Module Guide for UnderTree

Team 22, Capstoners
Palanichamy Veerash
Kannammalil Kevin
Qureshi Eesha
Ahmed Faiq

January 19, 2023

1 Revision History

Date	Version	Notes
January 14	1.0	Initial document created
January 15	1.1	Anticipated changes and hierarchy diagram added
January 17	1.2	Rest of document finished

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	**description**
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
HTTP	Hypertext Transfer Protocol, a protocol used to communicate over the internet
TCP	Transmission Control Protocol, a protocol used to communicate over the internet
PDF	A file type to display text and images

Contents

List of Tables

List of Figures

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The data structure and algorithm used to implement the virtual hardware.

AC2: How the components related to project editing will be displayed

AC3: How the current file being edited is displayed

AC4: The algorithm used to highlight syntax of the LaTex code

AC5: The algorithm used to highlight grammar and spelling errors of the latex code

AC6: How the list of files in the current project being edited will be displayed

AC7: The actions available on the file toolbar

AC8: Inputs needed to create a new file

AC9: Inputs needed to upload a file

AC10: Algorithm used for determine cursor position for all users currently editing the file

AC11: Algorithm used for highlighting text selected by other users

AC12: Algorithm used for synchronizing the file for all the users currently editing the file

AC13: The data type used to store files

AC14: The database queries used to save and retrieve file data

AC15: How the user will trigger the compilation of the LaTeX and download of the PDF file

AC16: How the user will view the PDF file

AC17: The algorithm used to compile the LaTeX code into a PDF

AC18: How the user will view and interact with the chat messages

AC19: The algorithm used to send and receive chat messages

AC20: The database queries used to save and retrieve chat data

AC21: The communication protocol used to communicate between users

AC22: The data type used for chat messages

AC23: How the instructions are displayed

AC24: How informational and actionable items related to project are displayed

AC25: How the project list are displayed

AC26: The methods used to delete a project

AC27: Different methods to create a project

AC28: Inputs needed to create a new project

AC29: Inputs needed to import a project

AC30: The different services offered in regards to creation, editing, and reading projects

AC31: The algorithm used to fetch project data from the database

AC32: The format of project data

AC33: The user interface to interact with GitHub

AC34: The user interface to login and logout with GitHub

AC35: The algorithm to authenticate the user with GitHub

AC36: The functions used to save and retrieve authentication data for a user

AC37: The datatype for authenticating a user

AC40: How data is stored

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

...

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 5. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Project Editing Module

M3: Editor Module

M4: Syntax Highlighting Module

M5: Spelling Error Module

M6: File List Module

M7: File Toolbar Module

M8: New File Module

M9: Upload File Module

M10: User Cursors Module

M11: Text Highlighting Module

M12: File Synchronization Module

M13: File Services Module

M14: File Database Interface Module

M15: PDF Module

M16: PDF Renderer Module

M17: PDF Compiler Module

M18: Chat Module

M19: Chat Services Module

M20: Chat Database Interface Module

M21: Chat Socket Module

M22: Chat Data Module

M23:	Instructions View Module
M24:	Projects
M25:	Project List
M26:	Project Deletion
M27:	Project Creation
M28:	New Project
M29:	Import Project
M30:	Project Services
M31:	Project Database Interface
M32:	Project Data
M33:	GitHub Module
M34:	GitHub Services Module
M35:	Authentication Module
M36:	Auth Service Module
M37:	Auth Database Interface Module
M38:	Auth Data Module
M39:	MongoDB

Table 1: Module Hierarchy

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Project Editing Module
	Editor Module
	Syntax Highlighting Module
	Spelling Error Module
	File List Module
	File Toolbar Module
	New File Module
	Upload File Module
	User Cursors Module
	Text Highlighting Module
	File Synchronization Module
	File Services Module
	File Database Interface Module
	PDF Module
	PDF Renderer Module
	PDF Compiler Module
	Chat Module
	Chat Services Module
	Chat Database Interface Module
	Chat Socket Module
	Chat Data Module
	Instructions View Module
	Projects Module
	Project List Module
	Project Deletion Module
	Project Creation Module
	New Project Module
	Import Project Module
	Project Services Module
	Project Database Interface Module
	Project Data Module
	GitHub Module
	GitHub Services Module
	Authentication Module
	Auth Service Module
	Auth Database Interface Module
	Auth Data Module
Software Decision Module	

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

7 Module Decomposition

Modules are decomposed according to the principle of “information hiding” proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *UnderTree* means the module will be implemented by the UnderTree software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

7.2 Behaviour-Hiding Module

7.2.1 Project Editing Module (M2)

Secrets: How the components related to project editing will be displayed

Services: Displays the various components of project editing which are the file editor, file list, PDF output and information about the project being edited

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.2 Editor Module (M3)

Secrets: How the current file being edited is displayed

Services: Displays the content of the current file being edited and details about that file

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.3 Syntax Highlight Module (M4)

Secrets: The algorithm used to highlight syntax of the LaTex code

Services: T module will highlight the text based on Latex syntax

Implemented By: Highlight.js

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.4 Spelling Error Module (M5)

Secrets: The algorithm used to highlight grammar and spelling errors of the latex code

Services: Given input latex code, this module will highlight the grammar and spelling errors in the latex code

Implemented By: BeyondGrammar

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.5 File List Module (M6)

Secrets: How the list of files in the current project being edited will be displayed

Services: Displays the list of files in the current project and allows the user to open them in the editor, rename or delete a file

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.6 File Toolbar Module (M7)

Secrets: The actions available on the file toolbar

Services: Allows for actions regarding file such as adding a new file or uploading a new file

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.7 New File Module (M8)

Secrets: Inputs needed to create a new file

Services: Creates a new file in the project using name and extension as input

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.8 Upload File Module (M9)

Secrets: Inputs needed to upload a file

Services: Uploads a local file to the current project being edited

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.9 User Cursors Module (M10)

Secrets: Algorithm used for determine cursor position for all users currently editing the file

Services: Gives the current cursor position for the a spcecifc user in the editor

Implemented By: YJS

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.10 Text Highlighting Module (M11)

Secrets: Algorithm used for highlighting text selected by other users

Services: Gives the current highlighted text for the a specific user in the editor

Implemented By: YJS

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.11 File Synchronization Module (M12)

Secrets: Algorithm used for synchronizing the file for all the users currently editing the file

Services: Displays the file with all the changes made by different collaborators up until that current instance of time

Implemented By: YJS

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.12 File Services (M13)

Secrets: The data type used to store files

Services: Offers services for creating new file, editing existing files, deleting a file, and getting information about one or more files

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.13 File Database Interface Module (M14)

Secrets: The database queries used to save and retrieve file data

Services: This module is responsible for providing the functions to make the relevant queries to save and retrieve file data

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.14 PDF Module (M15)

Secrets: How the user will trigger the compilation of the LaTeX and download of the PDF file

Services: This module is responsible for allowing the user to download, and start the compilation for the LaTeX to PDF

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.15 PDF Renderer Module (M16)

Secrets: How the user will view the PDF file

Services: This module is responsible for allowing the user to view the LaTeX file

Implemented By: PDF.js

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.16 PDF Compiler Module (M17)

Secrets: The algorithm used to compile the LaTeX code into a PDF

Services: Upon getting a request from the PDF module to compile the PDF this module is used to convert the LaTex code into a PDF which the user can view

Implemented By: node-latex

Type of Module: Library

7.2.17 Chat Module (M18)

Secrets: How the user will view and interact with the chat messages

Services: This module is responsible for allowing the user to view and input chat messages

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.18 Chat Services Module (M19)

Secrets: The algorithm used to send and receive chat messages

Services: This module is responsible for sending and receiving chat messages between users

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.19 Chat Database Interface Module (M20)

Secrets: The database queries used to save and retrieve chat data

Services: This module is responsible for providing the functions needed to make the relevant queries to save and retrieve chat data

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.20 Chat Socket Module (M21)

Secrets: The communication protocol used to communicate between users

Services: This module is responsible for providing the web sockets needed to communicate between users in a synchronized manner

Implemented By: Socket.IO

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.21 Chat Data Module (M22)

Secrets: The data type used for chat messages

Services: This module stores the datatype that will be used to represent chat messages

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.22 Instructions View Module (M23)

Secrets: How the instructions are displayed

Services: This module is responsible for allowing users to open and close as well as view the instructions in an organized manner

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.23 Projects (M24)

Secrets: How informational and actionable items related to project are displayed

Services: This module displays the project list, allows the creation of new project and also displays other items related to projects

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.24 Project List (M25)

Secrets: How the project list are displayed

Services: This module displays a list of all the available projects to the user which can be opened or deleted

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.25 Project Deletion (M26)

Secrets: The methods used to delete a project

Services: Allows users to delete the project

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.26 Project Creation (M27)

Secrets: Different methods to create a project

Services: Presents a user interface for users to choose how they want to create their project either from scratch or importing a pre-existing project

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.27 New Project (M28)

Secrets: Inputs needed to create a new project

Services: Presents a user interface for users to create, title and add contributors to a new project from scratch

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.28 Import Project (M29)

Secrets: Inputs needed to import a project

Services: This module provides a user interface for the user to import pre-existing projects into UnderTree

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.29 Project Services (M30)

Secrets: The different services offered in regards to creation, editing, and reading projects

Services: Offers services for creating new project, editing existing project, deleting a project, and getting information about one or more projects

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.30 Project Database Interface (M31)

Secrets: The algorithm used to fetch project data from the database

Services: Offers service to modify project data in the database

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.31 Project Data (M32)

Secrets: The format of project data

Services: Represents the format for the project data

Implemented By: UnderTree

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.2.32 GitHub Module (M33)

Secrets: The logic to display the GitHub buttons and modal

Services: Responsible for receiving the user data.

Implemented By: UnderTree

7.2.33 GitHub Services Module (M34)

Secrets: The algorithm to communicate with GitHub API

Services: Responsible for performing the GitHub operations.

Implemented By: UnderTree

7.2.34 Authentication Module (M35)

Secrets: The algorithm to access different authorization modules

Services: This module determines which module to use based on the action the user is performing

Implemented By: UnderTree

7.2.35 Auth Service Module (M36)

Secrets: The algorithm to handle the HTTPS requests and authentication

Services: This module contains the main logic for authentication with GitHub

Implemented By: UnderTree

7.2.36 Auth Database Interface Module (M37)

Secrets: The algorithm to save user authentication data

Services: This module serves as a wrapper and makes queries with the database MongoDB.

Implemented By: UnderTree

7.2.37 Auth Data Module (M38)

Secrets: The data types in authentication are declared here

Services: This module allows for the storing and retrieving of several key data structures that will be used for authentication.

Implemented By: UnderTree

7.2.38 MongoDB Module (M39)

Secrets: How data is stored

Services: Allows you to retrieve and store data from the database

Implemented By: MongoDB

7.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Table 2: Trace Between Functional Requirements and Modules

Req.	**Modules**
FR1	M23
FR2	M23
FR3	M33, M34, M35, M36
FR4	M33, M34, M35, M36
FR5	M24, M27, M28, M30, M31, M32
FR6	M24, M27, M28, M30, M31, M32
FR7	M33, M34, M37, M38, M39
FR8	M24, M27, M29, M30, M31, M32
FR9	M30, M31, M32, M33, M**??**
FR10	M30, M31, M32, M33
FR11	M26, M30, M31, M32, M39
FR12	M26, M30, M31, M32, M39
FR13	M24, M25, M30, M31, M32, M39
FR14	M24, M25, M30, M31, M32, M39
FR15	M24, M25, M30, M31, M32, M39
FR16	M2, M25, M30, M31, M32, M33, M35, M39
FR17	M2, M25, M30, M31, M32, M33, M35, M39
FR18	M6, M13, M14
FR19	M6, M13, M14
FR20	M3, M10
FR21	M3, M12
FR22	M3, M4
FR23	M3, M5

FR24	M7, M8
FR25	M7, M8
FR26	M6, M13, M14
FR27	M7, M9
FR28	M6, M13, M14
FR29	M6, M7
FR30	M6, M13, M14
FR31	M6
FR32	M13, M14
FR33	M13, M14
FR34	M3
FR35	M3, M13, M14, M38
FR36	M13, M14, M17
FR37	M2, M15
FR38	M13, M14, M17
FR39	M2, M15, M16
FR40	M2,M**??**
FR41	M2,M**??**
FR42	M2
FR43	M18
FR44	M18, M19, M20, M21, M22
FR45	M33, M35
FR46	M33, M35
FR47	M33, M35

Table 3: Trace Between Nonfunctional Requirements and Modules

Req.	**Modules**
NFR1	M2, M3, M18, M23, M24
NFR1	M2, M3, M18, M23, M24
NFR3	M2, M3, M18, M23, M24
NFR4	M33, M34, M35, M35
NFR5	M2, M3, M18, M23, M24
NFR6	M24, M27, M28, M30, M31, M32
NFR7	M33, M34, M35, M36, M37
NFR8	M2, M3, M18, M23, M24
NFR9	M30, M31, M32, M33, M39
NFR10	M30, M31, M32, M33
NFR11	M26, M30, M31, M32, M39
NFR12	M26, M30, M31, M32, M39
NFR13	M24, M25, M30, M31, M32, M39
NFR14	M24, M25, M30, M31, M32, M39
NFR15	M24, M25, M30, M31, M32, M39
NFR16	M2, M25, M30, M31, M32, M33, M34, M39
NFR17	M2, M25, M30, M31, M32, M33, M34, M39
NFR18	M6, M13, M14
NFR19	M6, M13, M14
NFR20	M3, M10
NFR21	M3, M12
NFR22	M3, M4
NFR23	M3, M5

NFR24	M7, M8
NFR25	M7, M8
NFR26	M6, M13, M14
NFR27	M7, M9
NFR28	M6, M13, M14
NFR29	M6, M7
NFR30	M6, M13, M14
NFR31	M2, M3, M18, M23, M24
NFR32	M13, M14
NFR33	M13, M14
NFR34	M2, M3, M18, M23, M24
NFR35	M3, M13, M14, M??
NFR36	M13, M14, M17
NFR37	M2, M15
NFR38	M13, M14, M17
NFR39	M2, M15, M16
NFR40	M2,M38
NFR44	M18, M19, M20, M21, M22

Table 4: Trace Between Anticipated Changes and Modules

AC	**Modules**
AC1	M1
AC2	M2
AC3	M3
AC4	M4
AC5	M5
AC6	M6
AC7	M7
AC8	M8
AC9	M9
AC10	M10
AC11	M11
AC12	M12
AC13	M13
AC14	M14
AC15	M15
AC16	M16
AC17	M17
AC18	M18
AC19	M19
AC20	M20
AC21	M21
AC22	M22
AC23	M23
AC24	M24
AC25	M25
AC26	M26
AC27	M27
AC28	M28
AC29	M29
AC30	M30
AC31	M31

AC32	M32
AC33	M33
AC34	M34
AC35	M35
AC36	M36
AC37	M37
AC??	M38
AC??	M39

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

Figure 1: Use hierarchy among modules

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.