User’s Manual as a Requirements Specification

Daniel M. Berry, 1991, 1998, 2002, and 2003 with help from K. Daudjee, J. Dong, I. Fainchtein, M.A. Nelson, T. Nelson, & L. Ou

A Note About This Talk

This talk was first written in 1991, prior to the community's identification of the concepts of *scenarios* and *use cases*, as being helpful in requirements engineering (RE).

It is interesting to see a lot of examples of these concepts in these slides, but not so-named.

I have added text in this font in 1998, to rephrase in the new vocabulary.

Introduction -1

I have been asked what I believe a good requirements specification (RS) is.

Here is my opinion.

t

Introduction -2

I believe that the most useful document to write during requirements engineering is the user’s manual (UM).

When done right and at the right time, it can serve as a useful elicitation, analysis, and validation tool, and can even serve as a RS.

Introduction -3

Please note that I am not discounting the other requirements documents, including the SRS.

They may be required by the customer.

They may give useful information that is not contained in the UM.

Introduction -4

However, I have found the production of the UM a good focal point in the requirements engineering process and a good way to get on to paper at least the kernel of all the information that goes in the other documents.

Information in a UM -1

An RS for a CBS should

- **describe the CBS’s function and not the CBS’s implementation,**
- **describe the CBS from the user’s point of view and not the implementer’s,**

Information in a UM -2

A good UM for a CBS should

- **describe the CBS’s function and not the CBS’s implementation,**
- **describe the CBS from the user’s point of view and not the implementer’s,**

Hmmmm????

Fred Brooks’s Observation

In 1975, in *MM-M*, Fred Brooks equated the UM with the written RS:

The manual must not only describe everything the user does see, including all interfaces; it must also refrain from describing what the user does not see. That is the implementer’s business, and there his design freedom must be unconstrained. The architect must always be prepared to show* an *implementation for any feature he describes, but he must not attempt to dictate* the *implementation.

Tom DeMarco

Also, Tom DeMarco suggests in several places using UMs as RSs, most notably in *The Deadline*.

Steve McConnell

In *Software Project Survival Guide*, Steve McConnell says, metime prior to placing the prototype under change control, work can begin on a detailed user documentation (called the User Manual/Requirements Specification). This is the documentation that will eventually be delivered to the software’s end users. Typically, this documentation is developed at the end of the project, but in this book’s approach, it is developed near the beginning.”

Lisa & Macintosh

It is said that the UMs for the Lisa and Macintosh computers were written completely before implementation of their software began.

The UMs were given to systems programmers as the RS of the user interfaces (UIs) and of the underlying systems.

UMs and 5 Roles of a RS -1

I claim that

- **The process of writing a UM for a CBS is a good way to learn the CBS’s requirements.**
- **The process of writing a UM for a CBS helps to reconcile differences among the CBS’s stakeholders.**
- **A UM allows the customer of the CBS to validate that the projected CBS will be what he or she wants before resources are spent implementing a possibly incorrect CBS.**

UMs and 5 Roles of a RS -2

- **A UM makes it clear what must be implemented to obtain the required CBS.**
- **A UM allows deriving both covering test cases and expected results that allow verification that the implementation of the CBS does what it is supposed to do.**

Good UMs -6

A good UM seems to have the following elements:

1. **descriptions of underlying and fundamental concepts of the software,**

i.e., a Lexicon!

Good UMs -7

2. **a graduated set of examples each showing**
    - **a problem situation the user faces**
    - **some possible user responses to the problem in the form of commands to the software**
    - **the software's response to these commands**

    i.e., Use cases!

3. **a systematic summary of all the commands**

Good UMs -11

A good way to organize the first part is around the abstractions that you have found in the problem domain.

i.e., a Domain Model!

Each abstraction that survives the analysis should be explained in terms of

- **what the objects are**
- **what they do**
- **what is done to them**

Another Opinion

According to Richard Fairley in his 1985 *Software Engineering Concepts*, a preliminary UM should be produced at requirements definition time.

He proposes the following outline for the (preliminary) manual.

Preliminary UM -1

1. **Introduction**
    - **Product overview and rationale**
    - **Terminology and basic features**
    - **Summary of display and report formats**
    - **Outline of the manual**

Preliminary UM -2

2. **Getting started**
   - **Sign-on**
   - **Help mode**
   - **Sample run**
3. **Modes of operation: Commands/Dialogues/Reports**
4. **Advanced features**
5. **Command syntax and system options**

Where are the Scenarios?

Chapter 3, about Modes of Operation, is precisely a list of scenarios, dressed up as a list of

- problems the user might be faced with, and
- how to solve them with the software being described.

CBSs Admitting UMs as RSs -1

The approach of offering a UM as the principal or only RS of a CBS works only for those CBSs for which a UM describes all but trivially explained requirements.

CBSs Admitting UMs as RSs -2

Thus,

- **The CBS must have at least one kind of user.**
- **The CBS must provide all but trivially explained functionality through at least one UI, and all of this functionality must be clear from the behavior seen by a user.**
- **Each of the CBS’s NFRs must be well understood and easily described in prose.**

CBSs Admitting UMs as RSs -3

If a CBS has several kinds of users, one UMs manual can be written for each kind.

However, then achieving and maintaining consistency of all UMs becomes a problem.

CBSs Not Admitting UM RSs -1

- **Autonomous systems with no real human users: However, a description of the real world’s or other CBS’s behavior might suffice.**
- **A CBS for which one or more algorithms it computes is the major issue of a RS (and the UI is not an issue): e.g., a weather predictor**

CBSs Not Admitting UM RSs -2

- **A CBS with nontrivial NFRs that are not specifically visible to the user, e.g., security, reliability, and robustness (SR&R), for which the user does nothing to cause the NFR to kick in. The way SR&R is achieved is a major issue for the RS.**

For none of these CBSs would a UMs manual serve as a good RS.

Quotations Worth Quoting

Never underestimate the lack of sophistication of the unsophisticated user!

Never underestimate the lack of sophistication of the *sophisticated* user!

The donut from X was *so* bad that the best tasting part was the hole!

Conclusions

Writing a good RS is hard.

It is hard to motivate people to write one.

UM can be ideal RS

A UM is an ideal RS in many cases because, if it is well-written,

- **it is written at the level of what the user sees,**
- **it describes the basic concepts, and**
- **it does not describe implementation details.**

That is, it is written at the right level of abstraction for a RS.

But No One Makes UMs Any More

♩♪♩ Oh where, oh where have all the manuals gone?

I am reminded of a Foxtrot comic strip that has Peter, who is trying to do a History homework, screaming “Would it kill Microsoft to include a manual?”

True, But ...

It’s true, many companies do not produce UMs any more.

However, they *do* make help systems, and these provide a good covering set of scenarios.

The help system can be used, as we suggest, as the RS.

Help Systems vs. UMs

In a sense, the scenarios of a good help system are better than most UMs for our purposes because each scenario focuses on one way that users use the software described by the help system to do their work.