

CS445 / SE463 / ECE 451 / CS645

Software requirements specification & analysis

11. The user manual

Fall 2010 — Mike Godfrey

Meta-comments

- Opinions and ideas expressed herein are due to Prof. Dan Berry and his collaborators K. Daudjee, J. Dong, I. Fainchtein, M.A. Nelson, T. Nelson, & L. Ou
 - Some of them are very personal and even controversial!
- 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.
 - So you'll see these ideas popping up

Meta-comments

- After this slide, “I” means “Prof. Dan Berry”
 - “I” [Godfrey] have performed some minor editing, mostly to shorten existing text. Some paraphrasing was done. Some text was *emphasized*. And many slides were left out from the original presentation.
 - Prof. Berry’s original slides can be found on the course web site.
- Although advice as to basic structure of your UMs is given in these slides, you should *carefully* read through the example UM on the course web page
 - *WD-pic User’s Manual* by Lihua Ou

Introduction

<Prof. Dan Berry speaking>

- 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

- 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.
- 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.

Why write an RS of a CBS
before implementing it?

The problems of writing an RS

- Despite these clear benefits, many projects are unable to produce the RS, for a variety of reasons, some technical and some social.

Writing UM may be a solution

- This talk offers writing a UM for a CBS before implementing it as a way achieving the writing of a RS of the CBS before implementing it.
- The method both
 - produces a document that delivers the five benefits of writing a RS before implementation, and
 - helps ameliorate or mitigate the three problems that discourage the production of a RS before implementation.

Info in a UM

- An RS 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
- A good UM 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.”*

Demarco and McConnell

- Also, Tom DeMarco suggests in several places using UMs as RSs, most notably in *The Deadline*.
- In *Software Project Survival Guide*, Steve McConnell says:
 - “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.
- *[The original version of DOS on the Intel 8086 chip — called QDOS — is supposed to have been implemented from a user manual for the CP/M operating system. – MWG]*

UMs and 5 roles of a RS

I claim that:

1. The process of writing a UM for a CBS is a good way to learn the CBS's requirements.
2. The process of writing a UM for a CBS helps to reconcile differences among the CBS's stakeholders.
3. 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

4. A UM makes it clear what must be implemented to obtain the required CBS.
5. 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.

Motivating writing of a RS

— *“It’s difficult to write a good RS”*

- Writing a good RS is difficult because the advice to describe “what, not how” is easier said than done.
- Clients and users tend to describe solutions to possibly non-existent problems rather than just problems that need to be solved.
- Writing a good UM forces focusing on the user’s view of the CBS.
- With typical user in mind as the future audience of the UM, it’s easier to focus on the user’s view, the *what* of the CBS, and to avoid mentioning implementation details.

“It takes too long”

“It’s a waste of time”

- If a project produces a UM, help system or test cases, it writes a RS.
 - Any project for commercial or contracted software does so.
 - Thus there is time to write a RS; it’s already being done.
- But the UM, help system, test cases are written later.
 - Ah, but writing them earlier saves time and money for each requirement error found earlier when it costs an order of magnitude less to fix it.

Writing UM preferable to writing RS

- UM or help system needs to be written *eventually* for user's benefit, but RS is not likely to be looked at beyond beginning of coding.
- Good UM exposes full set of use cases, from which test cases can be written, but most SRSs tend to focus on functional and NFRs and skip use cases.

Requirements for RS

- It is most important that a requirements document
 - be readable, and
 - accurately and completely describe the software system to be built, a system that meets the client's desires and needs.
- All else is frosting on the cake.

Requirements for RS

- It must be readable because if not,
 - no one will be able to judge whether the document meets the second document requirement,
 - no one will be able to write the software to meet the system requirements,
 - no one will be able to judge whether the software meets the system requirements.

Good UMs

- My favorite way to write a RS for a system that will have users is to write a UM or a collection of UMs, one for each kind of user.
- Writing UM requires a clear conception of what the system is supposed to do, clear enough that the manual author can visualize *user scenarios* and describe both
 - what the user should say to the system and
 - what the system will respond to the user.

Good UMs

- So what does a good UM look like?
 - Well, it should be clear, no longer than necessary, and fun to read!
 - Actually, almost everyone knows a bad user manual when he or she tries to read one and cannot!
 - There is something of an art to writing a good UM.

Good and bad UMs

- I personally have found the following manuals good:
 - The PARADOX UM from Borland
 - The TEXbook by Knuth
 - The C Programming Language by Kernighan and Ritchie
 - The PIC UM by Kernighan
 - The EQN UM by Kernighan and Cherry
- I personally have found the following manuals bad
 - The C++ Programming Language by Stroustrup
 - The NROFF/TROFF UM by Ossana
 - The SCRIBE UM by Unilogic
 - The TBL UM by Lesk

Good UMs

A good UM seems to have the following elements:

1. Descriptions of underlying and fundamental concepts of the software, [i.e., a lexicon!]
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 [i.e., a reference manual]

Good UMs

- Having only the command summary [i.e., a reference manual]
 - loses many readers who do not understand the concepts, and
 - turns off many readers who just plain get bored reading page after page after page of boring command syntax and semantics.
- Leaving out the lexicon makes it very hard for the author to use a consistent vocabulary in writing the rest of the manual.

Good UMs

- Leaving out the “use cases” leaves the reader without any sense of what is important and how to use the system to solve his or her problems.
- A well-written set of “use cases” makes reading the manual, even the command summary, fun.

Good UMs

- The command summary must be consulted in order to fully explain why the input of an example solved the problem the example claims it does.
- It is in writing the lexicon and “use cases” that diagrams are most useful
 - ... although I have seen command summaries that use a collection of related diagrams, one per command, to explain the system response to every command.

Good UMs

- A good way to organize the lexicon 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

Good UMs

- Writing a good UM takes skill, and there is no substitute for that skill.
 - I hope that at least one person in each group has that skill.
- In an industrial situation, the client and the software house must hire good writers to write skillful conception and requirements documents.

English majors as documenters

- Bob Glass reports how successfully non-software-knowledgeable English majors were able to write high-quality descriptions of the grubby details of programs in documentation about these programs for maintainers.

Fairly's UM model

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

Fairly's UM model

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

Using a UM as an RS

- ... works only for those CBSs for which a UM describes all but trivially explained requirements.
- 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.

Using a UM as an RS

- If a CBS has several kinds of users, one UM manual can be written for each kind.
 - However, maintaining consistency of all UMs becomes a problem.

This won't work well for ...

- 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
- 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.

Users vs. developers

- It can be argued that a UM favors the user over the developer (designer and implementer).
- If a UM is use-case centred, it's hard to identify functions that must be implemented
- However, a good UM has also a feature-centred part listing all the individual features and describing all of the options of each.
 - This part is organized much as is a traditional SRS.
 - The designer and implementer can find the functions already identified.

“It don’t come easy”

- Writing the UM is hard, but so is writing a RS!
 - So you are no worse off!

All the gory details

- No requirements specification method that does not force working out the details is going to work.
 - It is only in working out the details that all the show-stopping exceptions and interactions are going to be discovered.
- These details can be worked out in any of several media:
 - the software itself,
 - a complete formal specification,
 - a complete, traditional SRS, or
 - a complete, scenario-based UM.

All the gory details

- The advantage of UM is that changing the manual consistently is much cheaper than changing either the software itself or a complete formal specification.
- Also, unlike a complete, traditional SRS, a UM is both needed and perceived as needed after the software is delivered.
 - Thus, the motivation to keep it up to date is higher than that to keep a traditional SRS up to date.

All the gory details

- The advantage of the software itself or a complete formal specification is that it is hard to handwave over the details, to cheat to leave the impression of completeness when details are missing.
 - If details have been left out, the software will not work or the formal specification cannot be verified to satisfy requirements.

All the gory details

- While it is fairly easy to leave details out of a UM, since the UM is intended to be delivered with the software to help naive users, the incentive is to get those details in.
 - Thus, it is an issue of finding a right medium for expressing detailed requirements that is both cheap to change, but hard to handwave one's way to a false impression of completeness.

Summary

- 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.

</Prof. Dan Berry speaking>

Some examples

- WD-pic Manual, by Lihua Ou
 - <http://www.student.cs.uwaterloo.ca/~cs445/Fall2009/exampleDocs/WD-picManual.pdf>
- iPhone User Guide, Apple Computers
 - http://manuals.info.apple.com/en_US/iPhone_User_Guide.pdf
- GNU Image Manipulation Program (GIMP) User Manual
 - <http://docs.gimp.org/en/>

[This and the remaining slides are due to Mike Godfrey, tho he didn't actually create any of the content.]

1	Introduction	1
1.1	Product overview	1
1.2	First sample run	2
2	Convention	6
2.1	User assumption	6
2.2	Notational conventions	6
2.3	Terms	7
2.4	Abbreviation	10
2.5	Basic user interface goals	11
2.6	First sample redone	11
2.7	Using grid & gravity	13
2.8	Organization of this manual	17
3	Basic Use Cases	18
3.1	Affecting the session	18
3.1.1	Basic file manipulation	18
3.1.2	Standard Edit menu items	20
3.1.3	Using external editor on the IE	21
3.1.4	Selecting an object	23
3.1.5	Defining grid	23
3.1.6	Activating grid	24
3.1.7	Setting gravity	25

3.1.8	Setting the current insertion point	25
3.2	Affecting the IR	27
3.2.1	Inserting an object	27
3.2.2	Adjusting attributes during insertion	27
3.2.3	Indicating x, y coordinates	29
3.2.4	Adding text	30
4	Advanced Features	32
4.1	Affecting the session	32
4.1.1	Setting font and size of text	32
4.1.2	Preferences	34
4.2	Affecting the IR	34
4.2.1	Changing values of variables	34
4.2.2	Changing attributes of an existing object	35
4.2.3	Constructs	38
4.2.4	Macros	42
4.2.5	File conversion	44
4.2.6	Labelling	46
5	Trouble Shooting & Tips	48
5.1	Trouble shooting	48
5.2	Tips	48
6	Limitations	49

Chapter 1: Getting Started

- 9 Viewing the User Guide on iPhone
- 9 What You Need
- 10 Activating iPhone
- 10 Installing the SIM Card
- 10 Registering iPhone
- 11 Setting Up iPhone Using VoiceOver
- 11 Syncing
- 16 Mail, Contacts, and Calendar Accounts
- 18 Installing Configuration Profiles
- 19 Disconnecting iPhone from Your Computer

Chapter 2: Basics

- 20 iPhone at a Glance
- 23 Home Screen
- 26 Buttons
- 28 Touchscreen
- 31 Onscreen Keyboard
- 37 Searching
- 38 Voice Control
- 39 Stereo Headset
- 40 Connecting to the Internet
- 43 Battery
- 45 Security Features
- 46 Cleaning iPhone
- 46 Restarting and Resetting iPhone

Chapter 3: Phone

- 47 Phone Calls
- 51 Visual Voicemail
- 54 Contacts
- 54 Favorites
- 54 Ringtones and the Ring/Silent Switch

- 55 Bluetooth Devices
- 56 International Calls

Chapter 4: Mail

- 59 Setting Up Email Accounts
- 59 Sending Email
- 60 Checking and Reading Email
- 64 Searching Email
- 64 Organizing Email

Chapter 5: Safari

- 66 Viewing Webpages
- 69 Searching the Web
- 69 Bookmarks
- 70 Web Clips

Chapter 6: iPod

- 71 Getting Music, Video, and More
- 73 Music and Other Audio
- 81 Videos
- 84 Setting a Sleep Timer
- 85 Changing the Browse Buttons

Chapter 7: Messages

- 86 Sending and Receiving Messages
- 88 Sharing Photos and Videos
- 88 Sending Voice Memos
- 89 Editing Conversations
- 89 Using Contact Information and Links
- 90 Managing Previews and Alerts

Chapter 8: Calendar

- 91 About Calendar
- 91 Syncing Calendars
- 92 Viewing Your Calendar
- 93 Searching Calendars
- 93 Subscribing to and Sharing Calendars
- 94 Adding Calendar Events to iPhone
- 95 Responding to Meeting Invitations
- 96 Alerts

Chapter 9: Photos

- 97 About Photos
- 97 Syncing Photos and Videos with Your Computer

98	Viewing Photos and Videos	130	Chapter 16: Notes
99	Slideshows	130	Writing and Reading Notes
100	Sharing Photos and Videos	131	Searching Notes
102	Assigning a Photo to a Contact	131	Emailing Notes
102	Wallpaper	131	Syncing Notes
103	Chapter 10: Camera	132	Chapter 17: Clock
103	About Camera	132	World Clocks
104	Taking Photos and Recording Videos	133	Alarms
105	Viewing and Sharing Photos and Videos	133	Stopwatch
105	Trimming Videos	134	Timer
106	Uploading Photos and Videos to Your Computer	135	Chapter 18: Calculator
107	Chapter 11: YouTube	135	Using the Calculator
107	Finding and Viewing Videos	135	Standard Memory Functions
108	Controlling Video Playback	136	Scientific Calculator Keys
109	Managing Videos	138	Chapter 19: Settings
109	Getting More Information	138	Airplane Mode
110	Using YouTube Account Features	139	Wi-Fi
111	Changing the Browse Buttons	140	VPN
111	Sending Videos to YouTube	140	Notifications
112	Chapter 12: Stocks	141	Carrier
112	Viewing Stock Quotes	141	Sounds and the Ring/Silent Switch
113	Getting More Information	142	Brightness
114	Chapter 13: Maps	142	Wallpaper
114	Finding and Viewing Locations	142	General
119	Bookmarking Locations	150	Mail, Contacts, Calendars
119	Getting Directions	153	Phone
121	Showing Traffic Conditions	156	Safari
121	Finding and Contacting Businesses	157	Messages
123	Chapter 14: Weather	158	iPod
123	Viewing Weather Summaries	159	Photos
124	Getting More Weather Information	159	Store
125	Chapter 15: Voice Memos	159	Nike + iPod
125	Recording Voice Memos	160	Chapter 20: iTunes Store
126	Listening to Voice Memos	160	About the iTunes Store
127	Managing Voice Memos	161	Finding Music, Videos, and More
128	Trimming Voice Memos	162	Purchasing Ringtones
128	Sharing Voice Memos	162	Purchasing Music or Audiobooks
129	Syncing Voice Memos	163	Purchasing or Renting Videos
		164	Streaming or Downloading Podcasts

165	Checking Download Status	196	Phone and Voicemail
165	Syncing Purchased Content	197	Safari, Text, Mail, and Contacts
165	Changing the Browse Buttons	200	Sound, Music, and Video
166	Viewing Account Information	201	iTunes Stores
166	Verifying Purchases	201	Removing the SIM Card
167	Chapter 21: App Store	202	Backing Up iPhone
167	About the App Store	204	Updating and Restoring iPhone Software
167	Browsing and Searching	205	Appendix B: Other Resources
169	Info Screen	205	Safety, Software, and Service Information
170	Downloading Applications	206	Viewing the User Guide on iPhone
171	Deleting Applications	206	Disposal and Recycling Information
171	Writing Reviews	207	Apple and the Environment
172	Updating Applications	208	Index
172	Syncing Purchased Applications		
173	Chapter 22: Compass		
173	Getting Compass Readings		
174	Compass and Maps		
176	Chapter 23: Contacts		
176	About Contacts		
176	Adding Contacts		
177	Searching Contacts		
178	Managing Contacts on iPhone		
180	Chapter 24: Nike + iPod		
180	Activating Nike + iPod		
181	Additional Nike + iPod Settings		
182	Chapter 25: Accessibility		
182	Accessibility Features		
183	VoiceOver		
189	Zoom		
190	White on Black		
190	Mono Audio		
190	Speak Auto-text		
191	Triple-click Home		
191	Closed Captioning and Other Helpful Features		
193	Appendix A: Troubleshooting		
193	Apple iPhone Support Site		
193	General		
195	iTunes and Syncing		

[Preface](#)

[1. GIMP User Manual Authors and Contributors](#)

[I. Getting Started](#)

[1. Introduction](#)

[1. Welcome to GIMP](#)

[1.1. Authors](#)

[1.2. The GIMP Help system](#)

[1.3. Features and Capabilities](#)

[2. What's New in GIMP 2.6?](#)

[2. Fire up the GIMP](#)

[1. Running GIMP](#)

[1.1. Known Platforms](#)

[1.2. Language](#)

[1.3. Command Line Arguments](#)

[2. Starting GIMP the first time](#)

[2.1. Finally . . .](#)

[3. First Steps with Wilber](#)

[1. Basic Concepts](#)

[2. Main Windows](#)

[2.1. The Main Toolbox](#)

[2.2. Image Window](#)

[2.3. Dialogs and Docking](#)

[3. Undoing](#)

[3.1. Things That Cannot be Undone](#)

[4. GIMPLite Quickies](#)

[4.1. Intention](#)

[4.2. Change the Size of an Image \(Scale\)](#)

[4.3. Make JPEGs Smaller](#)

[4.4. Crop An Image](#)

[4.5. Find Info About Your Image](#)

[4.6. Change the Mode](#)

[4.7. Flip An Image](#)

[4.8. Rotate An Image](#)

[5. How to Draw Straight Lines](#)

[5.1. Intention](#)

[5.2. Examples](#)

[4. Getting Unstuck](#)

[1. Getting Unstuck](#)

[1.1. Stuck!](#)

[1.2. Common Causes of GIMP Non-Responsiveness](#)

II. How do I Become a GIMP Wizard?

5. Getting Images into GIMP

1. Image Types
2. Creating new Files
3. Opening Files
 - 3.1. Open File
 - 3.2. Open Location
 - 3.3. Open Recent
 - 3.4. Using External Programs
 - 3.5. File Manager
 - 3.6. Drag and Drop
 - 3.7. Copy and Paste
 - 3.8. Image Browser

6. Getting Images out of GIMP

1. Files
 - 1.1. Saving Images
 - 1.2. Saving Files
2. Preparing your Images for the Web
 - 2.1. Images with an Optimal Size/Quality Ratio
 - 2.2. Reducing the File Size Even More
 - 2.3. Saving Images with Transparency

7. Painting with GIMP

1. The Selection
 - 1.1. Feathering
 - 1.2. Making a Selection Partially Transparent
2. Creating and Using Selections
 - 2.1. Moving a Selection
 - 2.2. Adding or subtracting selections
3. QuickMask
 - 3.1. Overview
 - 3.2. Properties
4. Using the Quickmask
5. Paths
 - 5.1. Path Creating
 - 5.2. Paths and Selections
 - 5.3. Transforming Paths
 - 5.4. Stroking a Path
 - 5.5. Paths and Text
 - 5.6. Paths and SVG files

11. Pimp my GIMP

1. Preferences Dialog

- 1.1. Introduction
- 1.2. Environment
- 1.3. Interface
- 1.4. Theme
- 1.5. Help System
- 1.6. Tool Options
- 1.7. Toolbox
- 1.8. Default Image Preferences
- 1.9. Default Image Grid
- 1.10. Image Windows
- 1.11. Image Window Appearance
- 1.12. Image Window Title and Statusbar
- 1.13. Display
- 1.14. Color Management
- 1.15. Input Devices
- 1.16. Input Controllers
- 1.17. Window Management
- 1.18. Folders
- 1.19. Data Folders

2. Grids and Guides

- 2.1. The Image Grid
- 2.2. Guides

3. Rendering a Grid

4. How to Set Your Tile Cache

5. Creating Shortcuts to Menu Functions

6. Customize Splash-Screen

12. Scripting

1. Plugins

- 1.1. Introduction
- 1.2. Using Plugins
- 1.3. Installing New Plugins
- 1.4. Writing Plugins

2. Using Script-Fu Scripts

- 2.1. Script-Fu?
- 2.2. Installing Script-Fus

[LOTS of stuff deleted]

III. Function Reference

12. Toolbox

1. The Toolbox

1.1. Tool Options

2. Selection Tools

2.1. Common Features

2.2. Rectangle Selection

2.3. Ellipse Selection

2.4. Free Selection (Lasso)

2.5. Fuzzy selection (Magic wand)

2.6. Select By Color

2.7. Intelligent Scissors

2.8. Foreground Select

3. Brush Tools

3.1. Common Features

3.2. Painting Tools (Pencil, Paintbrush, Airbrush)

3.3. Bucket Fill

3.4. Blend

3.5. Pencil

3.6. Paintbrush

3.7. Eraser

3.8. Airbrush Tool

3.9. Ink

3.10. Clone

3.11. Heal

3.12. Perspective Clone

3.13. Blur/Sharpen

3.14. Smudge

3.15. Dodge or Burn

4. Transform Tools

4.1. Common Features

4.2. Move

4.3. Align

4.4. Crop

4.5. Rotate

4.6. Scale

4.7. Shear

4.8. Perspective

4.9. Flip

[LOTS of stuff deleted]

[I. Keys and Mouse Reference](#)

- [Help](#) — Key reference for Help menu
- [Tools](#) — Key reference for the Tools menu
- [File](#) — Key reference for the File menu
- [Dialogs](#) — Key reference for Dockable Dialogs submenu
- [View](#) — Key reference for View menu
- [Edit](#) — Key reference for Edit menu
- [Layer](#) — Key reference for Layer menu
- [Select](#) — Key reference for Select menu
- [Filters](#) — Key reference for Filters menu
- [Zoom tool](#) — Key reference for the Zoom tool submenu

[Glossary](#)

[Bibliography](#)

[A. GIMP History](#)

- [1. The Very Beginning](#)
- [2. The Early Days of GIMP](#)
- [3. The One to Change the World](#)
- [4. Version 2.0](#)
- [5. What's New in GIMP 2.2?](#)
- [6. What's New in GIMP 2.4?](#)

[B. Reporting Bugs and Requesting Enhancements](#)

- [1. Making sure it's a Bug](#)
- [2. Reporting the Bug](#)
- [3. What Happens to a Bug Report after you Submit it](#)

[C. GNU Free Documentation License](#)

- [1. PREAMBLE](#)
- [2. APPLICABILITY AND DEFINITIONS](#)
- [3. VERBATIM COPYING](#)
- [4. COPYING IN QUANTITY](#)
- [5. MODIFICATIONS](#)
- [6. COMBINING DOCUMENTS](#)
- [7. COLLECTIONS OF DOCUMENTS](#)
- [8. AGGREGATION WITH INDEPENDENT WORKS](#)
- [9. TRANSLATION](#)
- [10. TERMINATION](#)
- [11. FUTURE REVISIONS OF THIS LICENSE](#)
- [12. ADDENDUM: How to use this License for your documents](#)

[D. Eeek! There is Missing Help](#)

[Index](#)

[No stuff deleted, as it turns out]

CS445 / SE463 / ECE 451 / CS645

Software requirements specification & analysis

11. The user manual

Fall 2010 — Mike Godfrey