

Module Guide for Software Engineering

Team #22, TeleHealth Insights

Mitchell Weingust

Parisha Nizam

Promish Kandel

Jasmine Sun-Hu

January 14, 2025

1 Revision History

Date	Version	Notes
01/11/25	1.0	Initial Draft
Date 2	1.1	Notes

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
Software Engineering	Explanation of program name
UC	Unlikely Change
[etc. —SS]	[... —SS]

Contents

1	Revision History	i
2	Reference Material	ii
2.1	Abbreviations and Acronyms	ii
3	Introduction	1
4	Anticipated and Unlikely Changes	2
4.1	Anticipated Changes	2
4.2	Unlikely Changes	2
5	Module Hierarchy	2
6	Connection Between Requirements and Design	3
7	Module Decomposition	3
7.1	Hardware Hiding Modules (M??)	4
7.2	Behaviour-Hiding Module	4
7.2.1	Question Bank Module (M??)	4
7.2.2	English Question Bank Module (M??)	4
7.2.3	Mandarin Question Bank Module (M??)	5
7.2.4	Matching Question Bank Module (M??)	5
7.2.5	Repetition Question Bank Module (M??)	5
7.3	Software Decision Module	5
7.3.1	Etc.	6
8	Traceability Matrix	6
9	Use Hierarchy Between Modules	7
10	User Interfaces	8
11	Design of Communication Protocols	10
12	Timeline	10

List of Tables

1	Module Hierarchy	3
2	Trace Between Requirements and Modules	6
3	Trace Between Anticipated Changes and Modules	7

List of Figures

1	Use hierarchy among modules	7
2	Sketch of Assessment Selection Page	8
3	Sketch of Parent Checklist Page	8
4	Sketch of Consent Popup	9
5	Sketch of Video Audio and Mic Test Page	9
6	Sketch of Example Question Page	10
7	Sketch of Assessment Completion Page	10

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 laid 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 specific hardware on which the software is running.

AC2: The format of the initial input data.

...

[Anticipated changes relate to changes that would be made in requirements, design or implementation choices. They are not related to changes that are made at run-time, like the values of parameters. —SS]

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 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

Level 1	Level 2
Hardware-Hiding Module	
	Question Bank Module
	English Question Bank Module
	Mandarin Question Bank Module
Behaviour-Hiding Module	Matching Question Bank Module
	Repitition Question Bank Module
	?
	?
	?
	?
	?
Software Decision Module	?
	?

Table 1: Module Hierarchy

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 2.

[The intention of this section is to document decisions that are made “between” the requirements and the design. To satisfy some requirements, design decisions need to be made. Rather than make these decisions implicit, they are explicitly recorded here. For instance, if a program has security requirements, a specific design decision may be made to satisfy those requirements with a password. —SS]

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. *Software Engineering* means the module will be implemented by the Software Engineering 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 (M??)

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

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 Question Bank Module (M??)

Secrets: Integrates and manages question bank submodules, provides logic for selecting and instructing the appropriate submodule.

Services: Acts as a facade to provide unified access to all question banks. This module handles requests for retrieving, adding, updating, or delegating questions to appropriate submodules.

Implemented By: QuestionBankService

Type of Module: Abstract Object

7.2.2 English Question Bank Module (M??)

Secrets: The format for storing, tagging and/or indexing English questions

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: EnglishQuestionManager

Type of Module: Abstract Data Type

7.2.3 Mandarin Question Bank Module (M??)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: MandarinQuestionManager

Type of Module: Abstract Data Type

7.2.4 Matching Question Bank Module (M??)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: MatchingQuestionService

Type of Module: Library

7.2.5 Repetition Question Bank Module (M??)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: MatchingQuestionService

Type of Module: Library

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: –

7.3.1 Etc.

8 Traceability Matrix

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

Req.	Modules
R1	M??, M??, M??, M??
R2	M??, M??
R3	M??
R4	M??, M??
R5	M??, M??, M??, M??, M??, M??
R6	M??, M??, M??, M??, M??, M??
R7	M??, M??, M??, M??, M??
R8	M??, M??, M??, M??, M??
R9	M??
R10	M??, M??, M??
R11	M??, M??, M??, M??

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M??
AC2	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

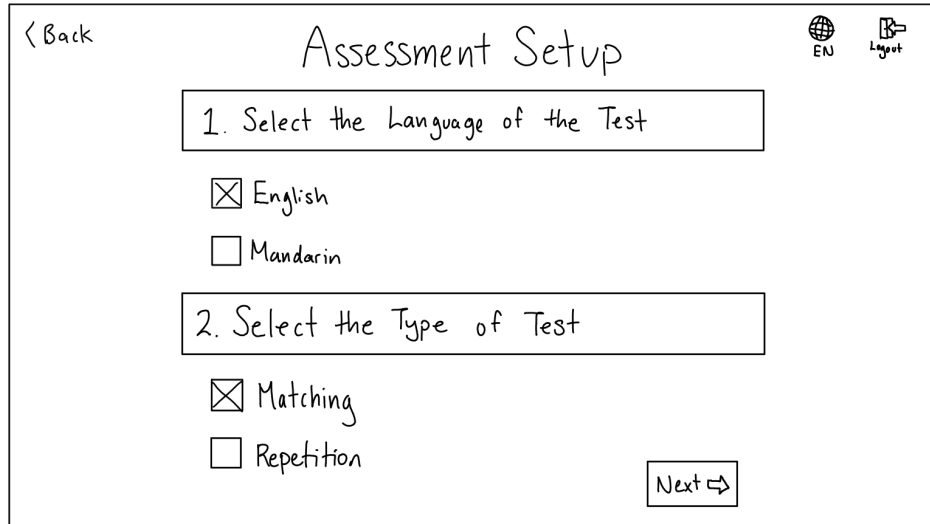
In this section, the uses hierarchy between modules is provided. [Parnas \(1978\)](#) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning 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.

[The uses relation is not a data flow diagram. In the code there will often be an import statement in module A when it directly uses module B. Module B provides the services that module A needs. The code for module A needs to be able to see these services (hence the import statement). Since the uses relation is transitive, there is a use relation without an import, but the arrows in the diagram typically correspond to the presence of import statement. —SS]

[If module A uses module B, the arrow is directed from A to B. —SS]

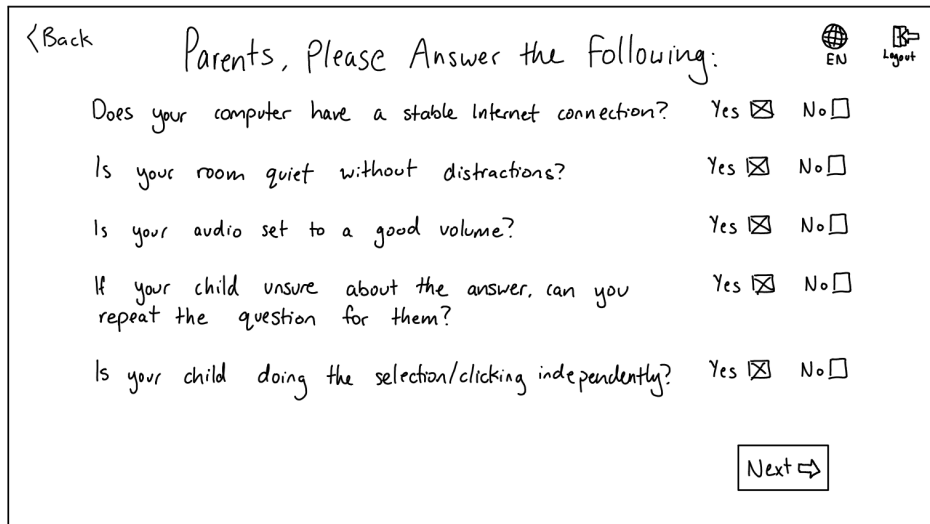
Figure 1: Use hierarchy among modules

10 User Interfaces



A hand-drawn sketch of a web page titled "Assessment Setup". In the top left corner is a "< Back" link. In the top right corner are two icons: a globe labeled "EN" and a key icon labeled "Logout". The main content area contains two numbered sections, each in a rectangular box. The first box is labeled "1. Select the Language of the Test" and contains two options: "English" with a checked checkbox and "Mandarin" with an unchecked checkbox. The second box is labeled "2. Select the Type of Test" and contains two options: "Matching" with a checked checkbox and "Repetition" with an unchecked checkbox. At the bottom right of the page is a button labeled "Next" with a right-pointing arrow.

Figure 2: Sketch of Assessment Selection Page



A hand-drawn sketch of a web page titled "Parents, Please Answer the Following:". In the top left corner is a "< Back" link. In the top right corner are two icons: a globe labeled "EN" and a key icon labeled "Logout". The main content area contains five questions, each followed by "Yes" and "No" options with checkboxes. The questions and their checked/unchecked status are: "Does your computer have a stable Internet connection?" (Yes checked, No unchecked), "Is your room quiet without distractions?" (Yes checked, No unchecked), "Is your audio set to a good volume?" (Yes checked, No unchecked), "If your child unsure about the answer, can you repeat the question for them?" (Yes checked, No unchecked), and "Is your child doing the selection/clicking independently?" (Yes checked, No unchecked). At the bottom right of the page is a button labeled "Next" with a right-pointing arrow.

Figure 3: Sketch of Parent Checklist Page

Video/Audio Recording Consent

We kindly ask for your consent to record video and audio. This recording will be used for analyzing responses and ensuring the quality of the assessment. Your data will be handled securely, only viewable by your clinician and used solely for the purpose outlined above.

Do you consent to the recording of video and audio for this assessment?

☐ Yes, I consent
 ☐ No, I do not consent

Figure 4: Sketch of Consent Popup

[< Back](#)

Video/Audio Test

EN
 Logout

Video Device

Webcam XYZ123

✓

Speaker Device

Speaker ABC

✓

EN
 Logout

Test

Microphone Device

Microphone 1

✓

Test

Next ➡

Figure 5: Sketch of Video Audio and Mic Test Page

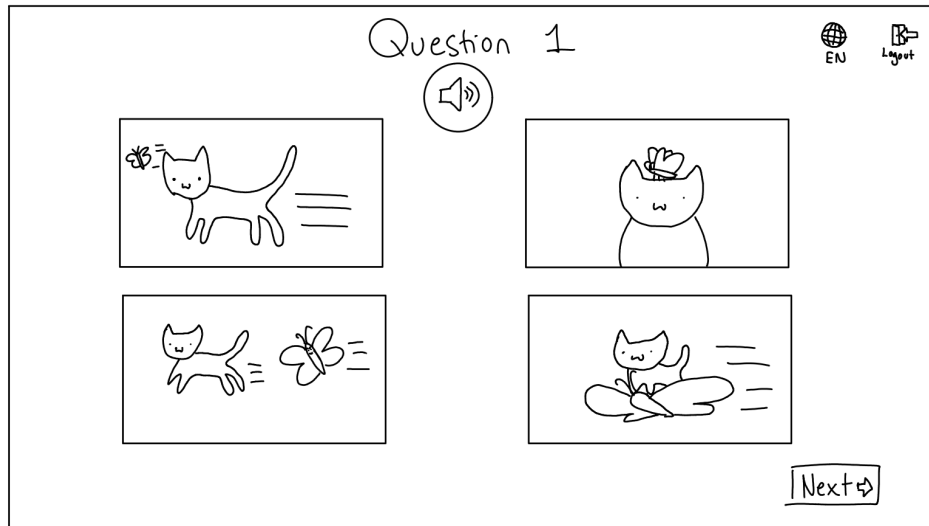


Figure 6: Sketch of Example Question Page

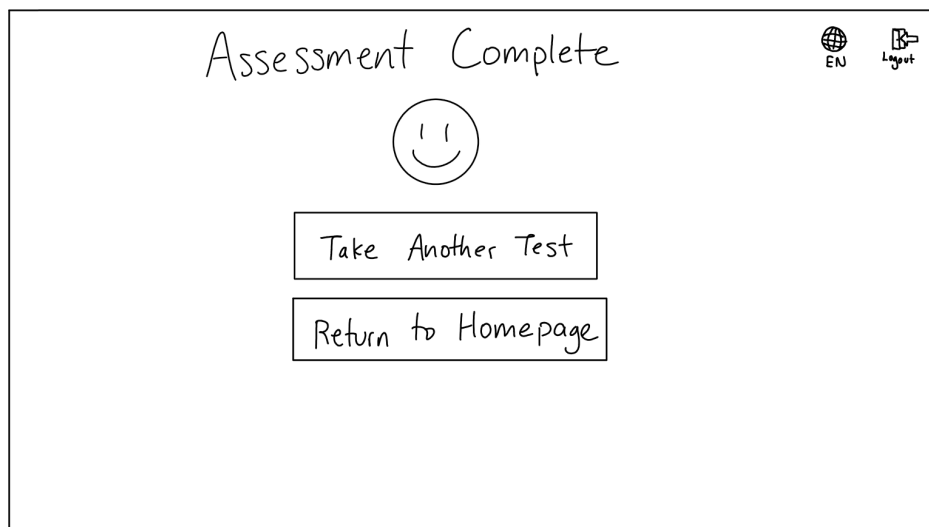


Figure 7: Sketch of Assessment Completion Page

11 Design of Communication Protocols

[If appropriate —SS]

12 Timeline

[Schedule of tasks and who is responsible —SS]

[You can point to GitHub if this information is included there —SS]

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.