Module Guide for ImgBeamer

Joachim de Fourestier

March 22, 2023

1 Revision History

Date	Version	Notes
2023/03/05	0.1.0	Creation
2023/03/10	0.1.1	Add anticipated changes
2023/03/12	0.1.2	Add unlikely changes
	0.1.3	Add the module hierarchy and Behaviour-Hiding module descriptions
2023/03/18	0.1.4	Rework modules and graph, update traceability
2023/03/19	0.1.5	Some minor tweaks, and update graph
2023/03/22	0.1.6	Clarify AC12 and UC1.
	0.1.7	Resolve some minor issues for clarity.

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
BLT	Bit Block Transfer [7]
CSS	Cascading Style Sheets
DAG	Directed Acyclic Graph
DOM	Document Object Model
GUI	Graphical User Interface
HTML	HyperText Markup Language
ImgBeamer	SEM image formation demo tool
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
RBGA	Red-Blue-Green-Alpha (pixel components)
SRS	Software Requirements Specification
UI	User Interface
UC	Unlikely Change
UX	User Experience

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 [6]. We advocate a decomposition based on the principle of information hiding [4]. 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. [6], 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 [6]. 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 [1]. 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 adopted here is called design for change.

AC1: The specific hardware on which the software is running.

AC2: The format of the initial input (ground truth) image data.

AC3: The format of the input parameters.

AC4: The constraints on the input parameters.

AC5: The format of the final output data.

AC6: The constraints on the output results.

AC7: How the overall flow and control of the calculations (for image rendering) are orchestrated.

AC8: The display of information on the visualizations and resulting images, meaning in terms what should (considered useful) be shown to the user. This information may include overall brightness and contrast values, the number of decimal places, image size, etc.

AC9: The ability to use different image quality metrics.

AC10: The implementation of rendering, compositing, clipping, or blending operations (e.g., “Bit BLT”).

AC11: The calculation of the average pixel or signal (intensity) value sampled by the beam.

AC12: The implementation of graphical user interface, meaning the programming language or UI/UX design (”look and feel”).

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 or HID devices (Input: File and/or Keyboard, On-screen keyboard, Output: File, Memory, and/or Screen).

UC2: The software will not simulate electron beam physics, collision cascades, sample nature, topography, or other electron-sample interactions.

UC3: The beam sampling layout or raster pattern.

UC4: The software must be able to import a groud truth image.

UC5: The software must be able to export the resulting image.

UC6: The software must be able to display a live visualization to the user while the imaging parameters are changed.

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.

M1: Hardware-Hiding Module

M2: Application Control Module

M3: Ground Truth Image Input

M4: Imaging Parameters Input

M5: Spot Profile Input

M6: Image Export

M7: Information and Metrics Display

M8: Ground Truth Visualization

M9: Subregion Visualization

M10: Spot Profile Visualization

M11: Spot Content Visualization

M12: Spot Signal Visualization

M13: Spot Layout Visualization

M14: Sampled Subregion Visualization

M15: Resulting Subregion Visualization

M16: Resulting Image Visualization

M17: Display Control

M18: Graphical User Interface

M19: Drawing Stage / Canvas Module

M20: Image Rendering

M21: Image Metrics Calculation

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.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. [6]. 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. If the entry is *WebBrowser*, this means that the module is provided by the HTML 5 [8] compliant web browser. If the entry is *jQuery*, this means that the module is provided by the jQuery DOM [9] manipulation javascript library [2]. If the entry is *CanvasAPI*, this means that the module is provided by the Canvas API of the HTML 5 living standard [10]. If the entry is *Konva*, this means that the module is provided by the Konva.js HTML5 2d canvas javascript library [3]. *ImgBeamer* means the module will be implemented by the ImgBeamer 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.

Level 1	Level 2	Level 3
Hardware-Hiding Module		
Behaviour-Hiding Module	Application Control	
	Input	Ground Truth Image Input
		Imaging Parameters Input
		Spot Profile Input
	Output	Information and Metrics Display
		Image Export
	Visualization Display	Ground Truth
		Subregion
		Spot Profile
		Spot Content
		Spot Signal
		Spot Layout
		Sampled Subregion
		Resulting Subregion
		Resulting Image
Software Decision Module	Display Control	
	Graphical User Interface	
	Image Manipulation	Drawing Stage / Canvas Module
		Rendering
		Metrics Calculation

Table 1: Module Hierarchy

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

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 Application Control (M2)

Secrets: The algorithm for the overall flow of the program.

Services: Provides the main program.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.2 Ground Truth Image Input (M3)

Secrets: The format, structure, and verification of the input ground truth image data.

Services: Prompts the user for an input ground truth image file. This module reads and converts, and provides the data in the data structure used by ImgBeamer.

Implemented By: CanvasAPI and ImgBeamer

Type of Module: Abstract Object

7.2.3 Imaging Parameters Input (M4)

Secrets: The format and validation of the input imaging parameters (number of rows and columns for the rasterization grid and magnification) data.

Services: Gets and provides the parameters values.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.4 Spot Profile Input (M5)

Secrets: The format, validation, and current values describing the spot profile (shape: normalized height vs. width, rotation/angle).

Services: Gets and provides the spot profile values.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.5 Image Export (M6)

Secrets: The format and structure of the data of the resulting image.

Services: Converts image data to an output image file.

Implemented By: CanvasAPI, Konva, and ImgBeamer

Type of Module: Library

7.2.6 Information and Metrics Display (M7)

Secrets: The display of information on the rendered images and minor calculation algorithms of supplemental/optional information (such as spot eccentricity).

Services: Displays information on the images and parameters such as the magnification, image quality metrics, spot shape information, spot signal pixel values (RBGA), and the drawing rate of resulting image (e.g., rows per milliseconds).

Implemented By: ImgBeamer

Type of Module: Library

7.2.7 Ground Truth Visualization (M8)

Secrets: The drawing stage and how to render the visualization for the ground truth image and subregion bounds.

Services: Draws the Ground Truth Image, a highlighted area representing the subregion area (as drawn by M9), and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.8 Subregion Visualization (M9)

Secrets: The drawing stage and how to render the subregion visualization.

Services: Draws the subregion image (zoomed-in area on the ground truth image) and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.9 Spot Profile Visualization (M10)

Secrets: The drawing stage and how to render the visualization for the spot profile (width, height, and rotation, see SRS [1]).

Services: Draws the spot profile and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.10 Spot Content Visualization (M11)

Secrets: The drawing stage and how to render the visualization for the spot content (sub-region "stenciled" with the spot profile, see SRS [1]).

Services: Draws the spot content and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.11 Spot Signal Visualization (M12)

Secrets: The drawing stage and how to render the visualization for the spot content (a signal or pixel value to represent what has been sampled by the spot or beam, see SRS [1]).

Services: Draws the spot signal and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.12 Spot Layout Visualization (M13)

Secrets: The drawing stage and how to render the visualization for the spot layout.

Services: Draws the spot layout and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.13 Sampled Subregion Visualization (M14)

Secrets: The drawing stage and how to render the visualization for the sampled image content of the subregion.

Services: Draws the sampled subregion (image content "stenciled" with the spot layout) and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.14 Resulting Subregion Visualization (M15)

Secrets: The drawing stage and how to render the visualization for the resulting subregion.

Services: Draws the resulting (resampled) subregion image and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.2.15 Resulting Image Visualization (M16)

Secrets: The drawing stage and how to render the resulting image.

Services: Draws the resulting (resampled full) image and returns an update function (reference/pointer) to call when the drawing should be updated/redrawn.

Implemented By: ImgBeamer

Type of Module: Abstract Object

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 Display Control (M17)

Secrets: The drawing stages (or canvases), the information required for each (object/function references, update and redraw events, drawing rates), and the connection to the rendering module (M20).

Services: Initializes and manages the display objects and is responsible for the image rendering and drawing.

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.3.2 Graphical User Interface (M18)

Secrets: The HTML DOM Manipulation, CSS rules, user interaction event handling, user input controls, states, data formats (such as text-boxes, buttons, file dialogs), visual layout and styling (e.g., colors or sizing).

Services: Sets up a visual interface for the user to see and interact with, handles user and GUI control events, updates the visual elements representing the state of the application, and sends messages back to the application control (M2) to update the application state accordingly.

Implemented By: OS, WebBrowser, jQuery, and ImgBeamer

Type of Module: Abstract Object

7.3.3 Drawing Stage / Canvas Module (M19)

Secrets: The format and structure of the image pixel data, layering / draw order, compositing algorithms, frame buffers, and other graphical operations.

Services: Provides the means to draw and manipulate computer graphics with the concepts of layers and geometry objects (such as rectangles, ellipses, lines, etc.).

Implemented By: Konva and CanvasAPI

Type of Module: Abstract Data Type

7.3.4 Image Rendering (M20)

Secrets: The image and pixel by pixel format (RGBA values) and data manipulations, calculations, and resampling algorithms.

Services: Provides the image rendering methods to create images needed for the visualization modules (M8, M9, M10, M11, M12, M13, M14, M15, M16).

Implemented By: ImgBeamer

Type of Module: Abstract Object

7.3.5 Image Metrics Calculation (M21)

Secrets: The algorithm, criteria, and validation for processing and comparing two images.

Services: Compares two images (ideally, one should be a reference or ground truth image and the other should be an image to compare to) and produces a score on the similarity or quality of the compared image with respect to reference image.

Implemented By: ImgBeamer

Type of Module: Abstract Object

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	M1, M3, M18
R2	M1, M5, M18
R3	M1, M4, M18
R4	M8, M9, M17, M18, M19, M20
R5	M13, M10 M17, M18, M19, M20
R6	M6, M11, M12, M14, M15, M16, M17 M18, M19, M20
R7	M7, M18, M19, M20, M21

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M3, M18, M19
AC3	M4, M18, M19
AC4	M4, M18
AC5	M6, M18, M19
AC6	M6, M18
AC7	M2, M20, M21
AC8	M7, M18
AC9	M18, M21
AC10	M17, M20
AC11	M20
AC12	M18

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas [5] 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.

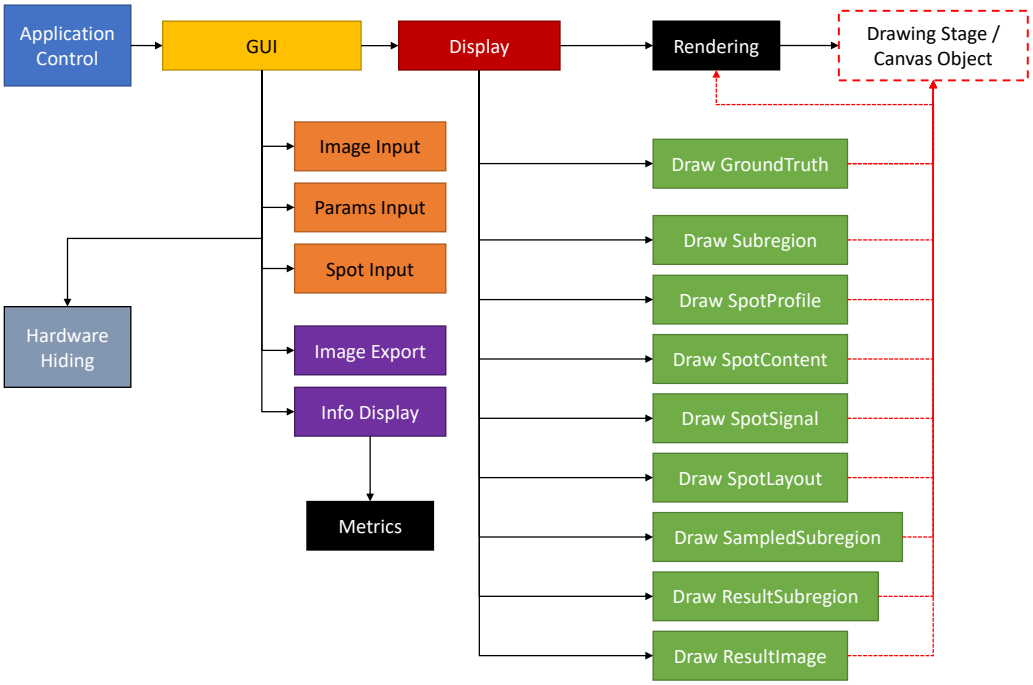

Figure 1: Use hierarchy among modules

References

[1] J. de Fourestier. Software requirements specification for imgbeamer: Scanning electron microscope image formation, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/SRS/SRS.pdf.

[2] OpenJS Foundation and contributors. jQuery: The write less, do more, javascript library, 2023. URL https://jquery.com/.

[3] Anton Lavrenov. Konva.js - JavaScript 2d canvas library, December 2021. URL https://konvajs.org/index.html.

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

[5] 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.

[6] 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.

[7] Rob Pike, Leo Guibas, and Dan Ingalls. Bitmap Graphics SIGGRAPH'84 Course Notes - Technical Memorandum. page 68. AT&T Bell Laboratories, May 1984. URL https://pdos.csail.mit.edu/%7Ersc/pike84bitblt.pdf.

[8] W3C. HTML living standard - specification, Mar 2023. URL https://html.spec.whatwg.org/multipage/introduction.html.

[9] W3C. DOM living standard, Mar 2023. URL https://dom.spec.whatwg.org/.

[10] W3C. HTML living standard, the canvas element, Mar 2023. URL https://html.spec.whatwg.org/multipage/canvas.html.