Курсовая работа

по дисциплине "Проектирование информационных систем"

**на тему: "Модель программного обеспечения холодильника"**

**Содержание**

Введение

1. Постановка задачи

2. Глоссарий

3. Создание модели вариантов использования

4. Анализ вариантов использования

5. Проектирование системы

5.1 Иерархия системы

5.2 Построение диаграммы классов (Class Diagram)

5.3 Построение диаграммы состояний (Statechart Diagram)

6. Реализация системы

6.1 Диаграммы компонентов (Component Diagram)

6.2 Диаграмма размещения

6.3 Генерация кода

Заключение

Список использованных источников

Введение

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем (ИС), создаваемых в различных областях экономики. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями:

* сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;
* наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);
* отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;
* необходимость интеграции существующих и вновь разрабатываемых приложений;
* функционирование в неоднородной среде на нескольких аппаратных платформах;
* разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
* существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков, и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

Перечисленные факторы способствовали появлению программно-технологических средств специального класса - CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Термин CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

1. Постановка задачи

**информационный система программный обеспечение холодильник**

Требуется разработать средствами Rational Rose модель программного обеспечения встроенного процессора холодильника. В холодильной камере имеется регулятор температуры, мотор, термометр, индикатор, таймер, датчик открытия двери камеры и устройство для подачи звуковых сигналов.

При помощи терморегулятора устанавливается максимально допустимая температура в данной камере. Мотор предназначен для поддержания низкой температуры. Термометр постоянно измеряет температуру внутри камеры, а индикатор температуры, расположенный на дверце, постоянно высвечивает ее значение. При повышении температуры выше предела, определяемого текущим положением регулятора, включается мотор. При снижении температуры ниже некоторого другого значения, связанного с первым, мотор отключается.

Доступ в камеру осуществляется через дверцу. Если дверь холодильной камеры открыта в течение слишком долгого времени, подается звуковой сигнал. Звуковой сигнал также подается в любых нештатных ситуациях (например, при поломке мотора). Холодильник ведет электронный журнал, в котором отмечаются все происходящие события:

* изменение положения терморегулятора камеры;
* включение и отключение мотора;
* доступ в камеру;
* внештатные ситуации.

Вариантом задания предусмотрена разработка схемы базы данных для хранения журнала событий холодильника. Содержимое журнала может быть передано в компьютер, подсоединенный к специальному гнезду на корпусе холодильника.

2. Глоссарий

Таблица

|  |  |
| --- | --- |
| **Регулятор температуры Холодильника** (Regulator T2) | С его помощью пользователь устанавливает максимально допустимую температуру в камере холодильника |
| **Мотор Холодильника** (Motor) | Предназначен для поддержания низкой температуры   * При снижении температуры ниже некоторого другого значения, связанного с первым, мотор отключается * При повышении температуры выше предела, определяемого текущим положением регулятора, включается мотор |
| **Термометр Холодильника** (Thermometer) | Постоянно измеряет температуру внутри камеры |
| **Таймер Холодильника** (Timer) | * Начинает отсчёт при открытии дверей * Отключается при закрытии |
| **Индикатор Холодильника** (Indicator) | Постоянно высвечивает значение температуры измеряемой термометром |
| **Устройство для подачи звуковых сигналов** (Sound Device) | Если дверь холодильника камеры открыта в течении долгого времени, подается звуковой сигнал |

3. Создание модели вариантов использования

*Действующие лица:*

Пользователь холодильника (User) – человек использующийся холодильником.

*Варианты использования:*

Исходя из потребностей действующих лиц, выделяются следующие варианты использования:

* Посмотреть температуру на индикаторе (Look T on indicator)
* Закрыть дверцу холодильника (Close door of the refrigerator)
* Открыть дверцу холодильника (Open door of the refrigerator)
* Отрегулировать температуру регулятором (Assign T by regulator)

**Построение диаграммы вариантов использования (Use Case Diagram)**

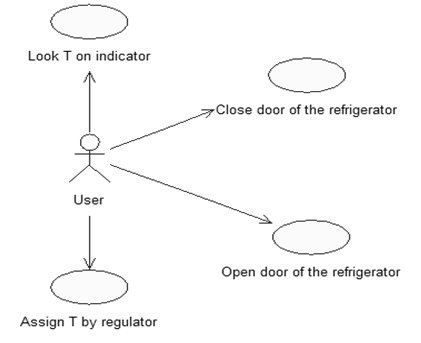


Рисунок 1 - Диаграмма вариантов использования

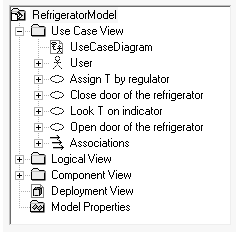


Рисунок 2 - Иерархия диаграммы вариантов использования

**4. Анализ вариантов использования**

**Открыть дверцу холодильника(Open door of the refrigerator)**

*Краткое описание*

Данный вариант использования описывает открытие дверцы холодильника пользователем.

*Основной поток событий*

Данный вариант использования описывает открытие дверцы холодильника пользователем.

В журнал заноситься информация об открытии дверцы.

*Альтернативный поток*

1. Возможны поломки
2. В журнал заноситься информация о поломке

*Предусловия*

Отсутствуют.

*Постусловия*

Включение таймера.

**Закрыть дверцу холодильника(Close door of the refrigerator):**

*Краткое описание*

Данный вариант использования описывает закрытие дверцы холодильника пользователем.

*Основной поток событий*

Данный вариант использования описывает закрытие дверцы холодильника пользователем.

В журнал заноситься информация об закрытие дверцы.

*Альтернативный потток*

1. Возможны поломки
2. В журнал заноситься информация о поломке

*Предусловия*

Возможно включенный звуковой сигнал.

*Постусловия*

1. Выключение таймера
2. Отключение звукового сигнала, если он сработал

**Отрегулировать температуру регулятором(Assign T by regulator)**

*Краткое описание*

Данный вариант использования описывает настройку пользователем температуры камеры холодильника регулятором.

*Основной поток событий*

Данный вариант использования описывает настройку пользователем температуры камеры холодильника регулятором.

1. Пользователь задаёт температуру
2. Происходит сравнение текущей температуры на термометре с заданным значением. Если текущая больше заданной включается мотор
3. В журнал заноситься информация о включении мотора

*Альтернативный поток*

1. Если текущая температура меньше или равна заданной, мотор не включается
2. Также если мотор работал, а заданная температура больше текущей в камере – мотор отключается
3. В журнал поступает информация о отключении мотора
4. Возможна поломка
5. В журнал поступает информация о поломке

*Предусловия*

1. Текущая температура термометра
2. Текущая температура заданная регулятором

*Постусловия*

1. Измененная температура
2. Возможно включение/выключение мотора

**Посмотреть температуру на индикаторе(Look T on indicator)**

*Краткое описание*

Данный вариант использования описывает просмотр температуры на индикаторе холодильника пользователем.

*Основной поток событий*

1. Пользователь узнает текущую температуру холодильника

*Альтернативный поток*

1. Возможны поломки
2. В журнал заноситься информация о поломке

*Предусловия*

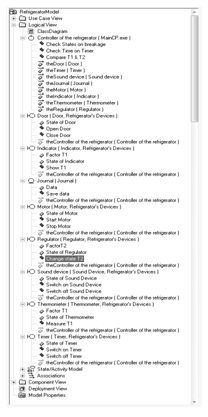
Термометр измеряет текущую температуру и выводит на индикаторе.

*Постусловия*

Пользователь узнал температуру.

**5. Проектирование системы**

**5.1 Иерархия системы**



**Описание классов**

*Граничные классы (Boundary):*

1. Дверца холодильника (Door)
2. Таймер (Timer)
3. Звуковое устройство (Sound Device)
4. Регулятор (Regulator)
5. Термометр (Thermometer)
6. Индикатор (Indicator)
7. Мотор (Motor)

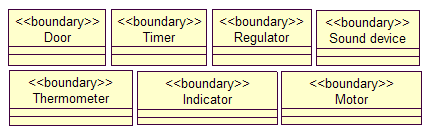


Рисунок 3 - Граничные классы

*Классы сущности (Entity):*

Журнал (Journal)

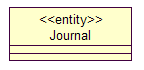


Рисунок 4 - Класс сущности Журнал (Journal)

*Управляющие классы (Control):*

Контроллер холодильника (Controller of refrigerator)

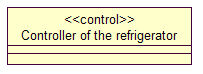


Рисунок 5 - Управляющие класс контроллер (Controller)

**Добавление операций и атрибутов к классам**

1. Дверца холодильника (Door)

*Операции:*

Close Door() : Boolean – Операция закрытия дверцы холодильника.

Open Door() : Boolean – Операция открытия дверцы холодильника.

При выполнении операций они возвращают 1 - если операция выполнилась без ошибок, 0 – если произошла поломка. Значение сохраняется в состоянии дверцы (State of Door).

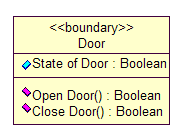


Рисунок 6 - Класс Door <<boundary>> с операциями и аргументами

1. Таймер (Timer)

*Операции:*

Switch on Timer() : Boolean – Операция включения таймера.

Switch off Timer() : Boolean – Операция выключения таймера (Сброса значения на 0).

При выполнении операций они возвращают 1 - если операция выполнилась без ошибок, 0 – если произошла поломка. Значение сохраняется в состоянии таймера (State of Timer).

*Атрибуты:*

State of Timer : Boolean - Состояние таймера (Рабочее/Не рабочее).

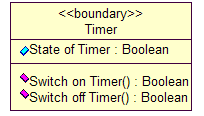


Рисунок 7 - Класс Timer <<boundary>> с операциями и аргументами

1. Звуковое устройство (Sound Device)

*Операции:*

Switch on Sound Device() : Boolean – Операция включения звукового сигнала при условии, что таймер отсчитал 30 секунд.

Switch off Sound Device() : Boolean – Операция выключения звукового сигнала.

При выполнении операций они возвращают 1 - если операция выполнилась без ошибок, 0 – если произошла поломка. Значение сохраняется в состоянии звукового устройства (State of Sound Device).

*Атрибуты:*

State of Sound Device : Boolean – Состояние звукового устройства (Рабочее/Не рабочее).

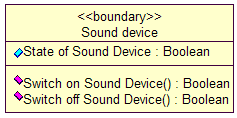


Рисунок 8 - Класс Sound Device <<boundary>> с операциями и аргументами

1. Регулятор (Regulator)

*Операции:*

Change factor T2() : Boolean – Операция изменения температуры T2 (Определение нижней границы температуры в холодильнике)

При выполнении операций они возвращают 1 - если операция выполнилась без ошибок, 0 – если произошла поломка. Значение сохраняется в состоянии регулятора (State of Regulator).

*Аргументы:*

Factor T2 : Float – Величина заданной температуры.

State of Regulator - Состояние устройства (Рабочее/Не рабочее).

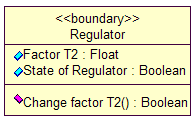


Рисунок 9 - Класс Regulator <<boundary>> с операциями и аргументами

Термометр (Thermometer)

*Операции:*

Measure T1() : Float – Операция измерения температуры в холодильнике. Выполняется постоянно.

При выполнении операций они возвращают 1 - если операция выполнилась без ошибок, 0 – если произошла поломка. Значение сохраняется в состоянии регулятора (State of Thermometer).

*Аргументы:*

State of Thermometer : Boolean - Состояние устройства (Рабочее/Не рабочее).

Factor T1 : Float – Аргумент хранит текущую температуру в холодильнике.

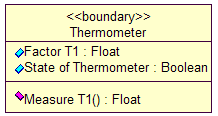


Рисунок 10 - Класс Thermometer <<boundary>> с операциями и аргументами

1. Индикатор (Indicator)

*Операции:*

Show T1() : Float – Вывести температуру на экран.

При выполнении операций они возвращают 1 - если операция выполнилась без ошибок, 0 – если произошла поломка. Значение сохраняется в состоянии регулятора (State of Indicator).

*Аргументы:*

Factor T1 : Float – Хранит значение переданной от термометра температуры.

State of Indicator : Boolean - Состояние устройства (Рабочее/Не рабочее)

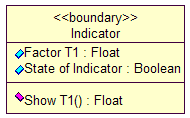


Рисунок 11 - Класс Indicator <<boundary>> с операциями и аргументами

1. Мотор (Motor)

*Операции:*

Start Motor() : Boolean – Операция запускает мотор при условии если T1>T2 (текущая температура с термометра превышает нижний порог заданный регулятором).

Stop Motor() : Boolean – Операция останавливает мотор если T1≤T2 (текущая температура с термометра ниже или равна нижнему порогу заданному регулятором). При выполнении операций они возвращают 1 - если операция выполнилась без ошибок, 0 – если произошла поломка. Значение сохраняется в состоянии регулятора (State of Motor).

*Атрибуты:*

State of Motor : Boolean - Состояние устройства (Рабочее/Не рабочее).

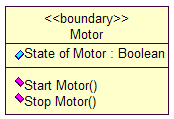


Рисунок 12 - Класс Motor <<boundary>> с операциями и аргументами

1. Журнал (Journal)

*Операции:*

Save data() – Сохранить данные направленные в журнал.

*Аргументы:*

Data : String – В него сохраняются данные.

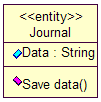


Рисунок 13 - Класс Journal <<entity>> с операциями и аргументами

1. Контроллер холодильника (Controller of refrigerator)

*Операции:*

Check States on breakage() : Boolean – Операция проверки на работоспособность устройств холодильника. Вся информация об ошибках записывается в журнал.

Check Time on Timer() : Boolean Проверить прошедшее время со включения таймера. Если время превышает 29 секунд, то передает команду на включение звукового сигнала.

Compare T1 & T2() – Сравнивает температуры, непрерывна. Если T1>T2 (текущая температура с термометра превышает нижний порог заданный регулятором) отдаёт команду на включение мотора.

Если T1≤T2 (текущая температура с термометра ниже или равна нижнему порогу заданному регулятором) передает команду мотору остановиться.

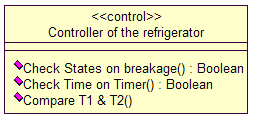


Рисунок 14 - Класс Controller of the refrigerator <<control>> с операциями и аргументами

**5.2 Построение диаграммы классов (Class Diagram)**

Диаграмма классов является основным логическим представлением модели и содержит детальную информацию о внутреннем устройстве объектно-ориентированной программной системы.

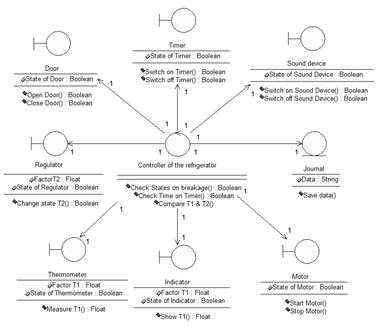


Рисунок 15 - Диаграмма классов (Class Diagram)

5.3 Построение диаграммы состояний (Statechart Diagram)

Определение состояний для классов моделируется с помощью диаграмм состояний. Главное назначение диаграммы состояний - описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение моделируемой системы в течение всего ее жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

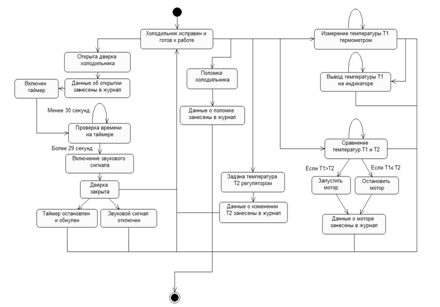


Рисунок 16 - Диаграммы состояний (Statechart Diagram)

**6. Реализация системы**

**6.1 Диаграммы компонентов (Component Diagram)**

Диаграмма компонентов служит частью физического представления модели, играет важную роль в процессе ООАП и является необходимой для генерации программного кода.

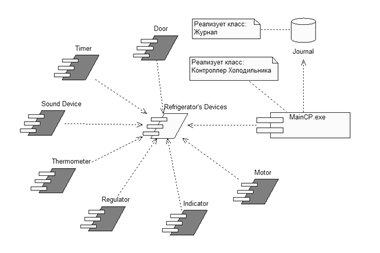


Рисунок 17 - Диаграммы компонентов (Component Diagram)

**6.2 Диаграмма размещения**

Диаграмма развертывания является второй составной частью физического представления модели и разрабатывается, как правило, для территориально распределенных систем. В данном проекте этот тип диаграммы не нужен, так как холодильник не имеет внешних связей с другими устройствами.

**6.3 Генерация кода**

*Процесс генерации кода состоит из четырех основных шагов:*

1. Проверка корректности модели.
2. Установка свойств генерации кода.
3. Выбор класса, компонента или пакета.
4. Генерация кода.

*Выполним предложенные шаги:*

1. Проверим модель на содержание ошибок.( Tools > Check Model).
2. Назначим компонентам классы(С помощью команды Assign) и выберем для всех компонент язык генерации ANSI C++.
3. Выберем все компоненты, которым назначили классы.
4. Выберем команду сгенерировать код на языке ANSI C++.

**Получившийся код**

Controller of the refrigerator.h

#ifndef CONTROLLEROFTHEREFRIGERATOR\_H\_INCLUDED\_B6AD0DF2

#define CONTROLLEROFTHEREFRIGERATOR\_H\_INCLUDED\_B6AD0DF2

//##ModelId=49521FA001E4

class Controller of the Controller of the refrigerator

{

public:

//##ModelId=495227D60148

Boolean Check States on Check States on breakage();

//##ModelId=49522913008C

Boolean Check Time on Check Time on Timer();

//##ModelId=4952295E0251

T1 Compare T1 T2();

};

#endif /\* CONTROLLEROFTHEREFRIGERATOR\_H\_INCLUDED\_B6AD0DF2 \*/

Controller of the refrigerator.cpp

#include "C:/Код/Controller of the refrigerator.h"

//##ModelId=495227D60148

Boolean Controller of the refrigerator::Check States on breakage()

{

}

//##ModelId=49522913008C

Boolean Controller of the refrigerator::Check Time on Timer()

{

}

//##ModelId=4952295E0251

Controller of the refrigerator::Compare T1 & T2()

{

}

Journal.h

#ifndef JOURNAL\_H\_INCLUDED\_B6AD5CEA

#define JOURNAL\_H\_INCLUDED\_B6AD5CEA

//##ModelId=4952203B037A

class Journal

{

public:

//##ModelId=495225C6036B

Save data();

//##ModelId=495225E7033C

String Data;

};

#endif /\* JOURNAL\_H\_INCLUDED\_B6AD5CEA \*/

Journal.cpp

#include "C:/КОД/Journal.h"

//##ModelId=495225C6036B

Journal::Save data()

{

}

Door.h

#ifndef DOOR\_H\_INCLUDED\_B6AD5A65

#define DOOR\_H\_INCLUDED\_B6AD5A65

//##ModelId=49521FD601D4

class Door

{

public:

//##ModelId=49522070029F

Boolean Open Door();

//##ModelId=4952207A0000

Boolean Close Door();

//##ModelId=4952210301F4

Boolean State of Door;

};

#endif /\* DOOR\_H\_INCLUDED\_B6AD5A65 \*/

Door.cpp

#include "C:/КОД/Door.h"

//##ModelId=49522070029F

Boolean Door::Open Door()

{

}

//##ModelId=4952207A0000

Boolean Door::Close Door()

{

}

Indicator.h

#ifndef INDICATOR\_H\_INCLUDED\_B6AD631B

#define INDICATOR\_H\_INCLUDED\_B6AD631B

//##ModelId=49522018032C

class Indicator

{

public:

//##ModelId=49522455007D

Float Show T1();

//##ModelId=495224950119

Float Factor T1;

//##ModelId=49527E1E02BF

Boolean State of Indicator;

};

#endif /\* INDICATOR\_H\_INCLUDED\_B6AD631B \*/

Indicator.cpp

#include "C:/КОД/Indicator.h"

//##ModelId=49522455007D

Float Indicator::Show T1()

{

}

Motor.h

#ifndef MOTOR\_H\_INCLUDED\_B6AD6A11

#define MOTOR\_H\_INCLUDED\_B6AD6A11

//##ModelId=49522059030D

class Motor

{

public:

//##ModelId=495225610280

Boolean Start Motor();

//##ModelId=495225920000

Boolean Stop Motor();

//##ModelId=495225A30186

Boolean State of Motor;

};

#endif /\* MOTOR\_H\_INCLUDED\_B6AD6A11 \*/

Motor.cpp

#include "C:/КОД/Motor.h"

//##ModelId=495225610280

Boolean Motor::Start Motor()

{

}

//##ModelId=495225920000

Boolean Motor::Stop Motor()

{

}

Regulator.h

#ifndef REGULATOR\_H\_INCLUDED\_B6AD09A2

#define REGULATOR\_H\_INCLUDED\_B6AD09A2

//##ModelId=4952202D03B9

class Regulator

{

public:

//##ModelId=495226D301D4

Boolean Change state T2();

//##ModelId=495224CE0119

Float FactorT2;

//##ModelId=49527E6F0177

Boolean State of Regulator;

};

#endif /\* REGULATOR\_H\_INCLUDED\_B6AD09A2 \*/

Regulator.cpp

#include "C:/КОД/Regulator.h"

//##ModelId=495226D301D4

Boolean Regulator::Change state T2()

{

}

Sound device.h

#ifndef SOUNDDEVICE\_H\_INCLUDED\_B6AD4A0B

#define SOUNDDEVICE\_H\_INCLUDED\_B6AD4A0B

//##ModelId=49521FEF0271

class Sound device

{

public:

//##ModelId=495223040242

Boolean Switch on Sound Device();

//##ModelId=495223150261

Boolean Switch off Sound Device();

//##ModelId=4952233101E4

Boolean State of Sound Device;};

#endif /\* SOUNDDEVICE\_H\_INCLUDED\_B6AD4A0B \*/

Sound device.cpp

#include "C:/КОД/Sound device.h"

//##ModelId=495223040242

Boolean Sound device::Switch on Sound Device()

{

}

//##ModelId=495223150261

Boolean Sound device::Switch off Sound Device()

}

Thermometer.h

#ifndef THERMOMETER\_H\_INCLUDED\_B6AD64D1

#define THERMOMETER\_H\_INCLUDED\_B6AD64D1

//##ModelId=4952200303B9

class Thermometer

{

public:

//##ModelId=4952239301B5

Float Measure T1();

//##ModelId=495223BC0186

Float Factor T1;

//##ModelId=49527E4C037A

Boolean State of Thermometer;};

#endif /\* THERMOMETER\_H\_INCLUDED\_B6AD64D1 \*/

Thermometer.cpp

#include "C:/КОД/Thermometer.h"

//##ModelId=4952239301B5

Float Thermometer::Measure T1()

{

}

Timer.h

#ifndef TIMER\_H\_INCLUDED\_B6AD34B3

#define TIMER\_H\_INCLUDED\_B6AD34B3

//##ModelId=49521FE1031C

class Timer

{

public:

//##ModelId=495221F501E4

Boolean Switch on Timer();

//##ModelId=4952229602CE

Boolean Switch off Timer();

//##ModelId=495222C00109

Boolean State of Timer;};

#endif /\* TIMER\_H\_INCLUDED\_B6AD34B3 \*/

Timer.cpp

#include "C:/КОД/Timer.h"

//##ModelId=495221F501E4

Boolean Timer::Switch on Timer()

{

}

//##ModelId=4952229602CE

Boolean Timer::Switch off Timer()

{

Заключение

В рамках курсового проекта было проведено проектирование системы по методологии UML с использованием программы Rational Rose и была построена модель программного обеспечения для холодильника, описание которого было изложено в пункте “Постановка задачи”, с использованием диаграмм и с генерацией конечного кода.

В ходе работы было создано несколько диаграмм, объявлены классы, их атрибуты и операции между ними, а также на диаграмме состояний был полностью описан процесс использования холодильника.

Структурный подход дает основу для создания диаграмм объектно-ориентированного подхода в среде Rational Rose.

Объектно-ориентированный подход включает в себя в первую очередь диаграмму вариантов использования, которая содержит действующих лиц и связанные с их деятельностью варианты использования.

Следующим этапом объектно-ориентированного подхода является создание классов с соответствующими атрибутами, операциями и описанием взаимодействия между классами.

Также в завершении проектирования модели был сгенерирован код программного обеспечения модели холодильника.

В дальнейшем может быть разработана база данных для хранения журнала событий холодильника.

Список использованных источников

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. 2-еизд.: Пер. с англ. – М.: Издательство Бином, СПб.: Невский диалект, 1999. – 332с.
2. Калянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение). М., "Лори", 1996. -202с.
3. Методология структурного анализа и проектирования. Марка Д.А., МакГоуэн К. М., "МетаТехнология", 1993. – 356с.
4. Новоженов Ю.В. Объектно-ориентированные технологии разработки сложных программных систем. М., 1996. – 245с.
5. Приемы объектно-ориентированного проектирования. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.: Пер. с англ. – М.: ДМК, 2000. – 354с.
6. Трофимов С. А. CASE-технологии: практическая работа в Rational Rose – М.: БИНОМ, 2000. – 405с.
7. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения.: Пер. с англ. – СПб: Питер, 2002. – 445с.
8. UML в кратком изложении. Применение стандартного языка объектного моделирования. Фаулер М., Скотт К.: Пер. сангл.– М.: Мир, 1999. – 273с.
9. UML и Rational Rose . Боггс У., БоггсМ.: Пер. сангл. – М.: Лори, 2000. – 266с.