Summary

[1 Introduction 2](#_Toc128557589)

[1.1 Case study 2](#_Toc128557590)

[1.1.1 Context 2](#_Toc128557591)

[1.1.2 Requirements 2](#_Toc128557592)

[2 Use case model 4](#_Toc128557593)

[2.1 UC.1– Start cooking 4](#_Toc128557594)

[2.2 UC.2- Update Target temperature 4](#_Toc128557595)

[2.3 UC.3 - Check cooker temperature 5](#_Toc128557596)

# Introduction

This document proposes the use case model of the “Rise Cooker” COSMIC Case study v. 2.0.1 (see <https://cosmic-sizing.org/wp-content/uploads/2018/08/Rice-Cooker-2.0.1-GoogleDocs.pdf>).

## Case study

This section describes the context and functional requirements in sections 1.1, 1.2 and 1.3 of the case study. The measurement analysis can be found on the COSMIC website at the following URL <https://cosmic-sizing.org/wp-content/uploads/2018/08/Rice-Cooker-2.0.1-GoogleDocs.pdf>.

### Context

This case study illustrates the application of the COSMIC measurement method (version 4.0.1) to measure the functional size of the software embedded within the prototype of a simple Rice Cooker.

This is the first Rice Cooker to be built by this manufacturing company: a decision was made to develop the product by using successive prototypes. The case study presented here corresponds to the first prototype to be developed plus some other possible functions specified for a second prototype. The lessons learned in manufacturing the prototypes will help make better informed decisions on whether or not some of the product functional requirements should be re-allocated to software rather than to the hardware.

For this first simple prototype of a Rice Cooker, the system engineer specified:

* which functions will be implemented through hardware devices (section 1.2);
* the software-hardware interactions (section 1.3); and
* which functions will be implemented through software

### Requirements

#### Hardware functions

A power switch provides electricity to all hardware devices. This must be turned on by the Rice

Cooker operator so that the cooker can work.

The Cooker is fitted with the following hardware devices that may send data to or receive data from the software directly. An exception is the Cooking Mode button which puts the cooking mode into a random access memory (‘RAM’) that is accessible by the software.

1. The Cooking Mode button is a set of three inter-connected buttons that allows an operator to set one of 3 cooking modes: slow, normal or fast.
2. When one of the Cooking Mode buttons is pressed by the operator, the selected cooking mode is put into a RAM where it may be accessed by the software.
3. Once the Start button has been pressed, the hardware will not allow the cooking mode to be changed.
4. The Start button.

When the operator presses the Start button it:

1. sends a ‘start’ signal to the software;
2. starts a hardware Timer that provides the sole time reference for the software. If the operator presses the Start button without having set the cooking mode:
3. the hardware sets the cooking mode to the default value of ‘normal’ in the RAM.
4. The Timer emits three types of signals to the software.
5. At 1 second intervals, the timer emits a signal conveying the value of the elapsed time ‘t’ (i.e. at t = 0, t = 1, t = 2, t = 3, ….; the elapsed time values are 0,1,2,3…. respectively);
6. Every 30 seconds the timer emits a signal for the software to update the target cooker temperature. (This signal is first emitted at t = 0 and after every subsequent interval of 30 seconds);
7. Every 5 seconds the timer emits a signal for the software to check the actual cooker temperature. (This signal is first emitted at t = 5 and after every subsequent interval of 5 seconds).

At time t = 0, the signals are emitted in a priority sequence b), a). In the cases where three types of signals must be emitted at the same time, they are emitted in a priority sequence b), a), c).

1. The Cooking Lamp is turned ON by a command signal received from the software at startup.
2. The Read-only Memory (‘ROM’) is used to store the 'target temperature / mode / elapsed time' data, which is accessible to the software. This table of values is used by the software to determine the target temperature depending on the elapsed time and the cooking mode.
3. The Temperature Sensor measures and makes the temperature available to the software when requested by the software.
4. The Heater is controlled (ON or OFF) by a signal received from the software.
5. The Stop button, when pressed at any time, will:
6. stop the Timer (if the timer is re-started the elapsed time signal will start at t =0);
7. cut off power to all devices. Safety interactions between the buttons, the power supply and the Rice Cooker door are controlled by hardware in this prototype of the Rice Cooker and can be ignored.

#### Software – Hardware interactions

The software must:

1. accept a signal from the Start button;
2. accept 3 distinct types of signals from the Timer (the signals at 5 and 30 second intervals and the elapsed time after a 30 second signal);
3. get the cooking mode from the RAM;
4. get the 'target temperature / mode / elapsed time' data from the ROM;
5. get the current temperature from the Temperature Sensor;
6. send a ‘Turn ON’ command to the Cooking Lamp;
7. send a ‘Turn ON’ or ‘Turn OFF’ command to the Heater.
8. store the current target temperature in the RAM, and retrieve (or ‘get’) it from the RAM

# Use case model

## UC.1– Start cooking

|  |  |  |
| --- | --- | --- |
| **ID**  *UC.1* | | *Start cooking* |
| **Description** | | This procedure allows to turn on the Ris Cooker. |
| **Primary actor** | | **Start Button** |
| **Supporting actors** | | **Cooking Lamp, Heater** |
| **Entry Condition** | | The Start Button is turned off |
| **Exit condition**  On success | | The Rise Cooker starts working |
| **Exit condition**  On failure | | N/A |
| **Priority** | | High |
| **Extension points** | | N/A |
| **Generalization of** | | N/A |
| **MAIN SCENARIO** | | |
| **1** | **Start Button** | The Start Button sends the Start signal to the system. |
| **2** | **System** | The System responds sending the ‘Turn ON’ signal to the Heater. |
| **3** | **System** | The System sends the 'Turn ON' signal to the Cooking Lamp. |

## UC.2- Update Target temperature

|  |  |  |
| --- | --- | --- |
| **ID**  *UC.2* | | *Update Target temperature* |
| **Description** | | This procedure allows to update the target temperature of the Rise Cooker. |
| **Primary actor** | | **Timer** |
| **Supporting actors** | | **N/A** |
| **Entry Condition** | | The Start Button is turned off |
| **Exit condition**  On success | | The new temperature is set correctly. |
| **Exit condition**  On failure | | N/A |
| **Priority** | | High |
| **Extension points** | | N/A |
| **Generalization of** | | N/A |
| **MAIN SCENARIO** | | |
| **1** | **Timer** | The timer ticks after 30 seconds. |
| **2** | **Timer** | The timer receives the current elapsed time signal. |
| **3** | **System** | The system gets the cooking mode related to the Current Settings. |
| **4** | **System** | The system gets the new target temperature related to the Target settings. |
| **5** | **System** | The system stores the new target temperature related to the Target settings. |

## UC.3 - Check cooker temperature

|  |  |  |
| --- | --- | --- |
| **ID**  *UC.3* | | *Check cooker temperature* |
| **Description** | | This procedure allows to check the cooker temperature and turn on or off the heater. |
| **Primary actor** | | **Timer, Temperature Sensor** |
| **Supporting actors** | | **Heater** |
| **Entry Condition** | | The timer ticks every 5 seconds. |
| **Exit condition**  On success | | The heater is turned on or off. |
| **Exit condition**  On failure | | N/A |
| **Priority** | | High |
| **Extension points** | | N/A |
| **Generalization of** | | N/A |
| **MAIN SCENARIO** | | |
| **1** | **Timer** | The timer ticks after 5 seconds. |
| **2** | **System** | The system gets the current target temperature. |
| **3** | **Temperature Sensor** | The temperature Sensor sends the actual temperature to the system. |
| **4** | **System** | The system decides if the heater must be turned on or off. |
| **5** | **Heater** | The system requests the Heater to process the command. |