**Networked Redistribution Grid**

**(NRG)**

**CS 437 Software Engineering**

**Software Requirements Document (SRD), Software Design Document (SDD), and Software Test Plan (STP)**

Prepared by:

William Klein

Michael Holloway

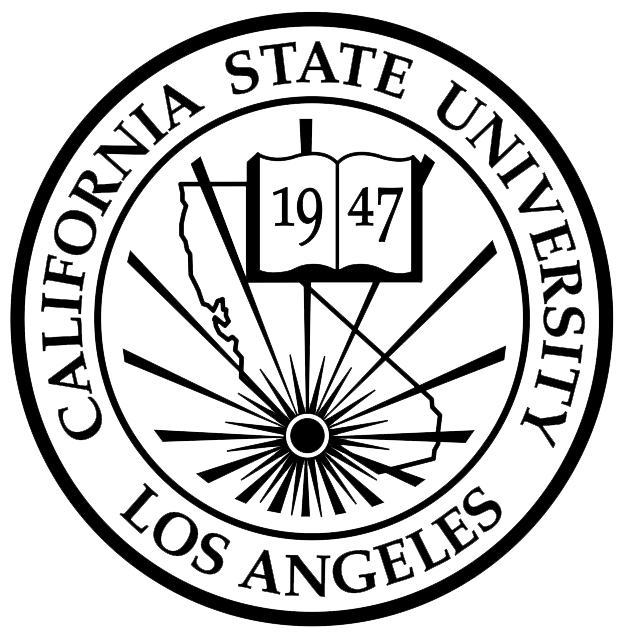
Tony Liang

Laura Mann

March 15, 2014

CALIFORNIA STATE UNIVERSITY

LOS ANGELES



Los Angeles, California

**Networked Redistribution Grid**

**(NRG)**

**Prepared By:**

William Klein

Michael Holloway

Tony Liang

Laura Mann

**Approved By:**

Jose Macias 01/15/2014

**\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_**

Professor Date

**Document Change Log**

|  |  |  |
| --- | --- | --- |
| **Update** | **Date Released** | **Changes** |
| Presentation | 03/6/2014 | Present project |
| Draft #1 | 03/15/2014 | Delivery of the Software Requirements document |

Table of Contents

[1.0 INTRODUCTION 7](#_Toc480470712)

[1.1 Purpose 7](#_Toc480470713)

[1.2 Scope 7](#_Toc480470714)

[1.2.1 Document Organization 7](#_Toc480470715)

[1.2.2 Relationship to Other Documents 7](#_Toc480470716)

[1.3 NRG Architecture 8](#_Toc480470717)

[1.3.1 Detailed Context Diagram (Data Flow Diagram Level 0) 8](#_Toc480470718)

[1.3.2 Description and Major Functions 8](#_Toc480470719)

[1.4 Documentation Development Process 9](#_Toc480470720)

[1.5 References 9](#_Toc480470721)

[1.5.1 Controlling Documents 9](#_Toc480470722)

[1.5.2 Applicable Documents 9](#_Toc480470723)

[1.5.3 Standards 9](#_Toc480470724)

[2.0 DETAILED FUNCTIONAL DESCRIPTION OF THE *NRG* 10](#_Toc480470725)

[2.1 Detailed NRG Functional Description 10](#_Toc480470726)

[2.1.1 Higher Level Data Flow Diagrams 10](#_Toc480470727)

[2.1.2 Detailed Functional Description of NRG’s Major Sub-Units 10](#_Toc480470728)

[3.0 *NRG* REQUIREMENTS 11](#_Toc480470729)

[3.1 NRG Functional Requirements 11](#_Toc480470730)

[3.2 Non-Functional Requirements 13](#_Toc480470731)

[3.3 Hardware Requirements 13](#_Toc480470732)

[4.0 NRGDETAILED DESIGN 13](#_Toc480470733)

[4.1 Introduction 13](#_Toc480470734)

[4.2 NRG External Interface Descriptions 13](#_Toc480470735)

[Module 1: Predictive Function 14](#_Toc480470736)

[Module 2: Response Function 14](#_Toc480470737)

[Module 3: Data Management Function 15](#_Toc480470738)

[Module 4: Control Interface Function 16](#_Toc480470739)

[5.0 NRG ELEMENTS OF IMPLEMENTATION 17](#_Toc480470740)

[Module 1: Predictive Function 17](#_Toc480470741)

[Test Case 6.1.1-1 17](#_Toc480470742)

[Test Case 6.1.1-2 17](#_Toc480470743)

[Test Case 6.1.1-3 17](#_Toc480470744)

[Module 2: Response Function 17](#_Toc480470745)

[Test Case 6.1.2-1 17](#_Toc480470746)

[Test Case 6.1.2-2 18](#_Toc480470747)

[Test Case 6.1.2-3 19](#_Toc480470748)

[Test Case 6.1.2-4 19](#_Toc480470749)

[Test Case 6.1.2-5 20](#_Toc480470750)

[Test Case 6.1.2-6 20](#_Toc480470751)

[Module 2.3: Data Management Function 21](#_Toc480470752)

[Test Case 6.1.3-1 21](#_Toc480470753)

[Test Case 6.1.3-2 21](#_Toc480470754)

[Test Case 6.1.3-3 22](#_Toc480470755)

[Module 2.4: Control Interface Function 22](#_Toc480470756)

[Test Case 6.1.4-1 22](#_Toc480470757)

[Test Case 6.1.4-2 22](#_Toc480470758)

[Test Case 6.1.4-3 22](#_Toc480470759)

[Test Case 6.1.4-4 22](#_Toc480470760)

[Test Case 6.1.4-5 23](#_Toc480470761)

[6.0 NRG TEST PLAN 23](#_Toc480470762)

[6.1 Introduction 23](#_Toc480470763)

[6.2 Functional Requirements Validation Matrix 23](#_Toc480470764)

[A. DATA DICTIONARY 24](#_Toc480470765)

[B. ACRONYMS 24](#_Toc480470766)

**List of Figures**

**DFD Level 0: page [i-8]**

**DFD Level 1: page [i-10]**

**Predictive Function Level 2 DFD: page [i-14]**

**Power Consumption Submodule Level 2 DFD: page [i-14]**

**Distribution Submodule Level 2 DFD: page [i-15]**

**Data Management Function Level 2 DFD: page [i-15]**

**Control Interface Function Level 2 DFD: page [i-16]**

**Control GUI Submodule Level 2 DFD: page [i-16]**

**List of Tables**

**Requirements Related to Module 2.1:  Predictive Function: page [i-11]**

**Requirements Related to Module 2.2:  Response Function: page [i-12]**

**Requirements Related to Module 2.3:  Data Management Function: page [i-12]**

**Requirements Related to Module 2.4:  Control Interface Function: page [i-12]**

**Predictive Function Validation Matrix: page [i-23]**

**Response Function Validation Matrix: page [i-23]**

**Data Management Function Validation Matrix: page [i-24]**

**Control Function Validation Matrix: page [i-24]**

# 1.0 INTRODUCTION

# 1.1 Purpose

The purpose of this document is **four-fold**:

1. Completely define a full set of requirements for the **NRG** – **Section 3.0**.

(These sections correspond to the Software Requirements Document (SRD)).

1. Completely define the design for the **NRG – Section 4.0**.

(These sections correspond to the Software Design Document (SDD)).

1. Define and (partially) implement feasible modules for the **NRG – Section 5.0**.

(These sections correspond to the Software Implementation Document (SID)).

1. Completely define the Test Plan for the **NRG – Section 6.0**.

(These sections correspond to the Software Test Plan, (STP)).

The complete definition of all NRGrequirements provides the source requirement inputs for the development of the subsequent supporting software subsystems documents.

# 1.2 Scope

The documentation developed as part of the CS337 class includes the NRG’s SRD and SDD (Sections 1.0 through 4.0). The remaining parts of the document (Sections 5.0 and 6.0) are left for this software engineering class, CS437.

The scope of this document includes the following:

* All functional and non-functional requirements on the **NRG** are captured. This includes Verification & Validation (V&V) requirements and inter-software subsystems requirements.
* A complete set of **NRG** Requirements, derived and traceable to the incoming CS437 class requirements.

These requirements are organized by key **NRG** functional units shown on the Level 1 Data Flow Diagram (DFD). The Level 1 DFD is shown on page [i-10].

* A trace matrix, relating all **NRG** functional requirements to functional subunits as expanded in lower level DFDs. Level 2 and 3 DFDs are provided on pages [i-14 to i-16].
* The functional requirements defined in the **NRG** Requirements section have been expanded to include more specific hardware requirements.

## 1.2.1 Document Organization

The organization of this document provides a natural 'flow' or allocation of requirements to each succeeding section.

Details regarding the overall document are given in sub-section 1.5 below.

## 1.2.2 Relationship to Other Documents

The **NRG** SRD/SDD/SID/STP is a complete self-contained document. Some relationships to other documents in the literature are indicated below in sub-section 1.5.

# 1.3 NRG Architecture

## 1.3.1 Detailed Context Diagram (Data Flow Diagram Level 0)

The **NRG** architecture is summarized in the Context Diagram (DFD Level 0) given below. A more complete Functional Description is given in Section 2.0 of this document.

NRG DFD Level 0 Diagram

Grid Data

NRG

Device Instructions

Predictive Function

Response Function

Data Management Function

Control Interface Function

Weather Data

Device Data

Interface Output

Interface Input

## 1.3.2 Description and Major Functions

***The primary objective of the NRG is to provide a system by which the daily tide of energy usage can be moderated to reduce dependence on inefficient power plants. The system aims to do this by anticipating demand and reducing the power used by devices that are non-essential for the uninterrupted flow of daily life. This would in turn reduce overall power generation costs and environmental stress.***

The NRG software system shall:

- Predict future grid usage based on historical data.

- Adjust power usage for non-essential devices.

- Maintain a database of relevant historical data.

- Provide device data to their owners.

- Allow a technician to take control of the system in an emergency.

The NRG system differs from other smart grid plans by taking a more active role in reducing power usage during peak hours. In other systems, the force behind any change in energy usage is ultimately the end user. This reliance on individual participation means that in order to see a significant result, a large number of people need to change their habits. This is unlikely. NRG avoids this by adjusting consumption rates automatically when needed without any input from the consumer.

***The next level description will be given in Section 2.***

# 1.4 Documentation Development Process

The **NRG** detailed functional description is documented in Section 2.0. Basically, Section 2.0 is a succinct software description for the document. The overall detailed functional description is based on higher level DFDs (above level 1). All major functional units are described in detail in this part of the document.

In general, all requirements affecting **NRG** are captured in Section 3.0. These requirements are a refinement and completion of requirements first collected as part of a Software Engineering project. The document is cited in Section 1.2.2. This section is the one worked in most detail to become a reasonably complete Software Requirements Document (SRD). It includes both functional and non-functional software requirements together with several detailed “rational” paragraphs whenever necessary to complete the understanding of each requirement.

Section 4 is the **NRG** detailed Design Description Document (SDD). This part of the document includes all higher level DFDs as described in Section 2.0 and all interface units. The document is highly technical and is based on Section 2.0 descriptions.

Section 5 includes elements of implementation of **NRG**. This section includes the various constraints that effectively limit the implementation as well as the sub-units that will be coded. The implementation goals are defined, and the code and pseudo code are included as an attachment to this section.

Section 6 is the last major section in this document and includes the overall Test Plan (TP) of the **NRG**. The test plan details the various techniques used to test the requirements and includes a Validation Matrix where each requirement specified in Section 3.0 is listed with its corresponding validation method. In addition, the TP specifies the mandated peer reviews needed to validate the stakeholders’ part of the requirements.

# 1.5 References

All references used in the creation of this document are listed below.

[http://smartgridcc.org/wp-content/uploads/2012/08/000000000001021126.pdf](http://smartgridcc.org/wp-content/uploads/2012/08/000000000001021126.pdf#_blank)

[http://www.ccst.us/publications/2011/2011smart-final.pdf](http://www.ccst.us/publications/2011/2011smart-final.pdf#_blank)

[http://www.youtube.com/watch?v=QqfMwEQ2pbc](http://www.youtube.com/watch?v=QqfMwEQ2pbc#_blank)

## 1.5.1 Controlling Documents

There is no document controlling this document.

## 1.5.2 Applicable Documents

Macias, Jose *aaFRD cs337 Draft.doc* “Our requirements template”

## 1.5.3 Standards

No Standard has been used in the creation of this document. However, some Standards described in textbooks have been examined as a reference. In particular, the IEEE standard has been briefly discussed in class.

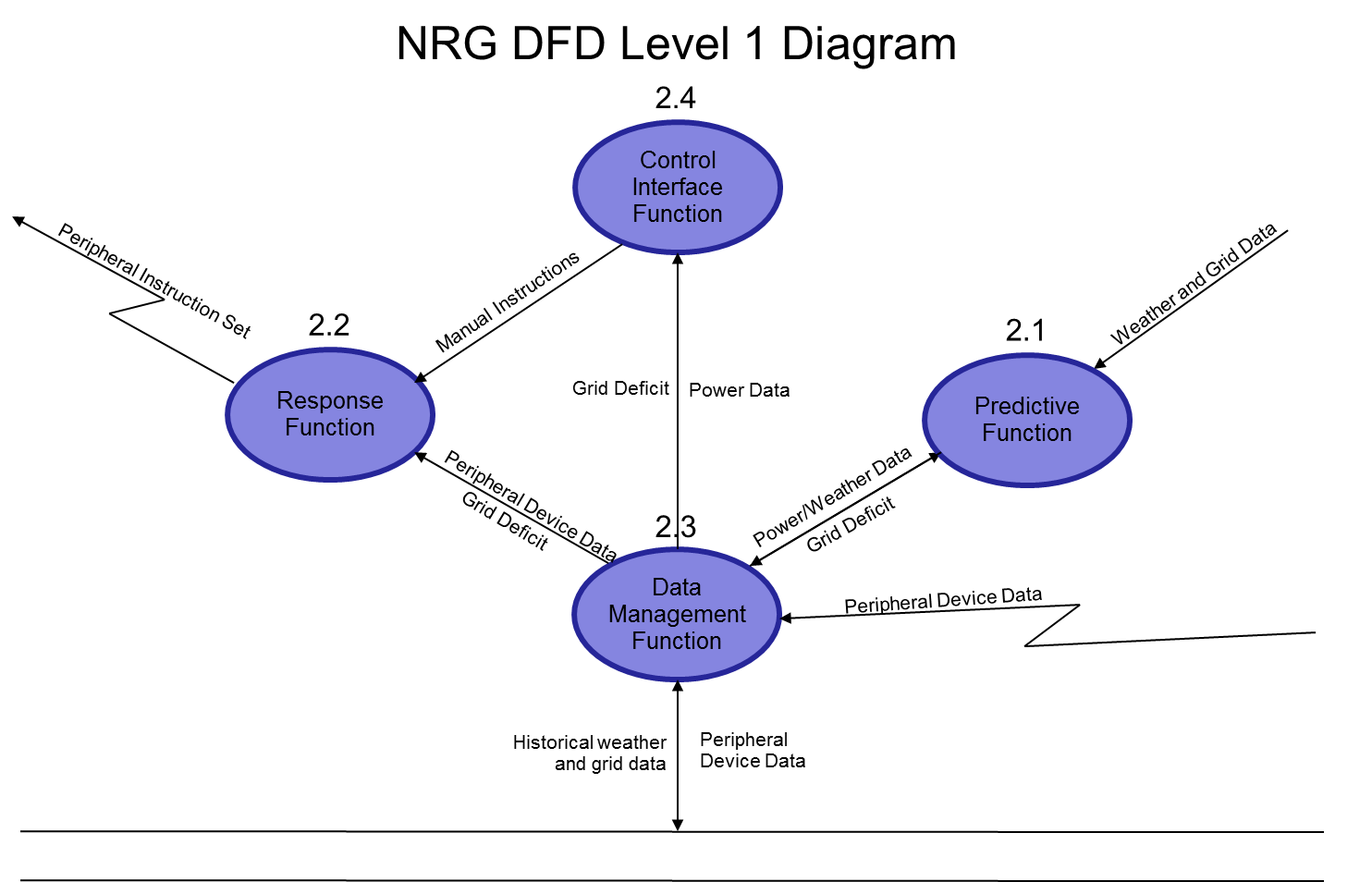
# 2.0 DETAILED FUNCTIONAL DESCRIPTION OF THE *NRG*

# 2.1 Detailed NRG Functional Description

The major tool used to design **NRG** is the DFD. The rationale behind the selection of DFDs as the preferred design tool is its simplicity and versatility. In the future, tools that are more sophisticated may be used particularly if a correlation from Design to Requirement to Implementation and Testing is found to be a necessary addition.

## 2.1.1 Higher Level Data Flow Diagrams

The **NRG** major functional design components are shown in the DFDs below.



## 2.1.2 Detailed Functional Description of NRG’s Major Sub-Units

The **NRG**’s major functional subunits shown in the DFDs in the previous sub-section are described in detail below.

2.1 Predictive Function (PF)

* The PF takes in a collection of data from two external sources: power data and weather data.
* Power data are information about power usage demand from power plants. The PF processes power data to find peak demand time from power plants that could cause stress on the power grid and creates data on predicted grid deficit.
* Weather data are current real-time data on weather forecast that affect the demand on power usage.
* The PF will take in weather data to adjust the predicted grid deficit to account for change in weather.

During hot weather climate, the demand on the power grid can exponentially increase.

* The PF calculates the information from the two external sources and generates the final prediction on grid deficit to be used in.

2.2 Response Function (RF)

* When the RF receives the device and grid deficit data from the Data Management Function, the RF will control the peripheral device's power consumption.
* To help get a fast emergency response, the RF constantly communicates with the Control Interface Function. When the RF receives the data, it analyzes the data and ranks non-essential appliances (e.g., driers, chargers, etc.) in order of importance.
* If more power needs to be limited, the RF adjusts the rankings of the devices and prioritizes based on the data it received. For example, if the weather is hot, a water heater is the first device to be limited of power consumption.
* Conversely, in cold weather, air conditioners are the first devices to be limited.
* Once this process is completed, the RF sends a wireless signal to the peripheral device's chip and lowers its power consumption as needed.
* When the CIF requests the RF to turn off some devices, the RF will automatically turn off power consumption of these devices.

2.3 Data Management Function (DMF)

* The DMF controls all database interaction within the system.
* It accesses and maintains tables containing historical weather data, power production data, grid deficit data, current peripheral device usage data, and individual device historical data.
* This centralized access point allows all database queries to be managed in one place and simplifies the task of data storage and retrieval.
* Additionally, this reduces the likelihood of concurrent queries that could result in errors.

2.4 Control Interface Function (CIF)

* The CIF is a built-in redundancy layer to allow direct human control over the system in case of errors or emergencies.
* Any system of this scope needs multiple safety systems.
* The ability of the PF and RF to assess the supply and demand of power to the grid accurately is necessarily limited by the ability of engineers to anticipate every possible situation and its consequences.
* Thus, by including a CIF, a person can interpret and respond to any extraordinary circumstances.

# 3.0 *NRG* REQUIREMENTS

# 3.1 NRG Functional Requirements

This Section collects all **NRG** Functional Requirements. The Section includes the complete set of functional requirements with explanation and rationale where the statement of the requirement was deemed insufficient or needing additional background/justification. All requirements relate to the design modules described in Section 2.0. An effort has been made to standardize the correlation between the design modules and the requirements to make their access and organization more consistent. For example, module 2.1 requirements are labeled 3.1, sub-module 2.1.1 requirements are labeled 3.1.1, and so on. The list of requirements follows.

|  |  |
| --- | --- |
| Requirements Related to Module 2.1: Predictive Function (PF) | |
| Requirement No. | Requirement Description |
| 3.1-1 | PF shall receive data on present grid demand and current weather conditions. |
| 3.1-1.1 | PF shall receive present grid data from the power grid. |
| 3.1-1.2 | PF shall receive present weather data from weather service. |
| 3.1-1.3 | PF shall send present grid and weather data to the database management function. |
| 3.1-2 | PF shall estimate grid deficits based on historical data and present conditions. |
| 3.1-2.1 | PF shall receive historical data on past weather and grid demand from the database management function. |
| 3.1-2.2 | PF shall overestimate demand by 1% to maintain a small supply buffer. |
| 3.1-2.3 | PF shall estimate grid and weather data if current data is unavailable. |

|  |  |
| --- | --- |
| Requirements Related to Module 2.2:  Response Function (RF) | |
| Requirement No. | Requirement Description |
| 3.2-1 | RF shall receive the device and grid deficit data from the Data Management Function. |
| 3.2-1.1 | RF shall analyze the device and grid deficit data it receives. |
| 3.2-2 | RF shall remove the current grid deficit by decreasing the power consumption of devices. |
| 3.2-2.1 | RF shall assign the majority of the deficit to lower priority devices. |
| 3.2-2.2 | RF shall send instructions to the control chips to reduce power consumption below supply levels. |
| 3.2-3 | RF shall keep as many devices on as long as possible. |
| 3.2-4 | RF shall send generalized instructions based on technician input to all devices on the system. |
| 3.2-5 | RF shall constantly reassess its response package based on the most recent predictions and commands from the Control Interface Function. |
| 3.2-5.1 | The response packages shall be based on what devices are currently in use. |
| 3.2-5.2 | The response packages shall have a list of all devices that are currently in use. |
| 3.2-5.3 | All responses shall include instructions for devices not currently being used to request a power use value when turned on. |
| 3.2-6 | RF shall constantly communicate with the Control Interface Function to receive new instructions. |
| 3.2-6.1 | RF shall automatically turn off devices that the Control Interface Function requested. |
| 3.2-6.2 | RF shall automatically turn on devices that the Control Interface Function requested. |

|  |  |
| --- | --- |
| Requirements Related to Module 2.3: Data Management Function (DMF) | |
| Requirement No. | Requirement Description |
| 3.3-1 | DMF shall accept usage data from peripheral device chips. |
| 3.3-2 | DMF shall have a database containing historical weather, grid supply, grid demand, and deficit prediction data. |
| 3.3-3 | The database shall be continuously updated with new weather and grid data as it is received. |
| 3.3-4 | DMF shall hold the current deficit prediction in memory. |
| 3.3-5 | DMF shall be able to modify single Device entries. |
| 3.3-6 | DMF shall create and maintain a connection to the database. |
| 3.3-7 | DMF shall organize data requested by other functions into lists of objects. |
| 3.3-8 | DMF shall have object classes corresponding to the entries in each database table. |
| 3.3-9 | DMF shall have the ability to delete data from any table created before a given time. |

|  |  |
| --- | --- |
| Requirements Related to Module 2.4: Control Interface Function (CIF) | |
| Requirement No. | Requirement Description |
| 3.4-1 | CIF shall provide an interface for a technician to assert direct control over the system. |
| 3.4-2 | CIF shall present current grid data to the technician through the interface. |
| 3.4-3 | The manual control interface shall not be accessible via the internet. |
| 3.4-4 | The interface shall allow technicians to set specific time frames for manual actions. |
| 3.4-4.1 | The default time frame shall be no more than 24 hours. |
| 3.4-5 | CIF shall provide a means for a technician to send instructions to the RF according by device type and priority. |
| 3.4-6 | CIF shall request information from the DMF. |
| 3.4-7 | CIF shall display real time graphical representations of weather and grid data. |
| 3.4-8 | CIF shall send and receive device instructions. |
| 3.4-9 | CIF shall provide a means to cut all power to NRG compliant devices. |
| 3.4-10 | CIF shall manually set a power limit for each priority group. |
| 3.4-11 | CIF shall allow a technician to remove all previous instruction sets. |
| 3.4-12 | CIF shall send an instruction set that automatically suspends the RF. |

# 3.2 Non-Functional Requirements

This section contains the NRG Non-Functional Requirements. Non-functional requirements are numbered “NF - n” where “n” indicates the nth requirement.

NF – 1 NRG shall have a database capable of storing citywide data on grid usage, weather patterns, and device data.

NF – 2 NRG shall require a Wi-Fi network for each device to connect.

# 3.3 Hardware Requirements

This section contains the NRG Hardware Requirements. Hardware requirements are numbered “H - n” where “n” indicates the nth requirement.

H – 1 A central control unit capable of managing grid usage for an entire city.

H – 2 Control chips embedded in each NRG compliant device on the grid.

H – 3 A control terminal for emergency control of NRG.

# 4.0 NRGDETAILED DESIGN

# 4.1 Introduction

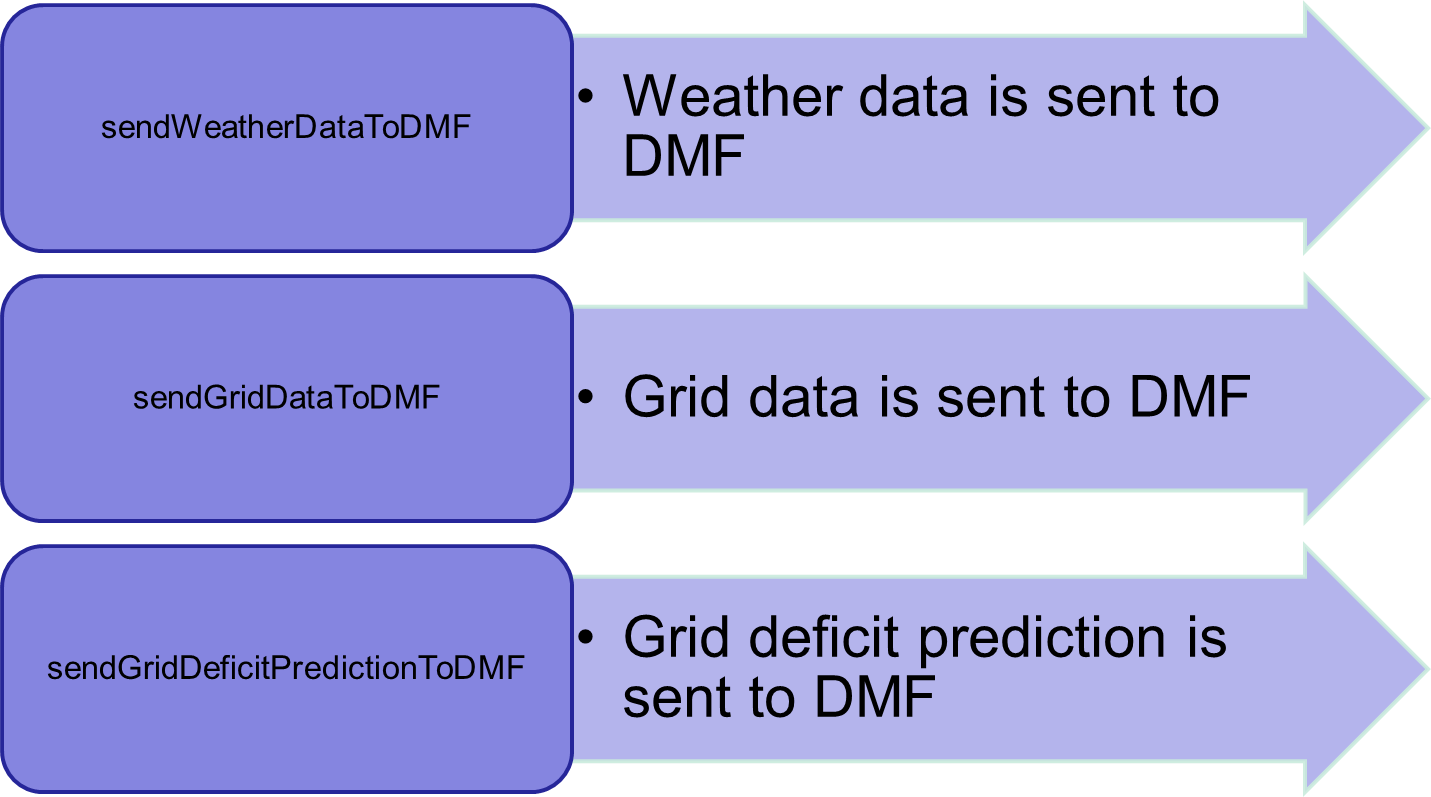
In this section, the **NRG** described in Section 2.0 with requirements listed in Section 3.0 shall be designed in detail including several higher level DFDs. Each major module detailed design is included in correspondence with the design sections defined in Section 2.0 and responding to the requirements listed in its correlated sub-section in Section 3.0.

# 4.2 NRG External Interface Descriptions

The detailed design of each of the four modules discussed in Section 2.0 with requirements presented in Section 3.0 is presented in the Figures below.

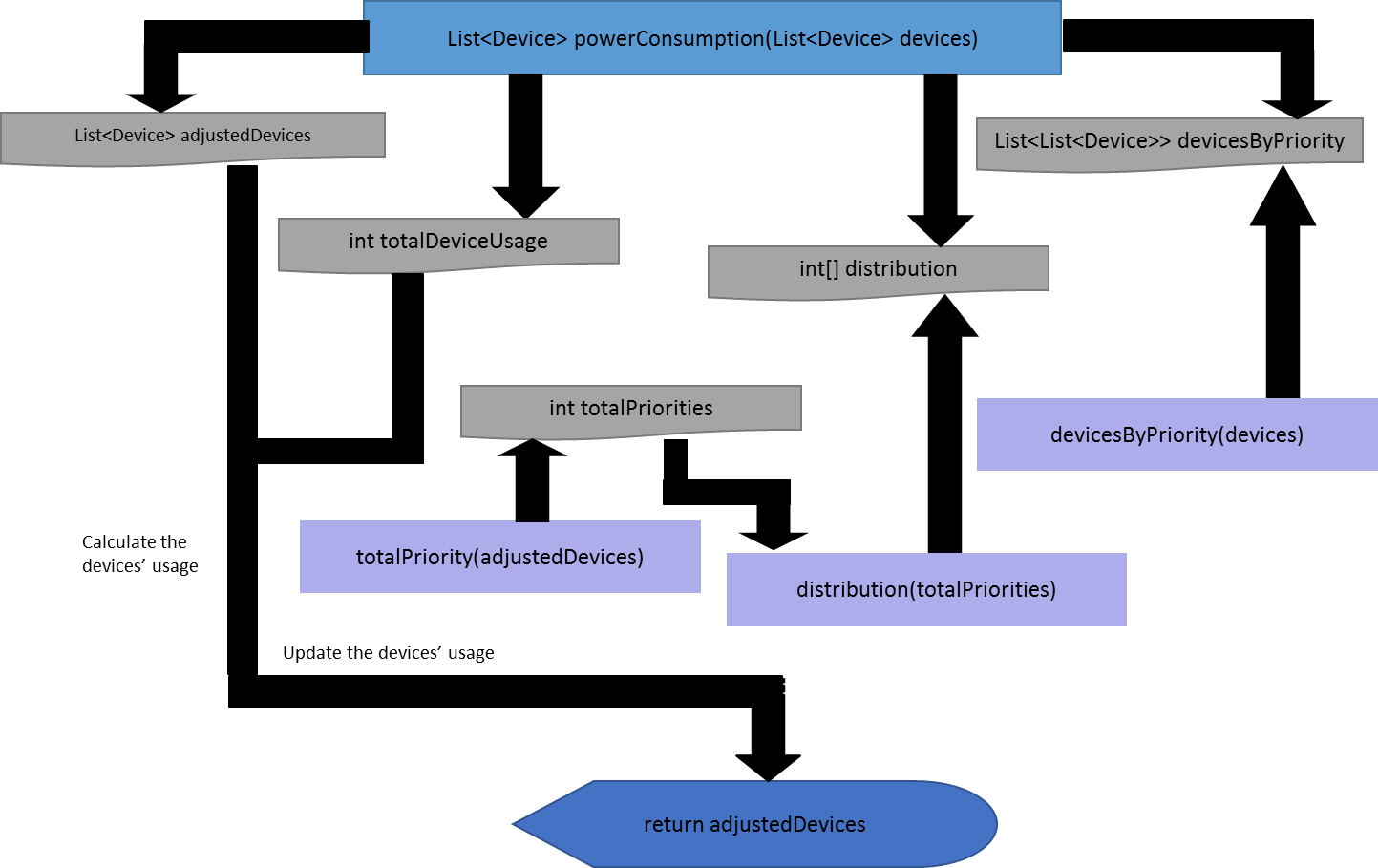
## Module 1: Predictive Function

Predictive Function Level 2 DFD

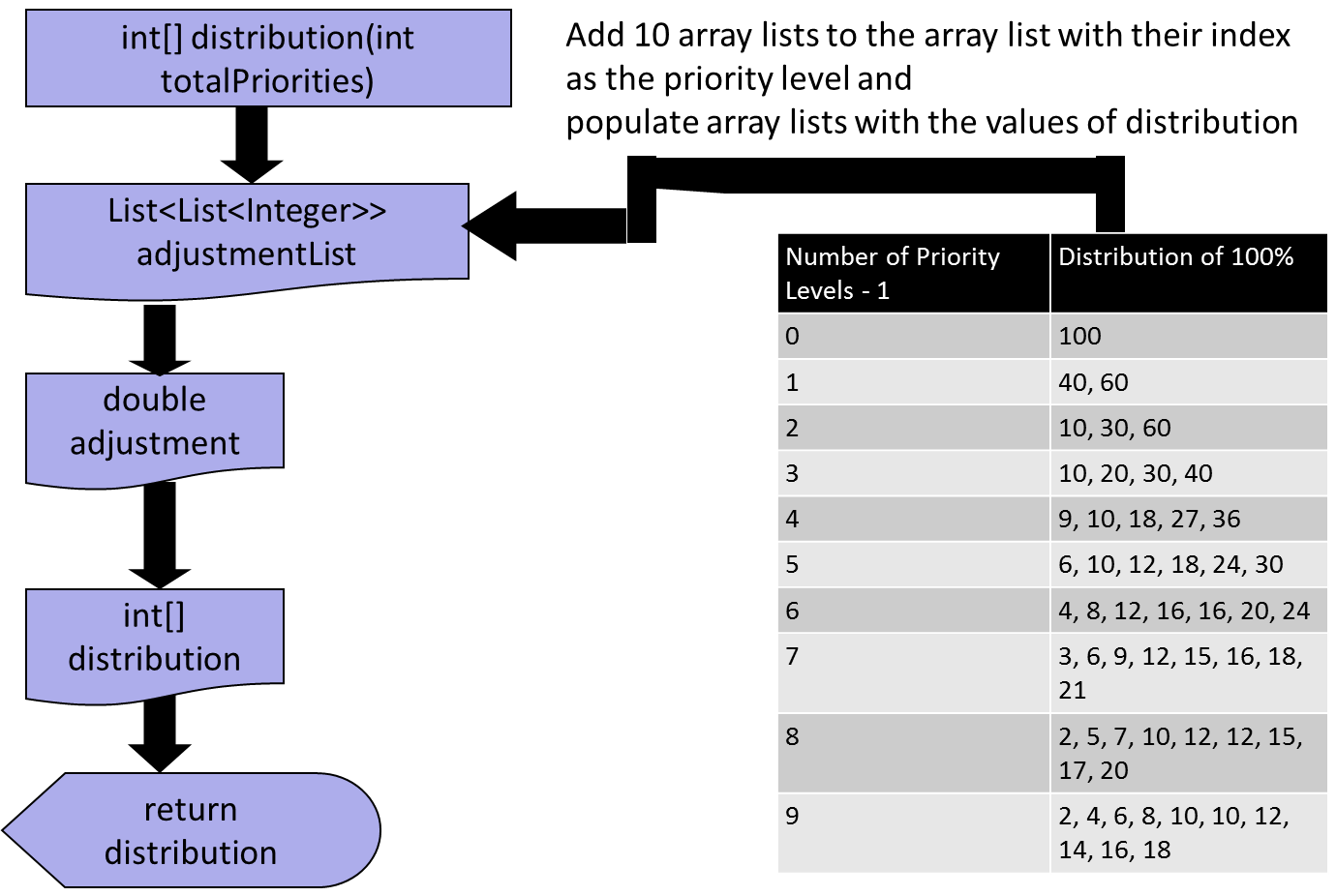


## Module 2: Response Function

Power Consumption Submodule Level 2 DFD

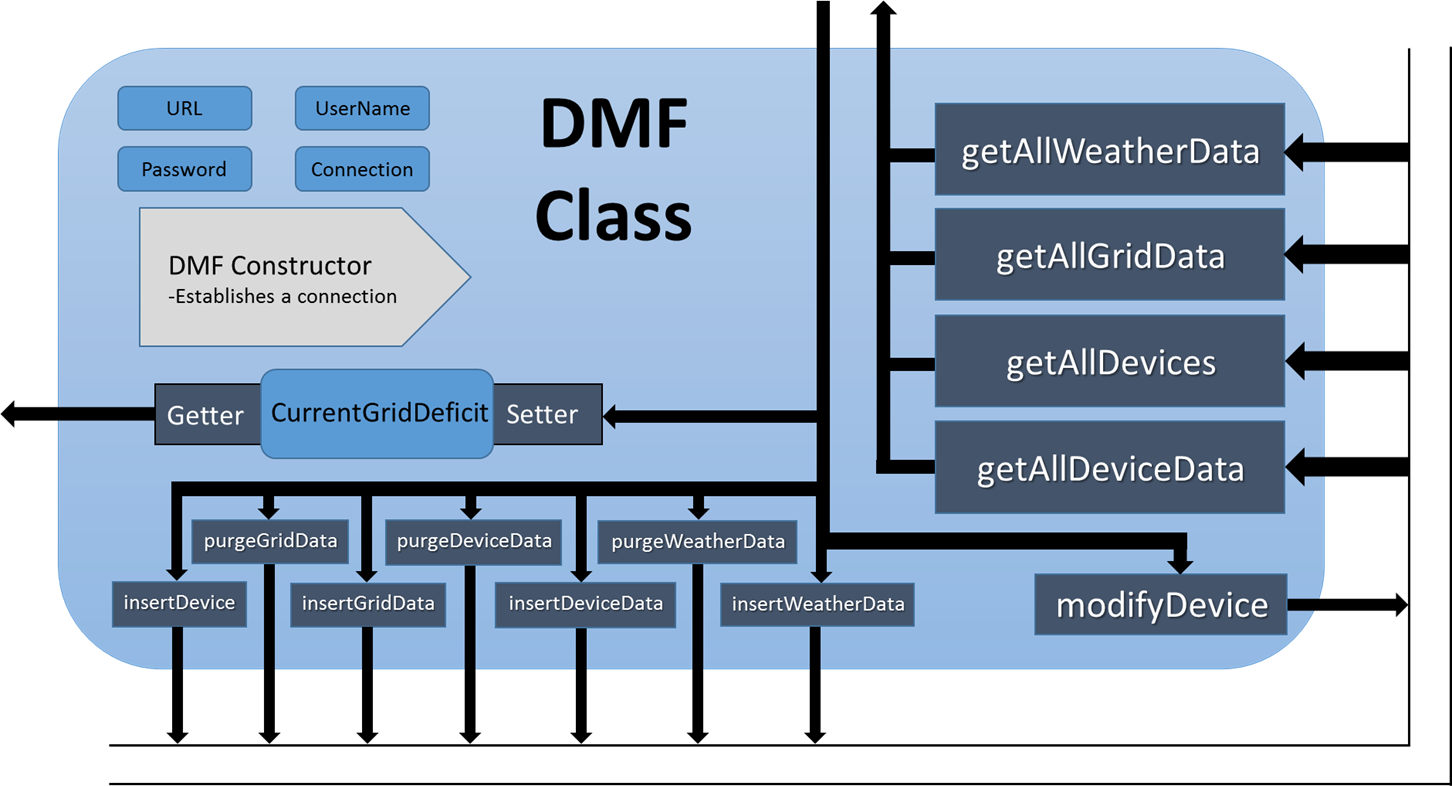


Distribution Submodule Level 2 DFD



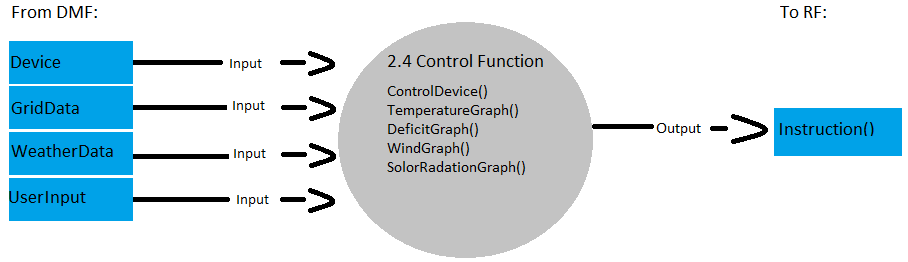
## Module 3: Data Management Function

Data Management Function Level 2 DFD

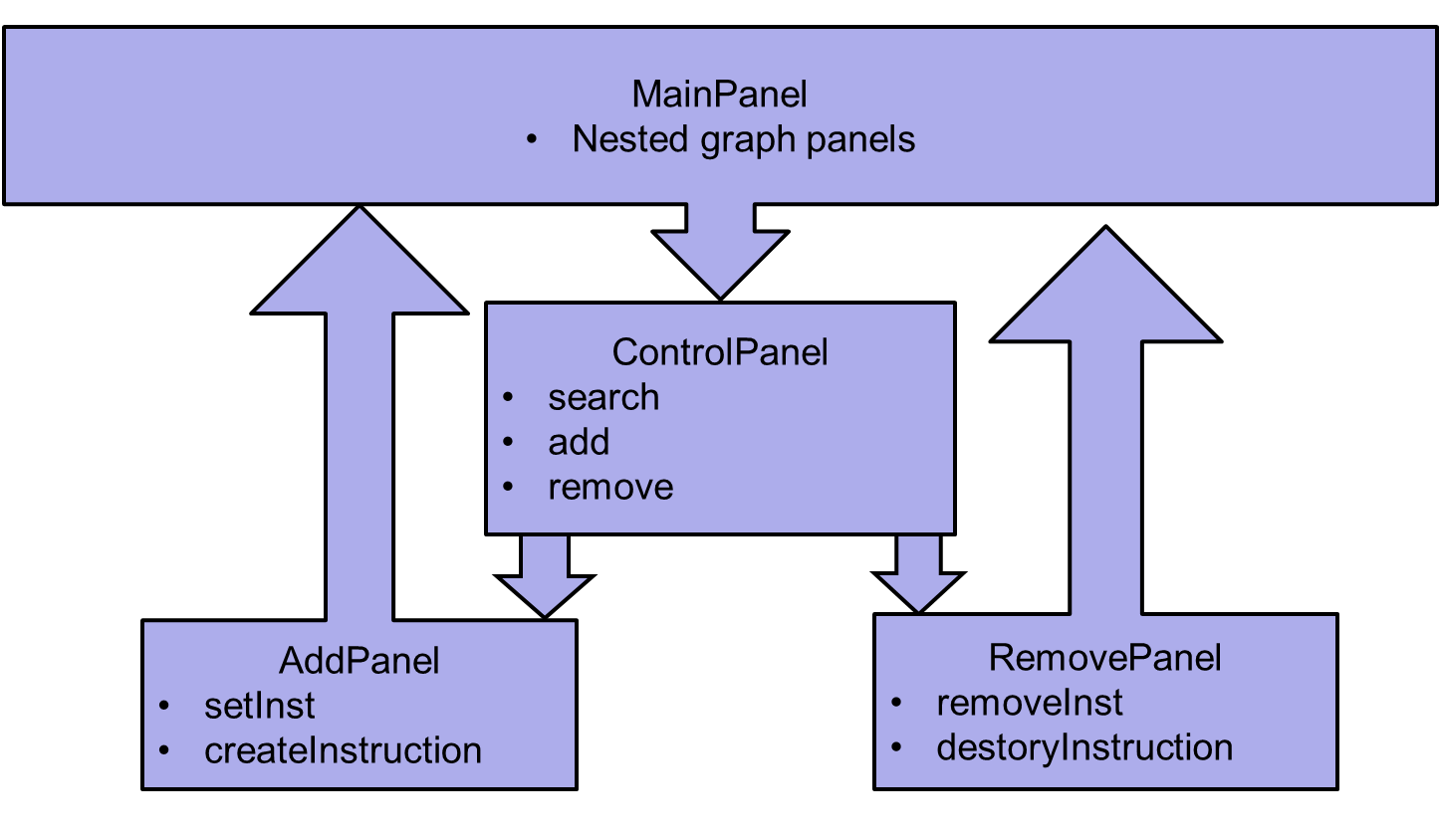


## Module 4: Control Interface Function

Control Interface Function Level 2 DFD



Control GUI Submodule Level 2 DFD



# 5.0 NRG ELEMENTS OF IMPLEMENTATION

In this section, (some of) the modules designed in Section 4.0 with requirements listed in Section 3.0 shall be implemented initially at the level of pseudo code. Actual code shall be provided as time permits. Each module is implemented in correspondence with the design sections defined in Section 2.0 and responding to the requirements listed in its correlated sub-section in Section 3.0.

\*\*\*Full source code is included alongside the SRD to retain organizational structure and prevent document bloat. Refer to these files for NRG implementation.

## Module 1: Predictive Function

### Test Case 6.1.1-1

Requirement(s): 3.1-1.1; 3.1-1.3

* Pseudo-Code:
  + Function [input: grid data from grid]:[no output]
    - Grid data 🡪 Database Management Function

### Test Case 6.1.1-2

Requirement(s): 3.1-1.2; 3.1-1.3

* Pseudo-Code:
  + Function [input: weather data from weather service]:[no output]
    - Weather data 🡪 Database Management Function

### Test Case 6.1.1-3

Requirement(s): 3.1-2.1; 3.1-2.2; 3.1-2.3

* Pseudo-Code:
  + Function [no inputs]:[no output]
    - Past grid data 🡨 Database Management Function
    - Past weather data 🡨 Database Management Function
    - Mapped data 🡨 MapDataByTimeStamp(past Grid data, past Weather data)
    - pastYears 🡨 GetPastYearsData(mapped data)
    - prediction 🡨 determinePrediction(pastYears)
    - prediction 🡨 prediction \* 1.01 // Add 1% demand buffer
    - prediction 🡪 Database Management Function

## Module 2: Response Function

### Test Case 6.1.2-1

Requirements Tested: 3.2-2.1

The distribution function has a list of ten lists of percentages that corresponds to the total number of priority levels that has at least one device associated with them. The current grid deficit is split up based on these percentages.

// Distributes power percent usage appropriately

int[] distribution(int totalPriorities)

{

// Arbitrary initial device usage

List<List<Integer>> adjustmentList -> new ArrayList<List<Integer>>()

FOR every integer from 0 to 9

List<Integer> list -> new ArrayList<Integer>()

Adds the list to adjustmentList

ENDFOR

Add 100 to the first list of the adjustmentList

Add 40 and 60 to the second list of the adjustmentList

Add 10, 30, and 60 to the third list of the adjustmentList

Add 10, 20, 30, and 40 to the fourth list of the adjustmentList

Add 9, 10, 18, 27, and 36 to the fifth list of the adjustmentList

Add 6, 10, 12, 18, 24, and 30 to the sixth list of the adjustmentList

Add 4, 8, 12, 16, 16, 20, and 24 to the seventh list of the adjustmentList

Add 3, 6, 9, 12, 15, 16, 18, and 21 to the eighth list of the adjustmentList

Add 2, 5, 7, 10, 12, 12, 15, 17, and 20 to the ninth list of the adjustmentList

Add 2, 4, 6, 8, 10, 10, 12, 14, 16, and 18 to the tenth list of the adjustmentList

double adjustment -> adjustmentList.get(totalPriorities - 1).get(0)

int[] distribution -> new int[totalPriorities]

int j -> 1

FOR every value in the distribution array

distribution[i] -> (int) adjustment

IF j >= the length of distribution THEN break ENDIF

adjustment -> adjustmentList.get(totalPriorities - 1).get(j)

INCREMENT j By 1

ENDFOR

return distribution

}

### Test Case 6.1.2-2

Requirements Tested: 3.2-1, 3.2-1.1, 3.2-2, 3.2-3

The powerConsumption function receives the device data and current grid deficit. It uses the percentage values received from the distribution function to split the current grid deficit. Then, it would remove the deficit by decreasing the power usage of all devices. To keep as many devices on as possible, the function adjusts the parts of the current grid deficit to be removed accordingly. When the current grid deficit is removed, the devices’ power usages are updated in the DMF and instructions are sent to the control chips.

// Rank devices by priority. Modify devices' usage so that the current grid deficit becomes 0.

protected List<Device> powerConsumption(List<Device> devices)

{

List<Device> adjustedDevices -> devices

First, find the sum of all device usage

// If the current grid deficit requires all devices to shutdown

IF currentGridDeficit == totalDeviceUsage THEN

Sets all devices' usage to 0

return adjustedDevices

ENDIF

// Gets the total number of priority levels with at least one device

int totalPriorities = totalPriority(adjustedDevices)

// Calculates the current grid deficit percentage for each priority level

int[] distribution -> distribution(totalPriorities)

// All devices group by priority level

List<List<Device>> devicesByPriority -> devicesByPriority(devices)

Limits the usage of as many devices as needed to decrease the grid deficit to 0

Updates the device usages in the database

return adjustedDevices

}

### Test Case 6.1.2-3

Requirements Tested: 3.2-2.2

The sentWirelessSignal function sends a wireless signal to the control chip of a device with its power usage value.

// Sends wireless signal

void sentWirelessSignal(Device device)

{

int deviceUsage -> device.getDeviceUsage()

// Sends the device usage value to the device

}

### Test Case 6.1.2-4

Requirements Tested: 3.2-6.1, 3.2-6.2

The turnOn function turns on a list of devices and set the power use values each device requested.

// Turns on devices

void turnOn(List<Device> devices, int[] deviceUsage)

{

FOR every device

Sets its usage to the current value in the deviceUsage integer array

TRY

Updates the device in the database using DMF

IF there are errors THEN

Write the errors to the console

ENDIF

ENDFOR

}

The turnOff function turns off a list of devices.

// Turns off devices

void turnOff(List<Device> devices)

{

FOR every device

Set the current device’s usage to 0

TRY

Update the device in the database using DMF

IF there are errors THEN

Write the errors to the console

ENDIF

ENDFOR

}

### Test Case 6.1.2-5

Requirements Tested: 3.2-5.1, 3.2-5.2

The ResponsePackage class contains a list of all devices and their power usages in percentages.

class ResponsePackage

{

List<Device> devices

int[] devicesPercentages

ResponsePackage(List<Device> devices, int[] devicesPercentages)

{

this.devices -> devices

this.devicesPercentages -> devicesPercentages

}

// Communication with Control Interface Function

void sentToCIF()

{

// Sends the response package to the Control Interface Function

}

List<Device> getDevices()

{

return devices

}

int[] getDevicesPercentages()

{

return devicesPercentages

}

}

### Test Case 6.1.2-6

Requirements Tested: 3.2-5.3, 3.2-6

The responsePackage function creates a response package for assessment.

// Receives response packages from devices

ResponsePackage responsePackage(List<Device> devices, int[] devicesPercentages)

{

ResponsePackage rp -> new ResponsePackage(devices, devicesPercentages)

return rp

}

The receiveInstruction function constantly receives instructions from the CIF.

// Receives instructions from the Control Interface Function

void receiveInstruction(Instruction instructions)

{

IF an instruction required devices to turn on THEN

Gets the list of devices from the Instruction object and all the values each device demands

Puts the values into an integer array of size = the total devices that needs to be turn on

turnOn(devices, deviceUsage)

ENDIF

IF an instruction required devices to turn off THEN

Gets the list of devices from the Instruction object that needs to be turn off

turnOff(devices)

ENDIF

}

## Module 2.3: Data Management Function

### Test Case 6.1.3-1

Requirements: 3.3-2, 3.3-3, 3.3-4, 3.3-6, 3.3-7, 3.3-8

insertWeatherData( testWeatherData )

insertGridData( testGridData )

insertDevice( testDevice )

for each weatherData in getAllWeatherData

print ( weatherData to String )

for each gridData in getAllGridData

print ( gridData to String )

for each device in getAllDevices

print ( device to String )

### Test Case 6.1.3-2

Requirements: 3.3-5

insertDevice( testDevice )

for each device in getAllDevices

print ( device to String )

modifyDevice( newTestDevice )

for each device in getAllDevices

print ( device to String )

### Test Case 6.1.3-3

Requirements: 3.3-9

Populate the database with a large amount of dummy data using the attached SQL script.

purgeOldWeatherData( time )

for each weatherData in getAllWeatherData

print ( weatherData to String )

purgeOldGridData( time )

for each gridData in getAllGridData

print ( gridData to String )

## Module 2.4: Control Interface Function

### Test Case 6.1.4-1

Requirement(s): 3.4-1, 3.4-2, 3.4-3

GUI -> manual testing of GUI

### Test Case 6.1.4-2

Requirement(s): 3.4-4, 3.4-4.1

JComboBox timeFrame [input: number of hours for instruction to last selected by hour limited to 24 hours]:[no output] :

String setDueDate(timeframe /\*selected value\*/) /\*Returns a timestamp as a recognizable format for the RF\*/-> ControlDevice.setDue(); // sets it as a part of a new ControlDevice

### Test Case 6.1.4-3

Requirement(s): 3.4-5, 3.4-10

Function [input: List of Devices, type, switch, due chosen in the UI ]:[output: ControlDevice] :

JRadioButton RB\_TYPE, RB\_Priority -> String groupBy /\*Identifies the ControlDevice group as an instruction by type or by priority\*/->

JComboBox timeframe;-> String setDueDate(String timeframe)-> String timeStamp /\*select the length of the\*/->

JTextField SearchT, SearchP;-> setDeviceList(SearchT // by type or SearchP by int overloaded method) -> List <Device> includedDevices // returns list of Devices to control->

ControlDevice newGroup(includedDevices, switch // auto set to false to turn off device, timeStamp, groupBy); /\* creates a new object with these properties that can be added to until it’s sent to the RF by the user \*/

### Test Case 6.1.4-4

Requirement(s): 3.4-6, 3.4-7

Function [input: Grid and Weather data from the DMF]:[output: Graphical displays of data, real time data for user to create instructions for the RF] :

Grid and Weather data -> DMF -> Temperature, SolarRadiation, Wind and Deficit Panels -> attached to GUI

Grid and Weather data-> DMF -> ControlDevices /\*Data is grabbed from the DMF analyzed and processed by the UI of the CF to create a list of ControlDevices that can be sent as an Instruction upon completion to the RF.\*/

### Test Case 6.1.4-5

Requirement(s): 3.4-8, 3.4-9, 3.4-11, 3.4-12

Function [input: List of device Control objects that are sent to the RF]:[output: Instruction] :

List controlDevices(/\*List of control device objects by priority and type from user input in GUI interface\*/)

Instruction () newInstructionSet; // creates a new instruction

newInstruction.destroy(); // destroys all previous instructions

newInstruction.create(List ControlDevices); /\*send all new list of controlDevice object instructions to the RF function\*/

# 6.0 NRG TEST PLAN

# 6.1 Introduction

In this section, the testing methodology to be used to V&V each of the requirements listed in Section 3.0 has been identified. At points, some additional testing may be required and they shall be documented as an attachment to this document.

The methodologies and testing strategies identified at this point include three major approaches with various variations to adapt to the **NRG** project:

* Testing using additional ad-hoc created software including a correlation-testing unit.
* Demonstration of the specified capability.
* Inspection of the software code possibly using additional inspection techniques.

# 6.2 Functional Requirements Validation Matrix

The **NRG** Functional and Performance Requirements Validation Matrix are given below.

|  |  |
| --- | --- |
| Requirements Validation Matrix Related to Module 1: Predictive Function (PF) | |
| Test Case No. | Validated Requirement(s) |
| 6.1.1-1 | 3.1-1.1, 3.1-1.3 |
| 6.1.1-2 | 3.1-1.2, 3.1-1.3 |
| 6.1.1-3 | 3.1-2.1, 3.1-2.2, 3.1-2.3 |

|  |  |
| --- | --- |
| Requirements Validation Matrix Related to Module 2: Response Function (RF) | |
| Test Case No. | Validated Requirement(s) |
| 6.1.2-1 | 3.2-2.1 |
| 6.1.2-2 | 3.2-1, 3.2-1.1, 3.2-2, 3.2-3 |
| 6.1.2-3 | 3.2-2.2 |
| 6.1.2-4 | 3.2-6.1, 3.2-6.2 |
| 6.1.2-5 | 3.2-5.1, 3.2-5.2 |
| 6.1.2-6 | 3.2-5.3, 3.2-6 |

|  |  |
| --- | --- |
| Requirements Validation Matrix Related to Module 3: Data Management Function (DMF) | |
| Test Case No. | Validated Requirement(s) |
| 6.1.3-1 | 3.3-2, 3.3-3, 3.3-4, 3.3-6, 3.3-7, 3.3-8 |
| 6.1.3-2 | 3.3-5 |
| 6.1.3-3 | 3.3-9 |

|  |  |
| --- | --- |
| Requirements Validation Matrix Related to Module 4: Control Interface Function (CIF) | |
| Test Case No. | Validated Requirement(s) |
| 6.1.4-1 | 3.4-1, 3.4-2, 3.4-3 |
| 6.1.4-2 | 3.4-4, 3.4-4.1 |
| 6.1.4-3 | 3.4-5, 3.4-10 |
| 6.1.4-4 | 3.4-6, 3.4-7 |
| 6.1.4-5 | 3.4-8, 3.4-9, 3.4-11, 3.4-12 |

# A. DATA DICTIONARY

**Device Data** – Data received from devices embedded with NRG Wi-Fi remote chip.

**Historical Data** – An accumulative collection of past grid usage data that include weather pattern and daily consumption used to predict future energy deficit.

**Peripheral Device** – Any devices embedded with NRG Wi-Fi remote chip.

**Power Data** – Information about power usage demand from power plant.

**Weather Data** – Current real-time data on weather forecast that affect the demand on power usage.

# B. ACRONYMS

**NRG -** Networked Redistribution Grid

**SRD -** Software Requirement Document

**SDD -** Software Design Document

**SID -** Software Implementation Document

**STP -** Software Test Plan

**PF -** Predictive Function

**DMF -** Data Management Function

**CIF -** Control Interface Function

**RF -** Response Function

**DFD -** Data Flow Diagram

**TP -** Test Plan

**V&V -** Verification & Validation

**TBD -** To Be Determined