Kyrgyz Republic Ministry of Education and Sciences

Kyrgyz State Technical University named after I. Razzakov

SOFTWARE EGINEERING DEPARTMENT

Bachelor Study Program in the field 710400

SOFTWARE ENGINEERING

Bachelor Module

Software Requirements Specification

Methodical Guide No.2

Sample of the Software Requirements Specification Developing Approach No.1 on Base of the Software to Solve the DMT’s Problems

On the discipline

“Software Requirements Specification”

Candidate of Technical Sciences, Professor

Iosif Grighorievitch Ten

BISHKEK 2016

Министерство образования и науки Кыргызской Республики

Кыргызский Государственный Технический Университет

им. И. Раззакова

Кафедра программного обеспечения компьютерных систем

НАПРАВЛЕНИЕ—710400

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Дисциплина для обучения студентов-бакалавров

«Разработка спецификации требований к ПО»

МЕТОДИЧЕСКОЕ ПОСОБИЕ №2

ИЛЛЮСТРАЦИЯ СПОСОБА №1 РАЗРАБОТКИ СПЕЦИФИКАЦИИ ТРЕБОВАНИЙ к ПО НА ПРИМЕРЕ ПО для РЕШЕНИЯ ЗАДАЧ ТПР

по дисциплине бакалаврской программы обучения

“Разработка спецификации требований к ПО”

кандидат технических наук, профессор

Тен Иосиф Григорьевич

Бишкек 2016

СОДЕРЖАНИЕ

|  |  |  |
| --- | --- | --- |
| № | Наименование задачи | Стр. |
| 1 | Предисловие | 4 |
| 2 | Throwaway prototyping approach | 5 |
| 3 | Use cases approach | 6 |
| 4 | Спецификация задачи ТПР. Задача № 5: Фирма планирует производство новой продукции быстрого питания в национальном масштабе | 7 |
| 5 | Part-III: THE SOFTWARE *REQUIREMENTS* SPECIFICATION FOR SOLVING the Task no.1 | 8 |
| 6 | III.1 Theoretical basis of the software requirements engineering and analysis | 8 |
| 7 | III.2 Software requirements specification to solve the DMT’s Problem Type-I such as Task no.1 | 11 |
| 8 | III.2.1: Functional requirements specification | 11 |
| 9 | III.2.2: The data requirements specification | 12 |
| 10 | III.2.3: Performance requirements | 13 |
| 11 | III.2.4: Constraints to software | 13 |
| 12 | III.2.5: Guidelines | 13 |

**Предисловие**

*Возможны несколько способов разработки спецификации требований к ПО для решения задачи ТПР:*

Способ №1 (THROWAWAY PROTOTYPING approach): в файле «TenIG(SRS\_Guide-2\_ForDMTProblemsType-I\_14pages)\_2018.01.28.docx»

Способ №2 (Use Cases approach): в файле «TenIG(SRS-3\_Task-5\_TwoUseCases\_7pages)\_2017.02.28.docx»

***Способ-1:*** {See file: “TenIG(SRS\_Guide-2\_ForDMTProblemsType-I\_14pages)\_2018.01.28.docx”} На основе Технической спецификации требований, которую должны студенты разработать и представить в тетради в качестве отчета по ***практической работе*** по предмету ТПР, и программы, которую создадут студенты в качестве отчета по ***лабораторной работе*** по предмету ТПР на основе Microsoft Access по Руководству Программиста из файла «DMT(ProgManToPSSType-1\_Ver-23-82pages)\_2016.05.17.docx», студенты должны разработать множество требований к ПО. ***Здесь при таком подходе очень важно, чтобы все требования были одинакового «масштаба», чтобы не было «сложного» требования к ПО, которое потребует 1000 строк кода, и «простого» требования, которое потребует 5-10 строк кода.***}

**Throwaway prototyping approach**

Construct software requirements specification on basis THROWAWAY PROTOTYPING approach to solve a DMT’s problem as it’s sketched below in the Figure SRS-1:



In order to catch an approach difference between THROWAWAY PROTOTYPING and EVOLUTIONARY PROTOTYPING need to compare Figure SRS-1 and SRS-2.



**Use cases approach**

One widely used approach to *documenting requirements* is “use cases”.

These are ***textual descriptions*** which can be ***augmented*** by UML use case diagrams.

Use cases take the point of view of the user or users of the system. A user who is carrying out a particular role is called an ***actor***.

A use case is a task that an actor needs the system to carry out.

*Use case – задокументированная последовательность действий (транзакций) в диалоге пользователя и системы, необходимых для решения данной задачи.*

For example, in the ***ATM*** system (*Appendix A.1: The ATM*), one of the things that a user does is ***withdraw cash***. This is a use case.

As part of ***withdrawing cash***, the user will have to carry out subtasks, such as offering up their card and entering a ***PIN***, but these smaller tasks are not use cases.

It is the overall user task that constitutes a use case.

You can find more detail description of the Use Cases approach with the example of the ***ATM use case*** in the file “TenIG\_SRS(Part-3\_Ver-14\_146slides)\_2017.01.29.pptx”

You will see that sometimes different ***use cases*** have parts in common.

It is sometimes difficult to identify *distinct use cases*. To overcome these difficulties, it’s useful to try get answer on the question such as: Is this collection a ***single use case***?

# Задача № 5: Фирма планирует производство новой продукции быстрого питания в национальном масштабе

***Спецификация проблемы:*** Фирма планирует производство новой продукции быстрого питания в национальном масштабе. Исследовательский отдел убежден в большом успехе новой продукции и хочет внедрить ее немедленно, без рекламной кампании на рынках сбыта фирмы. Отдел маркетинга положение вещей оценивает иначе и предлагает провестиинтенсивную рекламную компанию. Такая кампания обойдется в 100 000 долларов, а в случае успеха принесет 950 000 долларов годового дохода. В случае неуспеха рекламной компании (вероятность этого составляет 30%) годовой доход оценивается лишь в 300 000 долларов. Если рекламная компания не проводится вовсе, годовой доход оценивается в 400 000 долларов при условии, что покупателям понравится новая продукция (вероятность этого равна 0.8,) и в 200 000 долларов с вероятностью 0.2, если покупатели останутся равнодушными к новой продукции.

а) Постройте соответствующее дерево решений.

б) Как должна поступить фирма в связи с производством новой продукции?

Что требуется?

1. Этап №1: Получите решение задачи, используя критерий принятия решения «Средние ожидаемые прибыли»
2. Этап №2: Получите решение задачи, используя критерий принятия решения «Средние ожидаемые потери»
3. В каждом из этих этапов решения задачи подробно опишите процесс поиска решения, состоящий из 5 шагов, который приводит к решению задачи.

**Конец спецификации проблемы.**

**Part-III: THE SOFTWARE *REQUIREMENTS* SPECIFICATION FOR SOLVING the Task no.1**

**III.1 Theoretical basis of the software requirements engineering and analysis [1-8]**

At the present time, most requirements specifications are written in natural language.

One modern approach is to draw up two documents:

1. A **requirements specification** written primarily for users, describing the user’s view of  
   the system, expressed in natural language. This is the **substance of the contract**  
   between the users and the developers.
2. A **technical specification** that is used primarily by developers, expressed in some  
   more formal notation, and describing only a **part of the information** in the full  
   requirements specification.

If this approach is adopted, there is then the problem of ensuring that the two docu­ments are compatible.

A checklist of the contents of a requirements specification might look like this [1: Douglas Bell. Software Engineering for Students. A programming approach. 4rd Edition. Pearson Limited Edition, 2005. 447 pages]:

1. The functional requirements.
2. The data requirements.
3. Performance requirements.
4. Constraints.
5. Guidelines.

A number of notations and approaches are available to carry out requirement specification.

The notations range from informal (use case diagrams) through semi-formal (e.g. use cases) to formal (mathematics).

A useful checklist for the *ingredients* of a requirements specification is:

* + 1. functional requirements;
    2. data requirements;
    3. performance requirements;
    4. constraints;
    5. guidelines.

**The functional requirements**

The functional requirements are the real essence of a requirements specification. They state ***what*** the system should ***do***. It says nothing about ***how***the system should do their actions. Examples are:

*The system will display the titles of all the books written by the specified author.*

*The system will continuously display the temperatures of all the machines.*

Functional requirements are characterized by **verbs** that perform actions.

**The Data requirements**

Data requirements have three components:

1. users’ data that is input to or output from the system via screen, keyboard or mouse.
2. data that is stored within the system, usually in files on disk, for example, information about the books held in a public library.
3. information passed to or from another computer system, for example, to a server.

**Performance requirements**

These are measures of performance, some of which are quantitative, and some of which can be used as part of testing. Examples are:

* cost
* delivery date
* response times (e.g. *the system will respond to user requests within one second.*)
* data volumes (e.g. *the system must be able to store information on 10,000 employees*.)
* loading levels to be coped with (e.g. *the system must be able to cope with 100 transactions per minute from the point-of-sale terminals*).
* reliability requirements (e.g. *the system must have a mean time between failure of six months*.)
* security requirements.

**Constraints**

These are influences on the implementation of a system. An example is:

*The system must be written in Java.*

Constraints deal with such items as:

* \_ the computer hardware that is to be used
* \_ the amount of memory available
* \_ the amount of backing store available
* \_ the programming language to be used
* \_ interoperability constraints (e.g. the software must run under the latest version of Windows).

Constraints often address implementation (e.g. the specification of the programming language) and therefore should be included with caution. For example, this might be unnecessarily constraining:

*The search must use a binary chop method.*

**Guidelines**

A guideline provides useful direction for the implementation in a situation where there may be more than one implementation strategy. For example:

*The response times of the system to mouse clicks should be minimized.*

Or, as an alternative:

*The usage of main memory should be as small as possible.*

Many specifications mix up the areas identified above, so that, for example, design guidelines are sometimes confused with functional requirements.

Now that we have identified a suitable partitioning of a specification, we will examine some common deficiencies in specifications.

Vagueness is a common problem. For example:

*The interface will be user-friendly*

Sometimes requirements contradict each other, as in these two:

*The data will be stored on magnetic tape.*

*The system will respond in less than 1 second.*

Omissions or incompleteness can be difficult to identify. A typical area of specification that is omitted is that of how to deal with input errors from a user of the system. Many specifications mix up the areas identified above, so that, for example, design guidelines are confused with functional items.

Sometimes a requirement is simply unclear or susceptible to alternative interpreta­tions, and this, of course, may well be due to the use of natural language in the specifi­cation.

All in all, constructing a successful specification is a demanding activity that requires the clearest of thinking and review of the software requirements specification by a number of people.

Many specifications mix up the areas identified above, so that, for example, design guidelines are sometimes confused with functional requirements.

**III.2 Software requirements specification to solve the DMT’s Problem Type-I such as Task no.1**

**III.2.1: Functional requirements specification**/Software functional requirements state ***what*** the system should do. It says nothing about ***how***the system should do these actions. Функциональные требования к ПО – это описание того, ***что*** должна делать система***.*** Функциональные требования к ПО – не должны описывать ***как*** система реализует/выполняет эти действия.

SRS1.1: The software must have the possibility to enter the list of actions.

SRS1.2: The software must have the possibility to enter the list of events.

SRS1.3: Software makes it possible to check the validity of the event’s probability values of the Event List.

SRS1.4: Software makes it possible to check the validity of the collectively exhaustive property of the Event List.

SRS1.5: Software must assess the validity of the property of the Event List to constitute the collectively exhaustive set of events with the given tolerance value.

SRS1.6: The software must save the list of actions are used for solving the problem.

SRS1.7: The software must save the list of events are used for solving the problem.

SRS1.8: Software must use only mutually exclusive items of the Action List to create Act–Event combinations; it means that software must prevent the using of the ***repetitive*** items in the Action List since it leads to generate the useless act-event combinations and to increasing the running time of the program.

SRS1.9: Software must use only mutually exclusive and collectively exhaustive events of the Event List to create Act–Event combinations; it means that software must prevent the using of the ***repetitive*** items in the Event List since it leads to generate the useless act-event combinations and to increasing the running time of the program.

SRS1.10: The software must have the possibility to automatically generate the act–event combinations from the given Action List and Event List.

SRS1.11: The software must back-up the act–event combinations are used to solve the problem.

SRS1.12: The software makes it possible to submit for consideration the conditional profit for a user.

SRS1.13: The software makes it possible to compute and submit for consideration the weighted profit for a user.

SRS1.14: The software makes it possible to compute and submit for consideration the expected monetary value for a user.

SRS1.15: The software makes it possible to determine and submit for consideration the maximum available expected monetary figure for a user.

SRS1.16: The software must find out and submit for consideration the solution of the problem on basis expected monetary value criterion for a user.

SRS1.17: The software makes it possible to determine the conditional profit figure of the most favorable action for a given event.

SRS1.18: The software makes it possible to determine and submit for consideration the conditional opportunity losses figure for all act–event combinations for a user.

SRS1.19: The software makes it possible to determine and submit for consideration the weighted opportunity losses for a user.

SRS1.20: The software makes it possible to determine and submit for consideration the expected opportunity losses for a user.

SRS1.21: The software makes it possible to determine and submit for consideration the minimum of the expected opportunity losses figure for a user.

SRS1.22: The software must find out and submit for consideration the solution of the problem on basis expected opportunity losses criterion for a user.

SRS1.23: Software must check the validity of the input data and prevent using incorrect data.

SRS1.24: Software makes it possible to delete out-of-date Act–Event combinations.

SRS1.25: Software makes it possible to save actual/current Act–Event combinations.

SRS1.26: System shouldn’t create the act-event combinations if it detects an unacceptable situation where the Event List isn’t collectively exhaustive and it must give the advice for user to repair situation with the validity of the property of the Event List to constitute the collectively exhaustive set of events.

SRS1.27: Software must start to solve a problem as soon as (only if) the property of the Event List is the collectively exhaustive.

SRS1.28: The software makes it possible to determine and submit for consideration the sum of the probabilities figure of the Event List to check the validity. It is necessary to determine the sum of the probabilities figure of the Event List to check that the system has correctly defined this property.

SRS1.29: The software must stop of run and give an advice for user if it finds out a non-compliance with validity rule in collectively exhaustive property of the Event List to constitute a set of events or with validity of the entered data.

**III.2.2: The data requirements specification**/Software data requirements mean data that have three components: 1) the data that is ***input to*** or ***output from*** the system; 2) data that is ***stored within*** the system in files on disks and 3) the information passed to or from another computer system, for example, to a server. **–** Требования к данным ПО подразумевают/включают ***три*** компоненты: 1) входные или выходные данные; 2) данные, хранящиеся внутри системы на дисках; 3) данные, переданные в или принятые от другой компьютерной системы, например, на сервер.

SRS2.1: The software makes it possible to supply by the set of actions to solve the problem.

SRS2.2: The software makes it possible to supply by the set of events to solve the problem.

SRS2.3: The software must be supplied by the figure of the Tolerance value to check the acceptability of the collectively exhaustive property of the Event List.

SRS2.4: The software makes it possible to enter the quantitative/numerical assessment of the utility/benefits (for example, conditional profit figure) as the assessment of each act-event combination. *NB: Это максимально универсальное описание требований к ПО, которое подходит к ПО для решения любой задачи ТПР Типа-1.*

SRS2.5: The software makes it possible to determine the name of action that is the best solution on basis expected monetary value criterion and submit it as the output of the system.

SRS2.6: The software makes it possible to determine the name of action that is the best solution on basis expected opportunity losses criterion and submit it as the output of the system.

SRS2.7: The software makes it possible to produce the report or advice for the customer of the system why the recommended action is the best solution of the problem on basis expected monetary value criterion and submit it as the output from the system.

SRS2.8: The software makes it possible to produce the report or advice for the customer of the system why the recommended action is the best solution of the problem on basis expected opportunity losses criterion and submit it as the output from the system.

**III.2.3: Performance requirements**/Software performance requirements **–** требования к рабочим характеристикам; требования к функционированию ПО

SRS3.1: The software must be delivered by such-and-such a date (for example, 12 June 2016) and cost no more than $50,000.

SRS3.2: The computer must respond within 1 second to any request.

SRS3.3: The computer should be able to store information on up to 100,000 items in the list of actions.

SRS3.4: The computer should be able to store information on up to 300,000 items in the list of events.

SRS3.5: The reliability requirements – the system must have a mean time between failures of 6 months.

**III.2.4: Constraints to software**/Software design constraints **–** Ограничения на разработку ПО

SRS4.1: The computer system should respond to commands from the keyboard to enter the list of actions

SRS4.2: The computer system should respond to commands from the keyboard to enter the list of events.

SRS4.3: The database management system such as Microsoft Access should be used to construct/implement the software system.

SRS4.4: The requirements to the property of the event list to be the collectively exhaustive must be verified with the tolerance value no more than figure 1E–05.

SRS4.5: The software must be fully documented. It means software must have compulsory accompanying documents such as Software Requirements Specification, Software Technical Specification, User’s Manual, and Programmer’s Guide.

SRS4.6: The software must be easy to maintain. The former means that the architecture of the program has to be the right modularity as well as possible.

SRS4.7: The data that were used to solve the problem will be stored on hard disk.

SRS4.8: The software must run on a standard PC with a 500 gigabyte hard disk.

SRS4.9: The software must take into account to produce act–event combinations exceptionally mutually exclusive list of actions.

SRS4.10: The software must take into account to produce act–event combinations exceptionally mutually exclusive and collectively exhaustive list of events.

SRS4.11: Software must start to solve a problem only if the property of the Event List is the collectively exhaustive. It means that sum of the event’s probabilities of the set of events is satisfying the constraint:

**III.2.5: Guidelines –** Software developments guidelines. Guideline provides useful guidance for the implementation in a situation where there may be more than one implementation strategy. **(*Целевые* указания:** руководящие указания; методические рекомендации; директива – официальные предложения и советы по поводу действий в определенной ситуации, для достижения определенной ***цели*** и т. д.**)**

SRS5.1: The response times of the system to keyboard requests should be minimized.

SRS5.2: The usage of main memory should be as small as possible.

SRS5.3: The software should back up the set of previous solutions with support data as maximum as possible.

SRS5.4: Software makes it possible back up as much as possible the Act–Event combinations are created to solve the problem with the corresponding report about recommended solution.

SRS5.5: The program is to be maximally easy to maintain. It means that the architecture of the program has to be with the maximally modular-sized property or, in other words, the program shouldn’t have a ***spaghetti-structure codes***.

NB: *Many specifications mix up the areas identified above, so that, for example, design guidelines are sometimes confused with functional requirements.*