

CS 480 / CS 481

Guideline for preparation of the

Operational Specification (OpSpec)

William S. Junk

Computer Science Department

University of Idaho

Revision C

February 23, 2007

Table of Contents

The Operational Specification (OpSpec).....	1
Guideline Purpose	1
References.....	1
Government, National, and International Standards	1
Terms, Definitions, and Acronyms	1
Purpose of the Operational Specification	2
Operational Specification Annotated Outline	3
Title Page	3
Table of Contents	3
Revision History	3
1.0 Introduction	4
1.1 References and Applicable Documents	4
1.2 Terms, Definitions, and Acronyms	4
2.0 Operational Overview	5
2.1 System Context and Operational Characteristics	5
2.2 Use Cases	6
3.0 Operational Requirements	7
3.1 Operational System Platform & Environment	7
3.2 Operations.....	7
3.3 External Interfaces, Capacity, and Performance Requirements	10
3.4 Non-functional Requirements	11
3.5 Constraints	11
APPENDIX	11

Revision History

Revision	Date	Description of Change
C	23 February 2007	Made use cases a part of Section 2.
B	20 September 2005	Reinstated Section 3 and use it to define use cases.
A	22 August 2005	Provision for combining Sections 2 and 3.
Initial	30 January 2004	Initial document release.

The Operational Specification (OpSpec)

This guideline for preparing the Operational Specification (OpSpec) is intended for use in capstone design projects developed by students in the Computer Science Department, University of Idaho, Moscow, Idaho. This guideline is one of a series of guidelines that describe preparation of software-oriented technical documentation that should be produced as of an engineering-oriented system development life cycle.

It is not the intent of this guideline to describe or prescribe a particular development process or life cycle. The concept and content of the OpSpec described in this guideline are compatible with several different product development strategies, from highly research oriented, exploratory projects, to sustaining software maintenance efforts.

Guideline Purpose

This guideline addresses the writing of an Operational Specification (OpSpec) by providing guidance on its format and content. The intent of the Operational Specification is to provide a complete description of the external behavior of the software system to be developed with a strong emphasis on capturing the perspective seen by the user. With a complete Operational Specification the software design activity should be able to proceed with minimal additional input from the user. Of course, this depends to some extent on the type of life cycle being employed. The Operational Specification can be developed initially and used as the basis for further development, or it can be developed in several stages as an incremental activity for an evolutionary development approach.

The material that follows is intended to provide insight into the information that should be presented in the OpSpec. It is critical to quality of the completed project that format, representation techniques, and change control be properly planned and implemented before the OpSpec is started. Considerations must be given to the type of system to be developed (real time or non-real time), the available development tools, and the available resources.

The software engineering discipline suffers from potential confusion over the terminology that is in every day use. For purposes of this guideline we are considering the terms, *operation*, *requirement*, *feature*, and *capability*, to be approximately the same. Each is some fundamental activity the system will be required to provide. For the most part we have chosen to use the term *operation* as the basic element of system behavior for which the OpSpec is the defining document.

References

Government, National, and International Standards

ANSI/IEEE Std 830, IEEE Guide to Software Requirements Specifications, Institute of Electrical and Electronics Engineers, New York, 1984.

This standard provides guidance on the preparation and content of Software Requirements Specifications.

Terms, Definitions, and Acronyms

OpSpec	Operational Specification
SDD	Software Design Description
V&V	Verification and Validation

Purpose of the Operational Specification

The purpose of the Operational Specification is to capture and provide a basis for agreement on what the software product (or the portion of the product to be implemented in software) is to do. The OpSpec represents the first software-specific document produced during development. Its purpose is to provide a means of:

- communications among system analysts, software designers, engineers, and software quality engineers,
- supporting software design and software quality assurance activities,
- supporting system testing activities,
- supporting Verification and Validation activities, and
- supporting a controlled system evolution.

The OpSpec is the base document governing the system's software design and is very useful in the support of test design work.

Operational Specification Annotated Outline

The following paragraphs describe the contents of the Operational Specification. As you prepare the OpSpec, remember that it is being written for multiple audiences. The customer and possibly the end user are members of that audience. It must be understandable to them. The OpSpec will be the contract between you and the customer. From their perspective it defines which requirements the project will fulfill. Other potential readers of the OpSpec include software architects, software designers, software implementers, and software testers.

Title Page

The title page identifies the name of the system, the type of document (in this case an Operational Specification), the authors and their affiliation, the revision number of the document, and the date the document was released.

Table of Contents

The table of contents provides a quick reference to the major sections and subsections of the document.

Revision History

The revision history is a record of the changes that have been made to the document since its initial publication. A tabular representation is typical, containing columns for revision number, date of revision, and a brief description of the changes introduced in the revision. In association with revision publication, it is common practice to use change bars or other accepted mechanisms to identify the specific document content that has changed since the previous issue of the document was published. This makes it much easier for readers to identify what has changed, which is often the material in which they are most interested.

1.0 Introduction

The introduction contains a brief summary and description of the product to be developed and its intended use. The purpose of this section is to give the reader a preview of the document's contents and the product to which it applies. A single paragraph should suffice.

1.1 References and Applicable Documents

In this paragraph provide a list of the documents that directly relate to the preparation and understanding of the product described in this OpSpec document. Examples include books, articles, correspondence with the customer and documents describing the current system or operation. In general, any documents provided by the customer or intended user that help define the ideas presented in the OpSpec should be referenced. Only include documents that were used to develop the technical content of the OpSpec. Do not include the document guidelines for the course, or reference books on software engineering, etc. For any reference cited, include complete bibliographic information that will be sufficient to uniquely identify and locate the reference.

1.2 Terms, Definitions, and Acronyms

Provide definitions for all terms and acronyms used in your OpSpec that may not be commonly known by all potential readers of the document.

2.0 Operational Overview

As a general guideline, consider that the material appearing in Section 2 of the OpSpec should provide an overview of factors that affect the product and its requirements. It is what we often call an "Executive Summary." It should clearly define what the system is supposed to do, but only at a high level. This section should contain approximately one page of text and a supporting diagram. Remember, this is the first significant section that the readers of your document will see. Many of the readers (e.g., your instructor) may not be as familiar with the application as you are. Organizing Section 2 so that concepts and ideas are presented in a logical order gradually exposing the nature of your system and the context in which it will operate will significantly facilitate understanding of the material presented in subsequent sections of the OpSpec. Some readers may only read this section to get a general idea of what the system is supposed to do.

2.1 System Context and Operational Characteristics

This section should put the product you are developing into perspective with the users, other related activities, and products or projects within the customer's environment. You need to state if the product is independent and totally self-contained, if it must interface with another system (manual or automated), or if it receives data from another system.

If you are developing a product that is a component of a larger system or project, then this section should:

- briefly describe the functions of each significant component of the larger system or project and identify the interface among the functions.
- identify the principal external interfaces of the new software product.
- provide a diagram showing the major components of the larger system or project, and the interconnections and external interfaces with your product.
- clearly indicate where and how the user will interact with your new product.

For a product that is stand-alone (not part of a larger system), this section should:

- identify the principal external interfaces of the new software product.
- provide a block diagram that shows the relationship of your product to the "usage environment."
- clearly indicate where and how the user will interact with your new product.

The block diagram mentioned above is often called a product's context diagram. It should include the user interaction with your product, sources of external inputs, and destination of external outputs produced by your product. When you identify inputs and outputs on the diagram(s) use meaningful names for the data associated with them. **Don't just include the diagram. Be sure to provide an appropriate narrative of it.**

You should provide a high level summary of the most important capabilities the product will possess. As an example, for an accounting program the features might include customer account maintenance, preparation of customer statements, and invoice preparation. In your text you address these without mentioning the vast amount of detail that each of those functions requires. Features discussed should generally correspond to major functional capabilities of your product and might be something that you would want to include in advertising literature if you were going to market the product commercially. There may also be need for features that address the user interface, interfaces with other products or data. Some "non-functional" characteristics may be important enough that you mentioned them in this section.

Presentation of the operations supported by the product should be organized in a way that makes them understandable to the customer or to anyone else reading the document. Operations should be identified and described in terms common to the user's or problem's domain. Do **not** confuse this with specifying the structure of individual portions of programs or subprograms. We are not interested in presenting any design information in the OpSpec.

You should consider this section of the OpSpec to be “informative” only. It is not used to identify specific requirements or operations that must be provided by the product. This section must not be used to state specific requirements. Rather it should be used to clearly reflect an understanding of the user's needs. Requirements and operations are fully specified in Section 3 of the OpSpec.

2.2 Use Cases

This section of the OpSpec presents the operational model for the system or product as a collection of use cases. A use case is an abstraction that provides a user-oriented view of the system with respect to a specific operation, collection of related operations, or activities. The use cases present the “big picture” that will be helpful in the interpretation of the individual requirements and operations presented in Section 3. The use cases do not divulge internal implementation details of which the user is unaware.

This section is an expansion upon the Operational Overview presented in Section 2.1. This section may include a diagram and will include narrative text for each use case that describes what the system does, and how it will function to fulfill the need for which the product is being created. The narrative is written from the perspective of the product user. The use cases should address all of the primary operations or activities that the product will provide. It does not need to address minor capabilities and some “trivial” error processing. It does not address design issues.

A few pages of text and a few well-thought out diagrams are normally sufficient to adequately present the use cases, although the length of this section will necessarily be a function of the scope and complexity of the application being described.

This section of the OpSpec is “informative” only. It is not used to identify specific requirements or operations that must be provided by the product. The detailed description of operations and their associated requirements will appear in Section 3. Section 2 should not be used to impose a specific product design solution or to specify unnecessary design constraints on the solution. Concentrate on how your product fits into the user's environment and how the user will perceive his or her interaction with the system to accomplish specific operational needs. Although this section is consider “informative only” you should ensure that there are no conflicts between the use cases and the specific requirements and operations presented in Section 3. You should also ensure that there are no conflicts with the overview presented in Section 2.1.

3.0 Operational Requirements

Section 3 of the OpSpec provides the detailed definition of requirements and operations that your product must satisfy.

3.1 Operational System Platform & Environment

The primary purpose of this section is to ensure that you and the customer are in agreement concerning the specific hardware and software configuration the customer will use for the finished product. It also serves to document compatibility requirements if someone else wants to use your product in a potentially different hardware or software environment.

If the operational system hardware configuration has been dictated by the customer, describe it in this subparagraph. Usually it is best to describe the minimum capabilities of the system including the CPU, memory capacity, and special peripherals required by your product. When the specific hardware has been listed, state which of the models is essential to the operation of your product. For example, if your system needs a printer, but the particular make and model doesn't matter, you should state this and provide the general requirements that must be met, such as the number of print positions, graphics support, etc. On the other hand, if you will use the special features of a specific printer, then you should identify the printer by make and model number and emphasize the hardware dependency being introduced.

If the customer has not specified the hardware configuration, indicate that fact in this subparagraph. However, you will need to describe some general information about the platform you will use, such as "IBM PC or compatible" or "Engineering Workstation."

An additional objective of this subsection is to ensure that you and the customer are in agreement concerning the specific supporting software configuration that the customer will use. It serves to further document the compatibility requirements if someone else wants to use your product in a different environment.

If the customer has specified a particular software environment then this subsection describes the supporting software that will be required to operate your product. When specified, this will normally include identification of the operating system. It may also include other support or utility software such as statistical software packages, math libraries, graphic libraries, scripting software, data base management systems, etc. Also included here should be reference to any existing custom application software that will be used. Usually the language compiler (that you require for code development) is not required for system operation and should not be referenced here. For the software specified, always include the version identification.

If the customer has not requested a specific supporting software environment, then indicate that situation in this subparagraph. You still need to identify the basic operating system for which you will be developing.

3.2 Operations

In this subsection you are to address the specific operations that must be supported by the product. You are to present the requirements in a tabular or list format similar to that shown in Table 1, however, it is recommended that you not use Microsoft Word's table feature. Tables are sometimes difficult to format correctly and Word's line numbering feature does not number lines that inside of tables. The tabular format allows simple definition of the requirements, and also makes it easier to review and validate them during and at the end of the project. The first item for each operation should uniquely identify the operation. Then for each operation, the second item associates a concise, yet descriptive, name to the operation. This approach makes maintenance of the OPSpec easier when it becomes necessary to find, reference, delete, add, or modify operations.

In defining the operations it is important to be complete and specific about what the system is to do. State all the required operations that must be supported. Use mathematical notation if it makes for more

precise, concise, or explicit statement of an operation. Pay particular attention to identification of error processing. That is, don't just assume that your product will always encounter the information or actions it expects under "normal" circumstances. Real users just aren't that dependable.

Table 1. Operation Specification

ID:	Name:	Priority:
Pre-conditions:		
Accessibility:		
Stimulus:		
Description of Nominal Operation:		
Alternative action or error conditions:		
Post-conditions:		
Performance:		
References:		

Definition of Table Entries

ID: A unique identification assigned to the operation.

Name: A short descriptive name by which the operation is known.

Priority: The priority assigned to the operation.

Pre-conditions: System conditions or state that are true before the operation is (or can be) executed.

Accessibility: Defines restrictions on who can access the operation or when the operation is available.

Stimulus: The command, command sequence, or actions originating externally (from a user, external interface, or external system) that invoke the operation.

Description of Nominal Operation: An extended description of what happens during performance of the operation. Consider only the behavior expected from the "normal" case.

Alternative actions or error conditions: Description of the behavior exhibited by alternative actions or error conditions. Clearly specify the condition(s) under which the alternative action or error condition is invoked.

Post-conditions: System conditions or state after the operation has been completed. It may be necessary to specify different post-conditions for the nominal and alternative actions.

Performance: Specification of time allowed for the operation, accuracy of a computation, amount of a resource that may be used, etc.

References: Identification of where the information contained in the specification of this operation originated.

For each operation you identify how important (priority) the operation is to the overall success of the product. A *must* is something that has to be included or the product won't meet the customer's basic needs. On the other hand a *want* is something desirable to include but the product can still be successful if it isn't included. Of course, your goal is to deliver everything your customer identifies, but just in case this isn't possible, this list will help you place emphasis on things that are essential. You may also find it helpful to establish a numerical priority assignment for each operation. For example, *Priority 1* would be "mandatory," *Priority 2* would be "highly desirable," and *Priority 3* might be "implement if time permits."

Pre-conditions document the system conditions or state that is assumed to be true when the operation is initiated. If there are variants of the operation being documented concurrently, it is possible that a different pre-condition may exist for each variant.

All operations need to have their normal behavior specified.

Post-conditions document the system conditions or state that exist after the operations is performed. If there are variants of the operation being documented concurrently, it is possible that a different post-condition may exist for each variant.

Each operation should also have a stated performance requirement. Typically, items such as execution speed, interactive response, memory usage requirements, computational accuracy, etc., are defined here when they are applicable to a specific operation. If you are having a hard time defining performance requirements think about the response time (or other problem characteristic) that would be just barely acceptable to your customer. Acceptable response time is usually related to the value of the operation to

the user. For a high value operation the user normally is willing to wait longer for the product to produce the results.

In large development projects information used to define the operations may come from a variety of sources. You may receive information in formal documents, from memos, and in e-mails. It is important to document the source(s) used for each requirement so that in the future if there is any question about the representation of the operation and its ability to satisfy the user's need, the source(s) can be consulted to confirm or correct the specification.

Remember that the OpSpec merely states what needs to be accomplished, **not** how it will be accomplished. The key to writing a good specification is to be clear and definitive. Avoid ambiguities and generalities. If you haven't written down enough information so that you could give the OpSpec to another person to design the product, then you aren't finished with the OpSpec. Be careful that you don't dictate how the product is to be designed. Document what each operation of the application must do.

3.3 External Interfaces, Capacity, and Performance Requirements

This section should also be used to describe in detail the characteristic of the **external** data to be input by or output from your product. For each piece of information provide the following:

- Descriptive identification (Salary, TaxRate, etc.)
- Textual description of what this data is.
- Type of information (real, integer, character, etc.)
- Size (number of characters, number of digits, etc.)
- Range (domain) of the data values (minimum and maximum values, enumerated values, etc.)
- Accuracy of the data (if appropriate)
- Relationship to other data (if appropriate)

When you decide on the descriptive identification, it is a good idea to use names that could be legal syntax in the programming language you will be using to implement the system, if the language is known. This provides important consistency and continuity between the OpSpec, other system documents, and the code.

You may need to describe a grouping of the data. For example, an item that might be identified as Personal Information would be made up of Salary, Tax Rate, Social Security Number, etc., which were previously described items. This is similar to a structure in a programming language, but don't make it language specific.

When replicated items will be processed, indicate the volume of information to be handled. This should include both typical cases and maximum cases or minimum cases depending on which limiting case is significant.

Some additional characteristics of the system may be important to specify that aren't readily associated with any individual operation. If existing files or data structures are to be used as interfaces to other systems, describe the file formats in this section or provide a reference to where the format is already documented. If appropriate, a copy of the existing documentation can be placed in an appendix to the OpSpec. Describe the anticipated file sizes and indicate any minimum or maximum size limitations that affect your program.

It is usually appropriate to define the **content** of important outputs such as reports or displays, but it is not necessary to explicitly define their **format**. A **tentative** format is acceptable but should be clearly stated as being tentative.

There may also be overall performance requirements that must be met, in which case you add those here.

This section does not attempt to define all the data that will exist in internal data structures of your programs or subprograms. This section is confined exclusively to external interfaces.

3.4 Non-functional Requirements

Non-functional requirements describe necessary attributes of the product that do not have a functional or operational nature. For example, usability, portability, maintainability, and reliability are potentially quite important for a product. In some situations there may be additional important attributes. In this paragraph describe the characteristics that are pertinent to your product. Although you won't be able to write the quality requirements with the same precision that you used for the operations, try to be as specific as you can. For example:

- The Mean Time Between Failure for the program must exceed 300 hours.
- Program organization must facilitate expansion or modification at a later date.
- Program organization must facilitate transportation to another operational environment at a later date. (This might involve avoiding system dependent features and programming language extensions.)
- The program must be usable by a laboratory technician following completion of the training workshop without requiring additional time to process each sample.

3.5 Constraints

This subsection should provide a general description of any items that will limit the developer's options for designing the system. Constraints are generally to be avoided if possible because they impose limitations that may lead to a less than ideal solution. However, if constraints are necessary it is important that they be clearly documented and observed during development. These can include:

- Higher-order language requirements
- User's desire to have common characteristics and/or behavior among several systems
- Specific requirements on the human interface to the system that have been imposed by the customer.

Since this subsection specifies those requirements which apply in general to the whole project, it should be verified that items listed here are real constraints imposed either by the customer or by the computer system to be used.

APPENDIX

Include in the appendix all supporting documentation that is unique to the project and is not accessible in other forms or in other places within the OpSpec. This may include correspondence concerning the project and materials provided by the customer. Do **not** include material in the appendix, such as illustrations and diagrams that should be an integral part of a required section of the OpSpec.