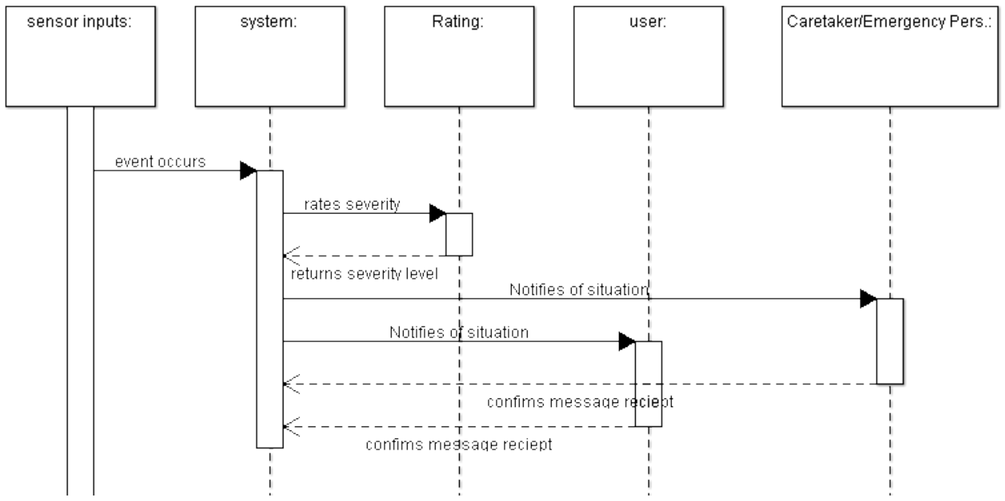
Post Condition: Proper actors notified of issue and activated

Basic Flow:

1. Some unplanned event occurs. Examples of events include:
   (a) Medical: stopped heart, low blood sugar, user fall, or other medical issue
   (b) Safety: Break in, fire, or pipe burst detected
   (c) User error: stove left on, refrigerator door left open
   (d) Maintenance: sensor battery dead, not responding, equipment failure
2. The level of severity of the issue is rated by the system using predefined templates
3. The system notifies the necessary actors based on severity level of the event.

Alternate Flow:

1. Actor does not respond to notification
2. System escalates notification to the next higher level of notification.



2.3 User characteristics

The CNS subsystem, as is the whole smart home, is targeted at elderly and disabled people who still desire to live at home, however need added assistance to function in their every day to day lives. These users do not generally have a high amount of aptitude using computerized systems and are scared by new technology. Finally, they have a myriad of medical and health concerns and other scheduling concerns to keep track of.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The primary goals of the NCS user interfaces are accessibility, universality, and reachability. The user interface is comprised of input and output devices. The input devices shall have the following components:

- Touch Screen
  - The touch screen interface must be unscrachable, finger controlable, and multi touchable.
- Mobile Device
  - The mobile device interface must be small, attachable, and compatible across different devices.
- Voice Recognition
  - The voice recognition system must accurately understand commands and, be able to handle noisy environments.
- Remote Controls
  - The remote control must be simple, have big buttons, and have universal compatibility with devices in the smart home.
- Motion Detector
  - The motion detector must be able to accurately detect motion with out error. It also must be able to operate in low and no light conditions.

The NCS user interface shall also support the following output methods:

- TV
  - The television output device must be able to display information in a large and easy to read format.
- Mobile Device
  - The mobile device output must be able to display Wireless Markup Language (WML) pages as well as receive SMS text message notifications.
- Voice
  - The voice output system must be able to accurately announce messages. The system also must be configurable and have a selection of voices as many of the users of the system suffer from forms of hearing loss in some frequencies.

3.1.2 Hardware Interfaces

Basically NCS has sensors to get data and actuators to give a physical service. To handle getting data and make a service, NCS has a main computer so-called Home Server and also terminal so-called client computer to provides an user interface.

Sensors :

- RFID(location)
  - RFID and location sensors check user's position in Smart Home environment.
- Motion
  - Motion sensor catches user's motion by detects hand motion, eye direction, etc.
- Smoke, thermal, CO-detector
  - Smoke, thermal and CO-detector sensors detect fire.
- Body sensors(blood pressure, glucometer level, spirometer, pulse oximeter, weight scale)
  - Body sensors is essential to check the customer's health status.

Actuators:

- Auto door
  - Opening and closing a door is challenge for elderly and disability people. Auto door provides convenience for them.
- Light switch
  - Based on user's location and behavior, light system can be smart. The system turn on/off lights through the light switch.
- Sprinkler
  - When the system detects fire, an initial action is very important to prevent tragedy. Sprinkler system provides water to extinguish fire.
- Auto window
  - As auto door, auto window provides automatic open/close functions for a window.
- Alarm(audible, visible)
  - If the system detects safety, security or emergency situation, the system alert it through the audible and visible alarm actuators.

These sensors and actuators are connected to the home server computer through OSGi(Open Services Gateway Initiative) interface. OSGi supports component-based software. OSGi is an open industry framework and service-oriented architecture. OSGi also provides deployment of services in platforms. OSGi defines a framework to mange bundles (units of distribution) and the services they export Services can be obtained by querying the framework through a set of properties

3.1.3 Software Interfaces

As the NCS is a subsystem of the Smart Home System, the NCS shall follow the standards set forth for the SHS software interfaces. Furthermore, the NCS shall provide an a standardized API and protocol that will enable other subsystems of the smart home to communicate with it.

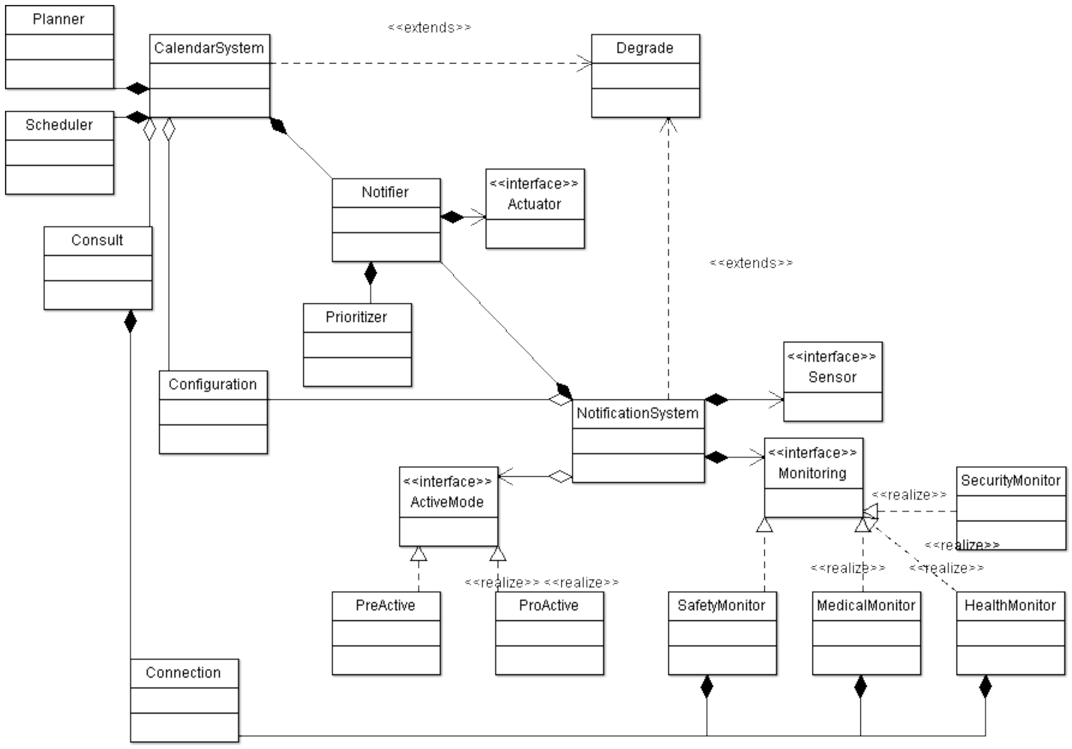
The Calendering System shall use the ical file format as defined in RFC 2445 as its data format for transferring information between itself and the rest of the smart home system as well as external systems as defined in section 3.1.4.

The Notification system also shall use an XML file format for communication with the rest of the home.

3.1.4 Communication Interfaces

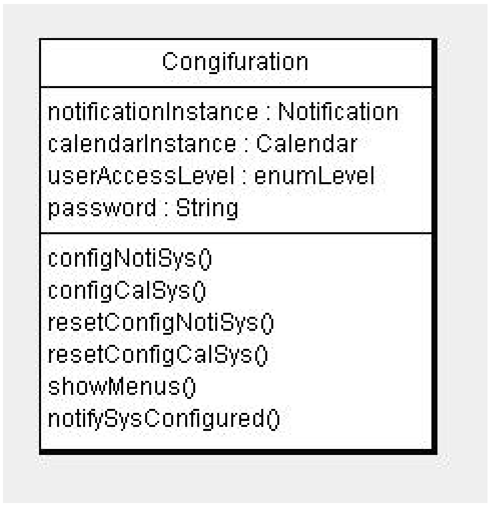
- The system shall be connected to the Internet/LAN.
- The system shall connect with the telephone lines.
- The system shall has a connection with an emergency protocol which is connected to a hospital, police, and fire station.

3.2 Classes



3.3 ClassDiagram

3.3.1 Congifuration



- notificationInstance : an instance of Notification class.

- calendarInstance : an instance of Calendar class.

- userAccessLevel : depends on user levels, configuration access modes are different. e.g. user and system engineer have different userAccessLevel.

- password : if userAccessLevel is higher than a system engineer, it needs password.

+ configNotiSys() : configure a notification system.

+ configCalSys() : configure a calculation system.

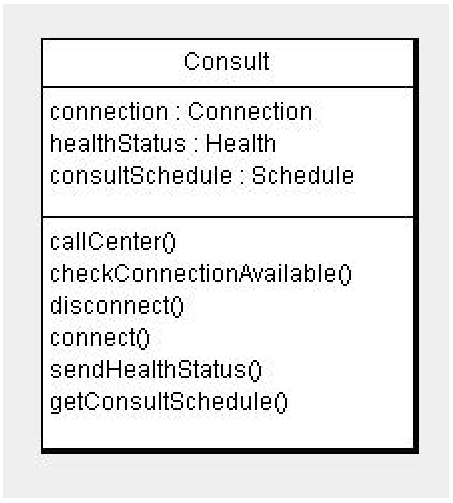
+ resetConfigNotiSys() : reset to default settings of a notification system.

+ resetConfigCalSys() : reset to default settings of a calendaring system.

+ showMenus() : show menus for configuration

+ notifySysConfigured() : after configured, notify it to the user.

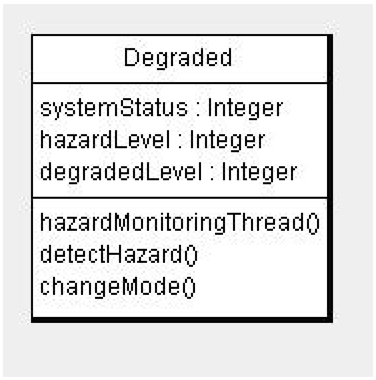
3.3.2 Medical consult



- connection : an instance of Connection Class. Connection Class manages network, phone call connections.

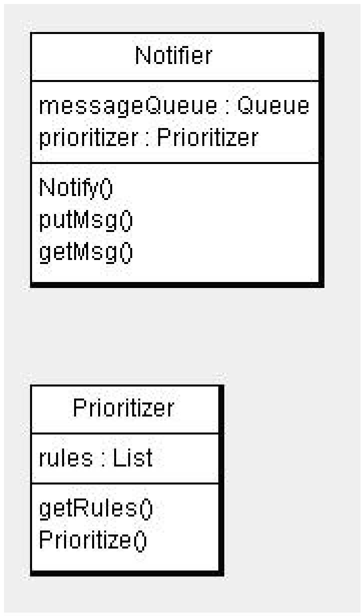
- healthStatus : an instance of HealthStatus. It has an information of the user's current health status.
- consultSchedule : Consulting schedule instance.
+ callCenter() : call the health center through phone call.
+ checkConnectionAvailable() : check available connections such as network and phone line.
+ disconnect() : after consulting, disconnect the connection.
+ connect() : connect to the network to send health status.
+ sendHealthStatus() : send health status of current user via network.
+ getConsultSchedule() : getConsultingSchedule information.

3.3.3 Degraded function mode



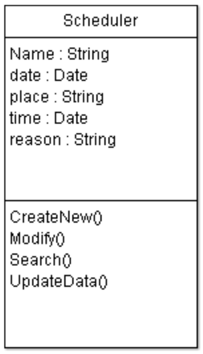
- systemStatus : represents current system status.
- hazardLevel : if it is normal status, hazardLevel is 0. If there is a hazard shows the hazard level from 1 to 10.
- degradedLevel : If there is a hazard, change the system mode to this degradedLevel.
+ hazardMonitoringThread() : Thread method to constantly check current system status.
+ detectHazard() : a method to detect a hazard.
+ changeMode() : Change a mode to the designated degradedLevel.

3.3.4 Prioritizer



Notifier
- messageQueue : a queue for saving messages to be notified.
- prioritizer : an instance of Prioritizer class.
+ Notify() : notify the message.
+ putMsg() : put message into the queue.
+ getMSg() : get message from the queue.
Prioritizer
- rules : a list of rules to give a proper priority to the message.
- getRules() : load rules from the rule XML file.
- Prioritize() : based on rules, give a proper priority to the message.

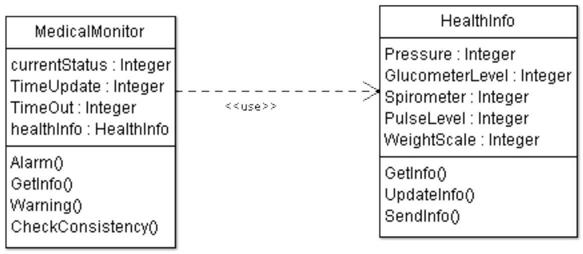
3.4 Scheduler



- Name : The contact's name

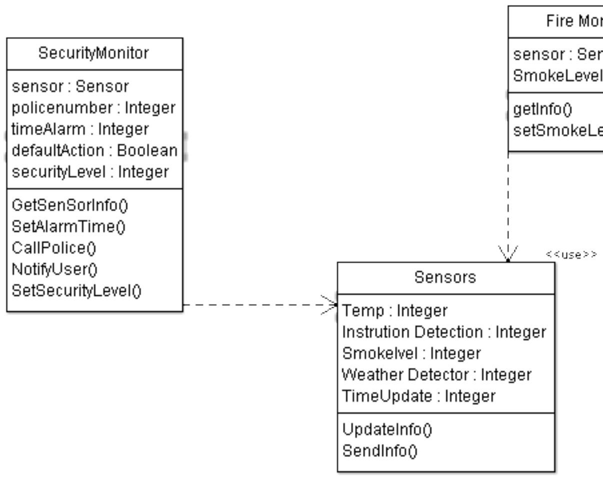
- date: the date for meeting
- place: the place of meeting
- time: time for meeting
- reason : why have to meet
- CreateNew(): Create new meeting
- Modify(): Modify scheduled meeting
- Search(): Search for a specific meeting
- UpdateData(): Update database

3.5 Medical Monitor



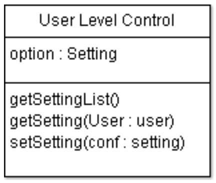
- currentStatus: stored current health status of user
- TimeUpdate: Specific the period of update info
- TimeOut: Specific the time out waiting user's response
- Alarm(): Alarm
- GetInfo() : get user's health status
- Warning(): Warning user
- CheckConsistency() : Check health status user based on information get from sensor
- Pressure : User's pressure
- GlucometerLevel: User's Glucometer
- Spirometer: User's spirometer
- Pulse Level: User's pulse level
- Weight Scale: User's weight

3.6 Security Monitor



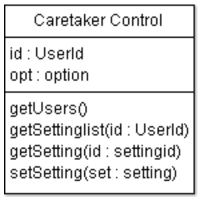
- policenumber: store police emergency number - default: 911
- timeAlarm: set a period of alarm
- defaultAction: true - carry on the protection steps for user and house.
- securityLevel: set specific level of protection
- SmokeLevel:
- Temp : house's temperature
- InstrutionDetection : Detect instrution
- SmokeLevel
- WeatherDetector : Check weather in that area
- TimeUpdate: Specific time of update info
- GetSensorInfo()
- SetAlarmTime()
- CallPolice()
- NotifyUser()
- SetSecurityLevel()
- SetSmokeLevel()

3.7 User Level Control



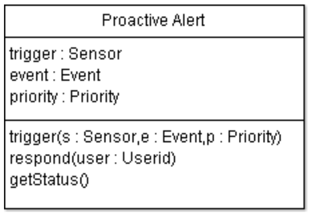
- option : an instance of the setting class.
+ getSettingList() : get list of available settings for modification.
+ getSetting(user: userid): get list of available settings for user
+ setSetting(conf:setting) : commit changes to database.

3.8 Caretaker Control


Caretaker Control
id : UserId
opt : option
getUsers()
getSettinglist(id : UserId)
getSetting(id : settingid)
setSetting(set : setting)

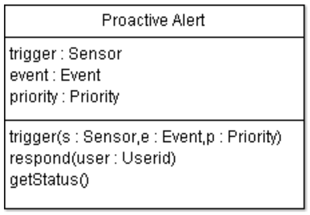
- id : an instance of Userid, this contains user identification.
option : an instance of the setting class.
+ getUsers(): get list of available users.
+ getSettingList() : get list of available settings for modification.
+ getSetting(user: userid): get list of available settings for user
+ setSetting(conf:setting) : commit changes to database.

3.9 Proactive Alert



- Trigger : which sensor triggered the notification.
- event: the event that the sensor is reporting.
- priority : how important the message is.
+ trigger(s, e, p) : called by the Active mode of the notification system informs us which sensor triggered, what the event was that caused the trigger, for example smoke sensor and what priority is the message.
+respond(userid): the user or caretaker responds to the notification
+getStatus(): this returns the status of the notification

3.10 Reactive Alert



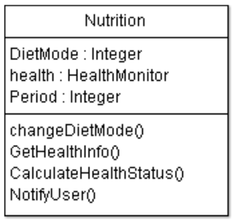
- event: event is a text stream in the ical file format

+trigger(): this function is called by the calender system with the ical notification statement

+respond(userid): the user or caretaker responds to the notification

+getStatus(): this returns the status of the notification

3.11 Nutrition Monitor/Planner



- DietMode: Diet modes for user
- Period: Time for current diet mode effective
- changeDietMode(): change to another diet mode
- GetHealthInfo() : Get user's health status
- CalculateHealthStatus
- NotifyUser() : warning user if anything inconsistency.

3.12 Performance Requirements

This sections specify performance requirements for calendering and notification sub systems:

- UI Transition:

  The information transfers between any devices (sensors/actuators) with main system should not more than 3s.

- Data access time:

  The system should access any data from database in reasonable time.

- Start-up Time:

  The time between system is reset and normally operated should be less than 10s.

- Interoperability:

  The system shall work smoothly with other smart home systems.

3.13 Software System Attributes

3.13.1 Reliability/Dependability

The list below describes the requirement of reliability for the calendering and notification sub-systems

- Mean-Time-Between-Failure:

  The failure rate of two sub-system must be below 1 times/month.

- Fault-Tolerance:

  Sub-system must have a backup copy to continue operation in case the primary system fails.

- Display data accuracy:

  The information displayed to users via user interface must be correct and prompt.

- User setting accuracy:

  The information about user's preferences of system must be consistent, reliable and up-to-date.

- Log accuracy:

  The log of everyday operation should be updated by the end of day and backed up on the weekend.

- Operation accuracy:

  Two subsystem must control hardware equipments like sensors and actuators precisely and safely.

3.13.2 Security

The list below describes the system requirement about security:

- Information confidentiality:

  - All information about users should be kept secret by strict information protection policy,
  - All information is classified according to clearance level and user's privilege.
  - Never store plain password, and auto-remember user password.
  - Restrict the number of reentering password simultaneously.
  - Using some password protect hardware device like dongle.
  - Implement User Access Control mechanism.

- Information Integrity:

  - The information can be only modified/deleted by users with specific clearance level.
  - The system is able to detect many unauthorized access, counter password-guessing technique, defend from outside attacks.

  - The system shall always synchronize data between prime-copy and back-up copy in every short period.

- Information Availability:
  - The system must provide information to right user with enough privilege in timely manner.
  - Always have a backup plan in every weak/month.

3.13.3 Availability

The list below describes availability requirement:

- The system shall provide the requested service with 24/7avalability.
- The response time of the system when a request arrive should be prompt and precise.

3.13.4 Maintainability

The list below describes maintainability requirement:

- The system shall have a backup system to be upgraded parallel/on-line when a new device comes or some modification taken by technician.
  - The display device can be removed when an user still can issues command
  - A new sensor can be added to the system without any conflict in communication and transmission.
  - Every sensor and actuator can be configured and controller via GUI.
- The system will introduce bugs with very low probability when updating changes.
  - The audit team can review source code and run code in simulation devices.
  - The system is able to update new code with any disruption.

3.13.5 Repair-ability:

The list below describes repair-ability requirement:

- The system shall have a quick repair down time.
- The system shall be able to be diagnose and replace malfunctioning parts while still running.

3.14 Design Constraints

The list below describes about design constrains in some aspects:

- Coding Constraint:

  Two sub systems shall be developed using Java language.
- Memory Constraint:

  The memory for all two subsystem shall not be larger than 1Gb.

- Line of code constraint:

  The total number of LOC should less than 100K.

- Functionality constraint:

  Two sub system shall provide ONLY functions required in the requirement

- Environment constraint:

  sub systems shall be developed under Windows NT OS.

- The interface between components shall be consistent and well described.

3.15 Other Requirements

NONE

4 Appendix A

4.1 Motivation:

One of important aspects in our NCS subsystemss is checking the feasibility of our models. We decide to choose simulation to verify our models and our subsystems. The main reasons to choose simulation are its lightweight, easy to use and able to use as a storyboarding tool to explicit customers' requirements. There are two level of simulation we carry on in our project:

1. Using simulation at UML model levels to check consistency and safety.

2. Using simulation at subsytem levels to check integerity and perfomance of the whole subsystems.

Our approach can be catergoried as Simulation-based Model Checking. Using simulation in model driven development is currently a hot topic which has been discuessed in many top SE conference like ICSE, OOPSLA, MODELS, TOOLS, etc. There are also alot of tools supporting for model developing such as Microsoft DSL Tools, Eclipse EMF/GMF/oAW, MetaEdit+, Rhapsody, etc.

In this project, we choose Rhapsody as our tools to simulate and check UML models. Rhapsody has many interesting properties that are suitable for our subsystems:

- Seamless Environment for Systems and Software Development Requirements

- Modeling Design-level Debugging on Target

- Directly Deployable C, C++, and Ada Code Generation

- Automatic Test Vector Generation

In short, Rhapsody animates the model by executing the code generated with instrumentation for classes, operations, and associations. Animated Sequence Diagram, State Machine Diagram, and Activity Diagram help design-level debugging. E.g., step through the model, set and clear breakpoints, inject events, and generate an output trace.

For simulation at the subsystem level (i.e NCS system), we choose BDI models. The purpose of BDI models is to characterize agents using anthropomorphic notions, such as mental states and actions. formally defined using logical frameworks that allow theorists to analyze, specify and verify rational agents [6]. The reason choosing BDI models based on four following properties:

- Well-known and studied model of practical reasoning agents.

- BDI model combines a respectable philosophical model of human practical reasoning.

- A number of implementations, several successful applications the now-famous fault diagnosis system for the space shuttle, as well as factory process control systems and business process management

- An elegant abstract logical semantics, which have been taken up and elaborated upon widely within the agent research community.

However, BDI models are only thoery model base. We need an language to describe this model and a tool to run it in practice. That why we choose AgentSpeak language and Jason tool to implement BDI model.

So far, in this project we choose Rhapsody to verify UML models via simulation and AgentSpeak with Jason tool to implement BDI model that are capable of simulating our NCS subsystems.

4.2 BDI Logic

Goal

- Surveying BDI logic and its language and tools.
- Investigating the way of service planning for agents in a smart home.
- Improve an evolvable context-aware system, which adapt to environmental changes (home environment, user intentions) in real time manner.

This project will investigate the way of service planning for multiple agents in a smart home. In this system, we introduce BDI logic to make a service invocation plan by synthesizing home electronics.

Introduction of BDI Bratman [1] proposed Belief, Desire, Intention (BDI) model based on belief, desire and intention as mental states. BDI models of agents have been widely researched by AI researchers. The purpose of these models is to characterize agents using anthropomorphic notions, such as mental states and actions. formally defined using logical frameworks that allow theorists to analyze, specify and verify rational agents [6].

What is BDI[2]?

Beliefs: Beliefs represent the characteristics of the environment. Beliefs are updated appropriately after each sensing action. Beliefs can be viewed as the *informative* component of the system.

Desires: Desires contain the information about the objectives to be accomplished, the priorities and payoffs associated with the various objectives. Desires can be thought as representing the *motivational* state of the system.

Intentions: Intentions represent the currently chosen course of action (the output of the most recent call to the selection function). Intentions capture the *deliberative* component of the system.

Why BDI?

- Best known and best studied model of practical reasoning agents.
- BDI model combines a respectable philosophical model of human practical reasoning.
- A number of implementations, several successful applications the now-famous fault diagnosis system for the space shuttle, as well as factory process control systems and business process management
- An elegant abstract logical semantics, which have been taken up and elaborated upon widely within the agent research community.

AgentSpeak(L)

- Attempt to bridge the gap between theory and practice
- A model that shows a one-to-one correspondence between the model theory, proof theory and the abstract interpreter.

- Natural extension of logic programming for the BDI agent architecture
- Based on a restricted first-order language with events and actions.
- The behavior of the agent is dictated by the programs written in AgentSpeak(L).

Jason[2]

- a fully-fledged interpreter for AgentSpeak(L)
- many extensions, providing a very expressive programming language for agents.
- allows configuration of a multi-agent system to run on various hosts.
- implemented in Java (thus it is multi-platform)
- available Open Source and is distributed under GNU LGPL.
- http://jason.sourceforge.net/

Our Example [Our own contribution]

Agent Script

```
/* Plans */
+time(T) : smokeTime(ST) & (T > ST + 5) <- !alarm(smoke).
+time(T) : ovenTime(ST) & (T > ST + 10) <- !alarm(oven).
+time(T) : mvTime(ST) & (T > ST + 13) <- !alarm(mv).
+time(T) : alarmTime(R,ST) & (T > ST + 7) <- !callFireman(R).

//oven & MV
+oven(on) : time(T) & not ovenTime(Q) <- +ovenTime(T).
+mv(on) : time(T) & not mvTime(Q) <- +mvTime(T).

//smoke
+smoke : time(T) & not smokeTime(Q) <- +smokeTime(T).

//alarm for smoke
+!alarm(R) : loc(home) <- alarmenv; .print("ALARM!!!!! ", R); !saveAlarm(R).
+!alarm(R) : loc(outside) <- !callFireman(R).
+!alarm(R).
+!saveAlarm(R) : time(T) & not alarmTime(Q) <- +alarmTime(R,T).

//heat
+heat <- !callFireman(R).

//call
+!callFireman(R): true <- .print("call because of ", R); callenv; .send(fireman,tell,fire).
```

4.3 Rhapsody

What Rhapsody can do?

- A environment for Systems and Software Development
- Flexible design environments supporting SysML, UML 2.0, DoDAF, and Domain Specific Languages (Rational Rose import and XMI support)
- Integrated Requirements modeling, traceability and analysis e.g., Lifecycle traceability and analysis
- Small and large team collaboration e.g., Model-based differencing and merging
- Design for Testability
  - Model simulation
  - Requirements Based testing
  - Auto test generation
  - Embedded target model level debugging
- Application generation
  - C, C++, Java, Ada
  - Code visualization and reverse engineering
  - Integration with Eclipse-based IDEs

Model Driven Development It starts out with making a requirements document and a series of models using UML, SysML or a similar modeling language. These models and requirements are then used directly for the generation of code. MDD tools use models as the primary means of development. One of the extremely useful tools it has is its *animator*. It allows you to view an *animated version of many diagrams* in real time as the program is running.

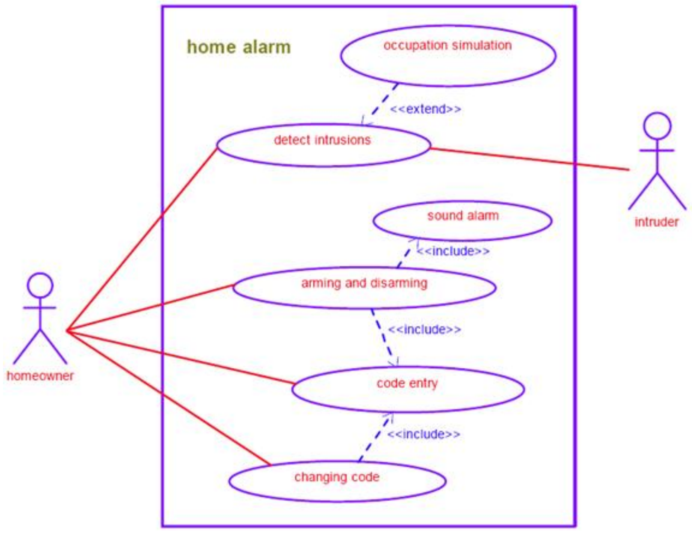
Use Animation to Validate the Model Rhapsody animates the model by executing the code generated with instrumentation for classes, operations, and associations. Animated Sequence Diagram, State Machine Diagram, and Activity Diagram help design-level debugging. E.g., step through the model, set and clear breakpoints, inject events, and generate an output trace.

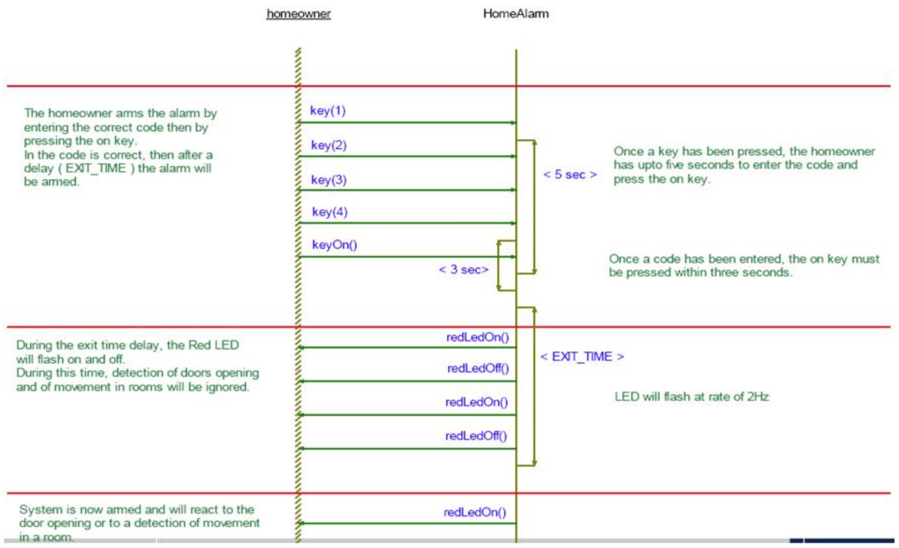
UML by Example: Home Alarm System[8] This example is from Rhapsody sample projects. It has a following functions.

- Door and movement sensors
- Arm/Disarm based on 4 digit code
- Code may be modified
- Timeout to allow entering/exiting house
- Indication lights

- Siren

Diagrams





The homeowner arms the alarm by entering the correct code then by pressing the on key.
In the code is correct, then after a delay (EXIT_TIME) the alarm will be armed.

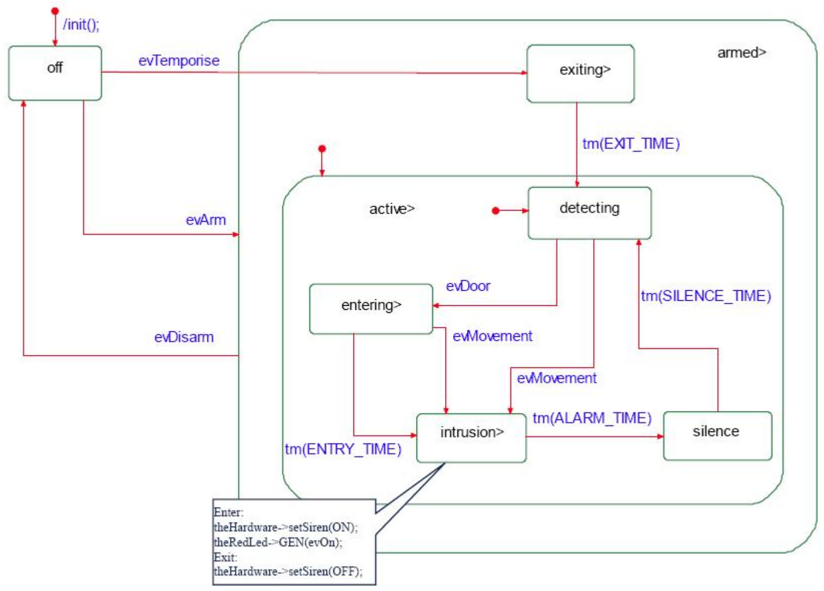
Once a key has been pressed, the homeowner has upto five seconds to enter the code and press the on key.

Once a code has been entered, the on key must be pressed within three seconds.

During the exit time delay, the Red LED will flash on and off.
During this time, detection of doors opening and of movement in rooms will be ignored.

LED will flash at rate of 2Hz

System is now armed and will react to the door opening or to a detection of movement in a room.



```
Enter:
theHardware->setSiren(ON);
theRedLed->GEN(evOn);
Exit:
theHardware->setSiren(OFF);
```

4.4 Conclusion:

At this moment, we do not fully develop the subsystems with two level of simulation: model level and system level. We expect the results would be:

- A comprehensive checking of model's consistency and safety via model simulation: The results will tell us which model is not conformed with requirements that we proposed and which model is non-safety model that can lead to hazards.
- A comprehensive checking of subsystem's intergrity and performance: It can describe in detail the time and space cost of each function/module and the problem in communication between module.

References

1. M. E. Bratman, “Intentions, Plans, and Practical Reason,” Harvard University Press, Cambridge, MA, 1987.

2. BDI Agents and AgentSpeak(L)(Romelia Plesa,PhD Candidate, University of Ottawa)

3. A BDI Agent-Based Software Process(Chang-Hyun Jo, California State University Fullerton, USA, Jeffery M. Einhorn, University of North Dakota, USA. AgentSpeak(L): BDI Agents speak out in a logical computable language (Anand S. Rao)

4. http://jason.sourceforge.net/mini-tutorial/getting-started/

5. P. R. Cohen and H. J. Levesque, “Intention is choice with commitment,” Artificial Intelligence , vol. 42, is. 2-3, pp.213-261, 1990.

6. Bordini, R. H., Hübner, J. F. and Vieira, R., “Jason and the Golden Fleece of agent-oriented programming.” In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, Multi-Agent Programming: Languages, Platforms and Applications, chapter 1, pp. 3–37,Springer-Verlag, 2005.

7. Rao, A. S. “AgentSpeak(L): BDI agents speak out in a logical computable language”, In Proceedings of the 7th European Workshop on Modelling Autonomous Agents in A Multi-Agent World (MAAMAW’96,), pp. 42-55, 1996

8. http://modeling.telelogic.com/products/rhapsody/index.cfm