Dr. UML

June 11, 2025

CSIE IV 蕭耕宏110590005
CSIE IV 黃冠鈞110590028
CSIE IV 張庭瑋110590035
CSIE IV 吳宥駒110590066
Homework #7

1 Change History

1.1 HW1

- Add section Problem statement.
- Add section Development Language.

1.2 HW2

- Add section 3-8.
- Change section "Development Language" to "Software Environments" as demanded.
- Change the project name to "Dr. UML".

1.3 HW3

- Add section 9(Domain Model).
- Merge UC7(Host Session) and UC8(Join Session).
- Change Acronyms for System Features into FEA.

1.4 HW4

- Extract UC01-extension *b to UC09.
- Refine pre-condition of UC01.
- Use case
  - Remove extension "3.a If User connects Gadget to nothing, a default Gadget will be generated automatically." in UC01.
- Domain Model
  - Add Timer and Verifier and their associates.
  - Add zIndex attribute to Components.
- Add Logical architecture
- Add System Sequence Diagrams with GRASP Patterns
- Add Design Class Model

1.5 HW6

- Figure Updated
  - Domain model
  - Domain model with associations
  - Domain model with associations and attributes
- Figure Enlarged
  - Domain model with associations
  - Domain model with associations and attributes
  - Design class diagram
  - Implement design class diagram
- Use case
  - UC01
    - In Main Success Scenario step 1, Change the wording from "drag" to "select".
    - Replace "1-4 steps can be repeated" in the Main Success Scenario with "User can skip any step from 1 to 4" in the Extension.
    - Remove extension 4.d.
  - UC08
    - Add System feedbacks(step 5) in main success scenario.
    - Fix Misc typos.
- Domain Model
  - Remove TextFiled, ImageFormat, Filename, Field, and Component from bad classes.

1.6 HW7

- Update Figure
  - Remove up-down arrows from the Domain model diagram.
  - Sequence diagram
    - Remove UI from all sequence diagrams.
    - Add UMLProject to UpdateProperty by controller pattern.
  - Revise Design Class Diagram based on midterm feedback
- Use case
  - Add missing "Technology and Data Variations List" and "Special Requirements" sections for each use case.

- Add missing classes to identified classes
- Add "Comparison with design and implementation class" table
- Add "Summary of implementation class/method changed" table
- Update lines of code

2 Problem statement

There are several tools available for creating UML diagrams on the internet but many of them come with paid subscriptions or limitations that make them less accessible. Moreover, they often resort to creating poorly formatted documents due to the lack of affordable, high-quality options. As the result, we propose Dr. UML.

Dr. UML is an innovative collaborative platform designed for software developers, system architects, and students who need to efficiently create and manage Unified Modeling Language (UML) diagrams.

The tool meets the pressing need for collaborative designing by allowing teams to work together simultaneously, regardless of their physical location. In addition, it offers real-time updates and integrated communication features. Dr. UML will be used primarily in design meetings, brainstorming sessions, and technical workshops where immediate visual feedback is essential. It is needed when precise and dynamic visual representation of complex systems is required to align team understanding and streamline development processes.

Dr. UML integrates a robust set of customizable UML elements with drag-and-drop functionality and real-time collaboration. This not only enhances the creative aspects of system design but also ensures that technical requirements are met with precision and clarity. The platform's intuitive design and collaborative capabilities make it an essential tool for modern software development teams, system architects, and students aiming to create high-quality UML diagrams efficiently.

3 Summary of System Features

- FEA01: Create a UML diagram file.
- FEA02: Edit a UML diagram(draw, edit Component properties, copy and paste Components)
- FEA03: Save and load progress.
- FEA04: Export UML diagram into image formats.
- FEA05: Start a online Session, allowing other Users to join.
- FEA06: Connect to online Sessions, edit UML with other users simultaneously.

- FEA07: Real-time chatroom in a online Session.

4 Use Case Diagram

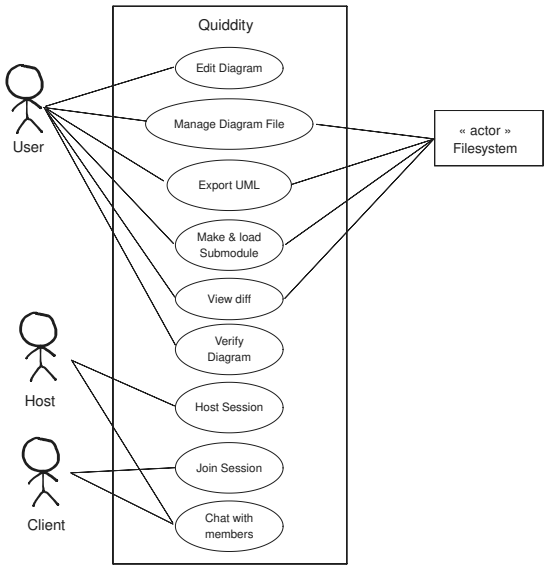


Figure 1: Use case diagram

5 Use Cases

5.1 UC01: Edit UML

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: User
- **Stakeholders and Interests**:
  - **User**: Wants to create and connect Components.
- **Preconditions**:
  - User has opened a UML Project.
  - At least one UML Diagram is opened.
  - UML Editing Canvas and Toolbox are loaded.
- **Success Guarantee**: UML is edited according to the User's specifications.
- **Main Success Scenario**:

1. User selects Gadgets from Toolbox.
2. User edits the Gadgets.
3. User establishes connections between Gadgets via Associations.
4. User modifies the Associations as needed.

- **Extensions**:
  - *a In the event of System failure, User restarts System. UML will revert to the last successfully saved state.
  - *b When editing text, User may design their text as desired(See UC09)
  - *c User may select Gadgets within the canvas.
  - *d When User drags Gadget with multiple Associations, System will automatically update them.
  - *e User can determine the layering order when Gadgets overlap.
  - *f User may copy and paste Components.
    1. Copying or pasting Associations will also include connected Gadgets.
  - *g User may undo or redo actions.
  - *h User can skip any step from 1 to 4
  - 1.a Edit fails if Gadget is dragged to an invalid location.
  - 1.b User can also import an available Submodule in the current Project.
  - 2.a Different types of Gadgets will have distinct Fields available for editing.
  - 2.b Edited Gadgets will automatically scale to fit the changes.
  - 2.c User can modify the color of a Gadget. The color will apply to Gadget's background.
  - 2.d User can move the Gadget as long as the destination is valid. The associations will be updated automatically.
  - 3.a Deleting a Gadget will also remove the associated connections.
  - 3.b Self-Associations are allowed.
  - 4.a User may change the type of Association.
  - 4.b User may add, remove, and move text Fields in Association.
  - 4.d User can modify the path of an Association.
- **Special Requirements**:
  - When multiple users are editing, Nounintended behavior should occur.
- Technology and Data Variations List:
  - 1.a Based on the diagram type, the available Gadgets will vary (e.g., class Gadgets for a class diagram).
  - 2.a User sets and modifies Attributes of a Gadget by mouse or keyboard input.

  - 3.a Based on the diagram and related-Gadget type, the available Association will vary from valid UML components (e.g., Composition for an Association linking two class Gadgets).
  - 4.a User sets and modifies Attributes of an Association by mouse or keyboard input.

- **Frequency of Occurrence**: Often
- **Open Issues**: None specified

5.2 UC02: Manage UML

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: User
- **Stakeholders and Interests**:
  - **User**: Wants to create, save, export, delete a Diagram.
  - **Filesystem**: Requires the file to be read/saved properly.
- **Preconditions**:
  - System is up.
- **Success Guarantee**:
  - User manage Project and Diagram.
- **Main Success Scenario**:
  1. User selects existing Project.
  2. User selects Diagram in the Project.
  3. User edit Diagram as described in UC1.
  4. User exports Diagram to Filesystem.
  5. User deletes the Diagram.
- **Extensions**:
  - *a In both Project and Diagram, the time of last edit is recorded and can be viewed by User.
  - *b If System fails to manage (load, save, and export) Project or Diagram, User will be prompted to either retry the operation or abort.
  - 1.a User may choose to create a new Project.
  - 1.b User can modify the name of the Project.
  - 2.a User may choose to create a new Diagram.

  - 3.a User can modify the type, background color, filename of the Diagram.
  - 4.b When exporting Diagram, User may select one of the supported UML formats (for future verification purposes).

- Special Requirements
  - The system should allow users to manage diagrams efficiently, supporting batch operations such as removing or saving multiple diagrams at once.

- Technology and Data Variations List
  - 4.a The saved file is in a json5-based format.

- **Frequency of Occurrence**: Occasionally
- **Open Issues**: None specified

5.3 UC03: Export UML

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: User
- **Stakeholders and Interests**:
  - **User**: Wants to export UML-Project to image formats with desired name and extension.
  - **Filesystem**: Requires the exported image to be saved properly.
- **Preconditions**:
  - System is up.
  - User has opened a UML project.
- **Success Guarantee**: Exported image is saved to Filesystem.
- **Main Success Scenario**:
  1. User starts exporting current diagram.
  2. User selects a supported image format and specifies a filename.
  3. System saves the exported image to Filesystem.
- **Extensions**:
  - 3.a If Filesystem fails to save the exported image, User will be prompted to either retry the operation or abort.
- Special Requirements

  - System is expected to generate the exported image within 5 seconds.
  - System should allow users to export diagrams efficiently, supporting multiple diagram exports.

- Technology and Data Variations List
  - 2.a Supported formats:
    * JPEG
    * PNG
    * SVG
    * WebP

- **Frequency of Occurrence**: Occasionally
- **Open Issues**: None specified

5.4 UC04: Manage Submodule

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: User
- **Stakeholders and Interests**:
  - **User**
    * Imports Submodule to UML Project, which can be included it to any UML Diagram in the Project.
    * Removes any imported Submodule(s) from UML Project
    * Exports a part of components to a Submodule and save it to Filesystem.
  - **Filesystem**: Wants to save and load a Submodule file.
- **Preconditions**:
  - System is up.
  - User has opened a UML-Project.
  - For exporting Submodule, User is also required to open a UML Diagram and have a number of Component drawn.
- **Success Guarantee**:
  - Submodule is exported on Filesystem.
  - Submodule is imported to UML-Project.
- **Main Success Scenario**:
  1. User imports a Submodule file from Filesystem.

2. From the Project menu, User selects an outdated Submodule and removes it.
3. After drawing a number of Components, User selects and exports them as a Submodule.
4. The Submodule file is saved to Filesystem.

- **Extensions**:
  - 1.a If Filesystem fails to import Submodule, User will be prompted to either retry the operation or abort.
  - 3.a If nothing is selected, User cannot export it as Submodule.
  - 3.b Optionally, the saved Submodule may have empty Fields.
  - 4.a If Filesystem fails to save Submodule file, User will be prompted to either retry the operation or abort.
- Special Requirements
  - System is expected to export a Submodule within 5 seconds.
  - System can distinguish a saved Submodule file from a Diagram one (See UC02).
- Technology and Data Variations List
  - 4.a The saved file is in a json5-based format.
- **Frequency of Occurrence**: Sometimes
- **Open Issues**: None specified

5.5 UC05: Verify UML

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: User
- **Stakeholders and Interests**:
  - **User**: Wants to verify the correctness of UML.
- **Preconditions**:
  - System is up.
  - A UML is available.
- **Success Guarantee**: System verifies and displays the correctness of the UML diagram.
- **Main Success Scenario**:
  1. User opens a UML project.

2. User instructs System to verify the UML.
3. System checks the correctness of the UML.
4. System displays the verification results.

- **Extensions**:
  - 1.a If System fails to open the UML file, User will be prompted to either retry the operation or abort.
  - 3.a If System fails to verify the UML, User will be prompted to either retry the operation or abort.
  - 3.b System verifies the UML by diagram type.
    1. If the UML type verification is unavailable, inform the User.
  - 4.a If System fails to display result, User will be prompted to either retry the operation or abort.
  - 4.b System informs User of verification results, which may include:
    1. All clear, UML is correct in terms of diagram type.
    2. Warnings, UML is mostly free of syntax errors, but contains some bad smells™.
    3. Invalid, UML contains critical errors.
- **Special Requirements**:
  - Error messages should be user-friendly and provide actionable insights.
- **Technology**:
- **Frequency of Occurrence**: Sometimes
- **Open Issues**: None specified

5.6 UC06: Join Session

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: Client
- **Stakeholders and Interests**:
  - **Host**: Wants Client to join current opened project.
  - **Client**: Wants to connect to an existing project.
- **Preconditions**:
  - System is up.
  - A stable connection exists between Host and Client.

  - Host opened a UML project.

- **Success Guarantee**: Connection is established and maintained, UML is synced, and Noconflicts occur.

- **Main Success Scenario**:

  1. Host starts a Session.
  2. Client connects to the Session.
  3. Both Host and Client edit the UML as described in UC1.
  4. Step 3 is repeated until UML is complete.
  5. Host ends the Session.

- **Extensions**:

  - *a User may opens a session for a project. Assuming the role of Host.
  - *b At any time, a Component can only be edited by exactly one User.
  - 2.a If a connection error occurs, notify Client of the issue encountered.
  - 3-4.a Host can remove any Client from the Session.
  - 3-4.b If an action fails to send, the System retries the action.
    1. If retry limit is reached, System will remove that Client from current Session.
  - 3-4.c Clients may redo and undo their own edits.
  - 5.a Session ends if Host closes it, whether intentionally or unintentionally.

- **Special Requirements**:

  - Only exactly one User is allowed to edit a Component at a time, ensuring there won't be a racing condition.
  - The error message of a failed connection should be easy to read.

- **Technology and Data Variations List**:

  - 1-2.a System uses TCP sockets to establish and connect Sessions.
  - 1-4.a A well-defined protocol should be followed for communication among Users.

- **Frequency of Occurrence**: Sometimes

- **Open Issues**: Undo/redo conflicts

5.7 UC07: Chat with Members

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: User
- **Stakeholders and Interests**:
  - **User**: Wants to communicate with other users in Session via text messages.
- **Preconditions**:
  - System is up.
  - Users have joined a Session.
- **Success Guarantee**:
  - Users in Session can communicate with each other with text messages.
  - Users in Session can view chat history.
- **Main Success Scenario**:
  1. User opens the chatroom for the current Session.
  2. User views chat history.
  3. User types then sends messages.
- **Extensions**:
  - *a The time of each sent message is recorded and can be viewed by User.
  - 1.a If a new Session is created, System creates a new chatroom with Nomessages.
  - 2.a If System fails to load the chatroom, it will attempt to retry.
    1. If retry limit is reached, notify User of the issue encountered. User will not be able to view the previous messages.
  - 3.a If System fails to send User's message, it prompts the User to either remove or resend the message.
- **Special Requirements**:
  - Every User in the Session sees all Messages even it was sent before they join.
  - The error message of a failed connection should be easy to read.
- **Technology and Data Variations List**:
  - 1-3.a System uses TCP sockets for Chatroom.
  - 1-3.b A well-defined protocol should be followed for communication among Users.
- **Frequency of Occurrence**: Sometimes
- **Open Issues**: None specified

5.8 UC08: Edit Attribute of a Component

- **Scope**: Dr. UML
- **Level**: User goal
- **Primary Actor**: User
- **Stakeholders and Interests**:
  - **User**: Wants to edit Attributes of a existed Component.
- **Preconditions**:
  - A Component is created.
- **Success Guarantee**:
  - Attributes is edited according to the User's specifications.
- **Main Success Scenario**:
  1. User selects a Component.
  2. User adds a new Attribute into the Component.
  3. User selects an Attribute.
  4. User changes the size, font, type of the texts.
  5. System updates according to last step.
  6. Steps 1–5 are repeated in any order until the editing is complete.
- **Extensions**:
  - *a If a Component or Attribute is deselected during editing, System saves its current state.
  - 2.a User may select an exited Attribute instead of creating a new one.
  - 3.a After an Attribute is selected, User may also remove it from the Component.
  - 4.a Text type has these options:
    - Bold
    - Italic
    - Underline
- **Special Requirements**:
  - The error message of a failed action should be easy to read.
  - A backup font should be there whenever System cannot find the User-specified font.
- **Technology and Data Variations List**:
  - Supported font formats are .ttf and .woff2.
- **Frequency of Occurrence**: Often
- **Open Issues**: None specified

6 Non-functional Requirements and Constraints

6.1 Performance Requirements

- **NFR1: Response Time**: Operations (e.g., dragging components, editing text, collaboration) should respond within **1s**.
- **NFR2: Concurrent Users**: The system should support at least **4 users** editing a UML diagram in real-time.
- **NFR3: File Handling**: Loading or saving UML files should take **less than 3 seconds** (for diagrams with 100+ elements).
- **NFR4: Export Speed**: UML diagrams should be converted to PNG, JPEG, SVG, or WebP formats within **5 seconds**.
- **NFR5: Network Efficiency**: Collaboration mode should minimize data traffic and prioritize critical updates to reduce bandwidth usage.

6.2 Usability Requirements

- **UR1: User-friendly Interface**: Users should be able to understand basic operations within **20 minutes**.
- **UR2: Undo/Redo**: The system should support **at least 50 levels** of undo and redo history.
- **UR3: Accessibility**: Keyboard shortcuts should be provided to enhance usability.
- **UR4: Collaboration Features**: Users should see real-time updates and be able to communicate via chat or annotations.

6.3 Reliability & Availability Requirements

- **RAR1: Uptime**: The system should maintain **99.9% availability**.
- **RAR2: Autosave**: Progress should be automatically saved every **30 seconds**.
- **RAR3: Error Handling**: The system should handle network failures gracefully, allowing users to reconnect without data loss.
- **RAR4: Data Consistency**: All users should see the same UML diagram state in collaborative mode.

7 Glossary

- Submodule: a part of UML diagram, it can be imported into other UML diagram
- Gadget: a block contains text Fields

- Toolbox: A bar containing Components, allowing Users to add them to the canvas.
- Association: connection between two Gadgets
- Field: the place where User can insert text
- Session: A shared project allowing other Users to collaborate.
- Component: Gadget or Association o the canvas.
- Host: A User who has started a Session.
- Client: A User who has joined a Session.
- User: A general term that refers to any participant, including both the Host and Client.
- Attribute: Property of a Components.
- Attribute Tree: A tree-like UI listing Attributes of a component.

8 Software Environments

Golang.

9 Domain model

9.1 Domain Class Diagram Showing Only Concepts

9.1.1 Classes Identified

The nouns listed below are found in the use case.

Table 1: Classes Identified

*User	System	*UMLDiagram	Component	*Gadget
*Association	TextField	Path	UMLFile	Filesystem
*UMLProject	ImageFormat	Filename	*Submodule	Field
DiagramType	VerificationResult	Host	Client	*Session
Project	Connection	*Chatroom	*Message	ChatHistory
TextStyle	*Timer	*Verifier	fontSize	isBold
isItalic	isUnderline	position	Toolbox	Canvas

Note: Classes marked with asterisk(*) are good classes.

9.1.2 Bad Classes

Table 2: Categorization of Terms

Attributes	Abstract Concepts	Implementation Construction	Too UI
fontSize	System	UMLFile	Toolbox
isBold	Host	Filesystem	Canvas
isItalic	Client	VerificationResult	TextStyle
isUnderline	Project	ChatHistory	
DiagramType		Connection	
position			

9.1.3 Good Classes

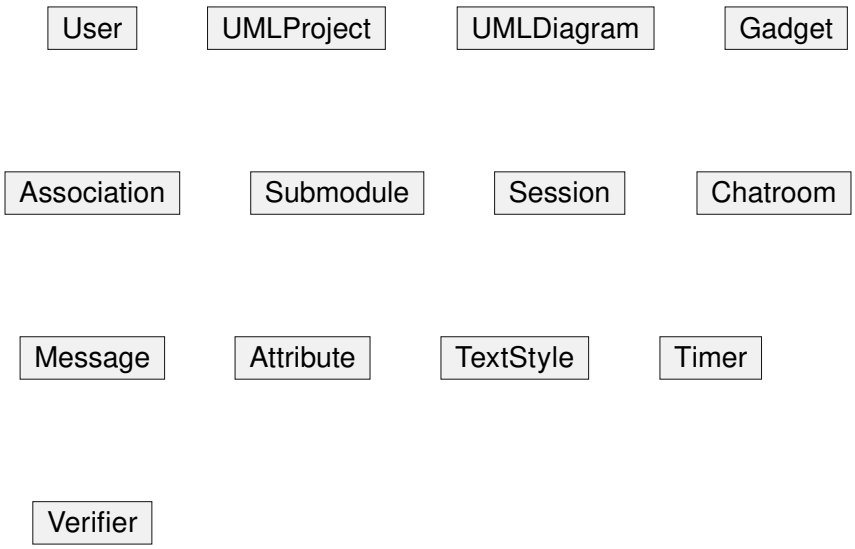


Figure 2: Domain class diagram showing only concepts

9.2 Add Associations

- One User hosts or joins several Sessions.
- One User manages one UMLProject.
- One UMLProject manages one UMLDiagram.
- One UMLDiagram consists of several Submodules.
- One Submodule consists of several Gadgets.
- One Gadget associates with one or two Associations.
- One Association contains several Attributes.

- One Attribute is described by one TextStyle.
- One Session connects to one Chatroom.
- One Chatroom contains several Messages.
- One User sends several Messages.

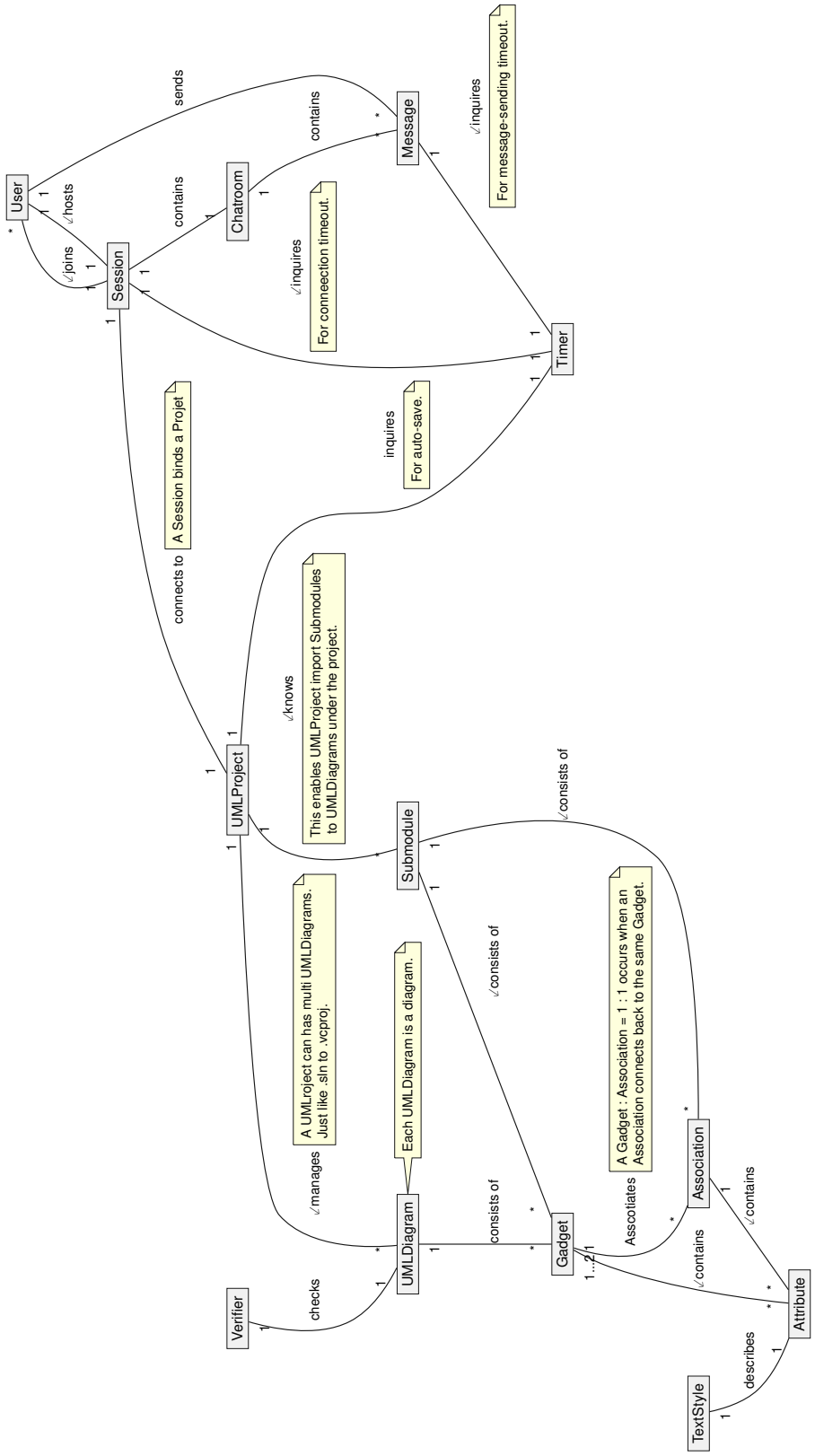


Figure 3: Domain class diagram with associations added

9.3 Add Attributes

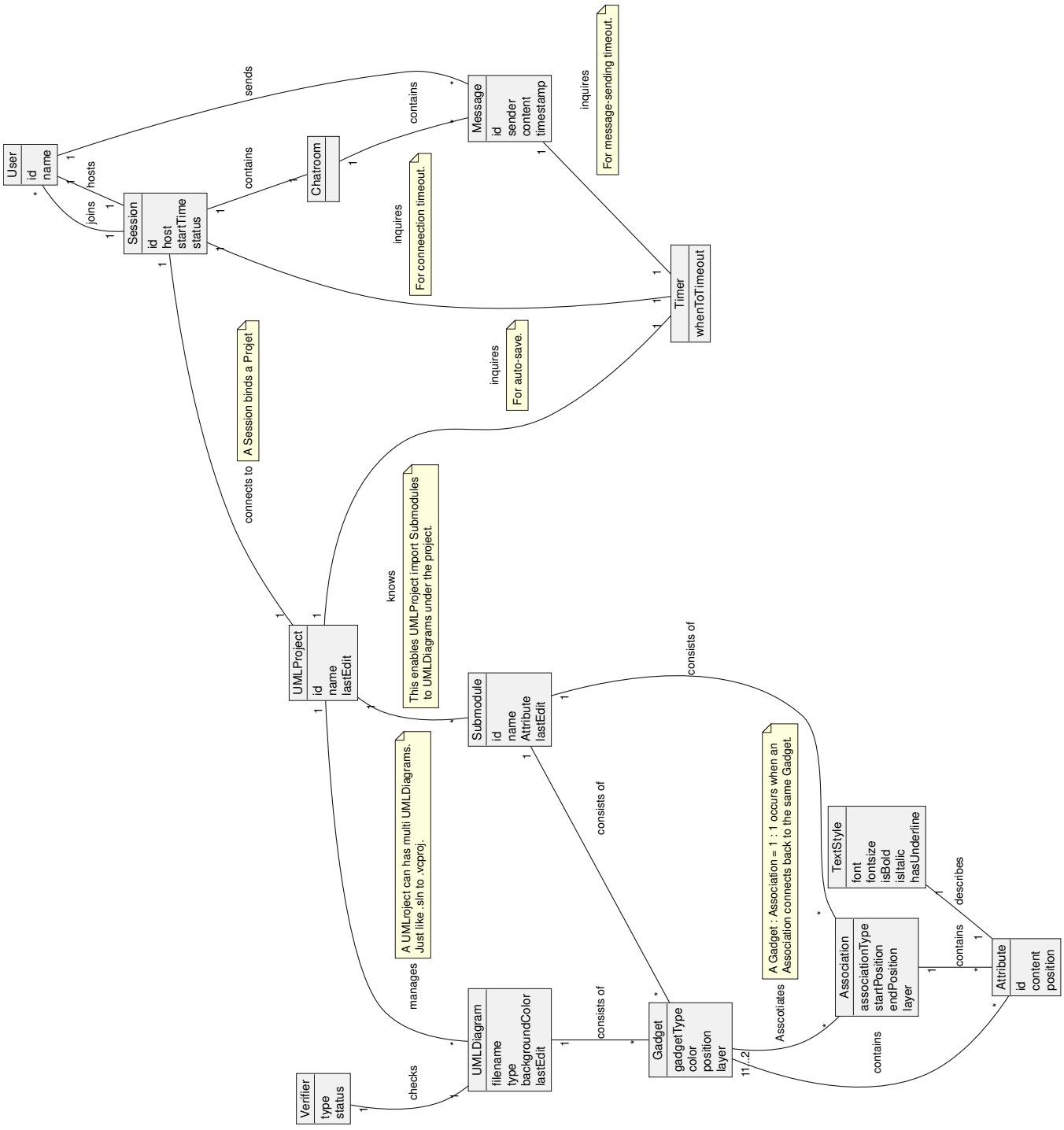


Figure 4: Domain class diagram with attributes added.

10 Logical Architecture

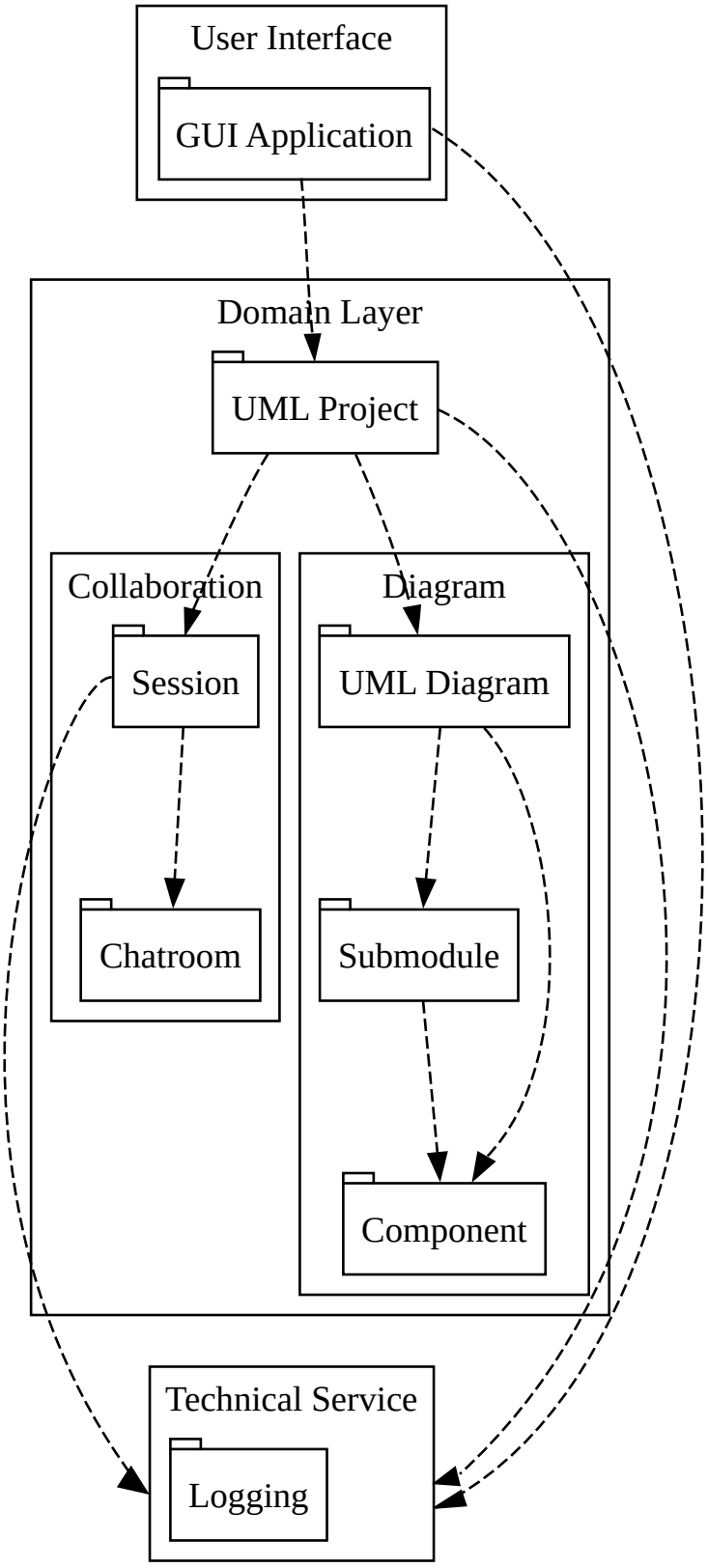


Figure 5: Logical architecture

11 System Sequence Diagrams with GRASP Patterns

We pick UC1 as our most significant use-case. Since UC1 is made of series of system events, we decide to provide SSDs for each one of them.

11.1 main SSD

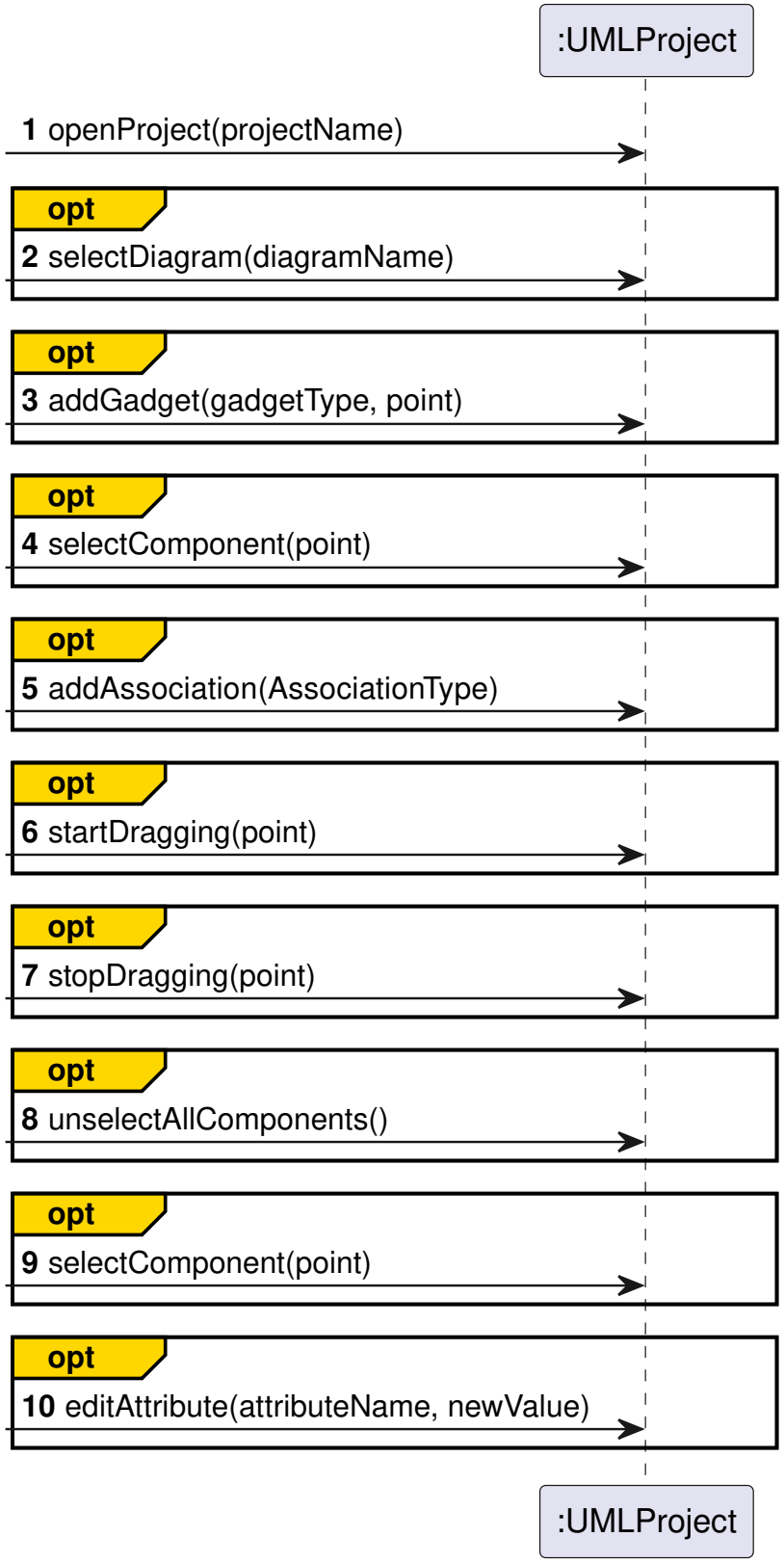


Figure 6: main SSD

11.2 addAssociationToDiagram

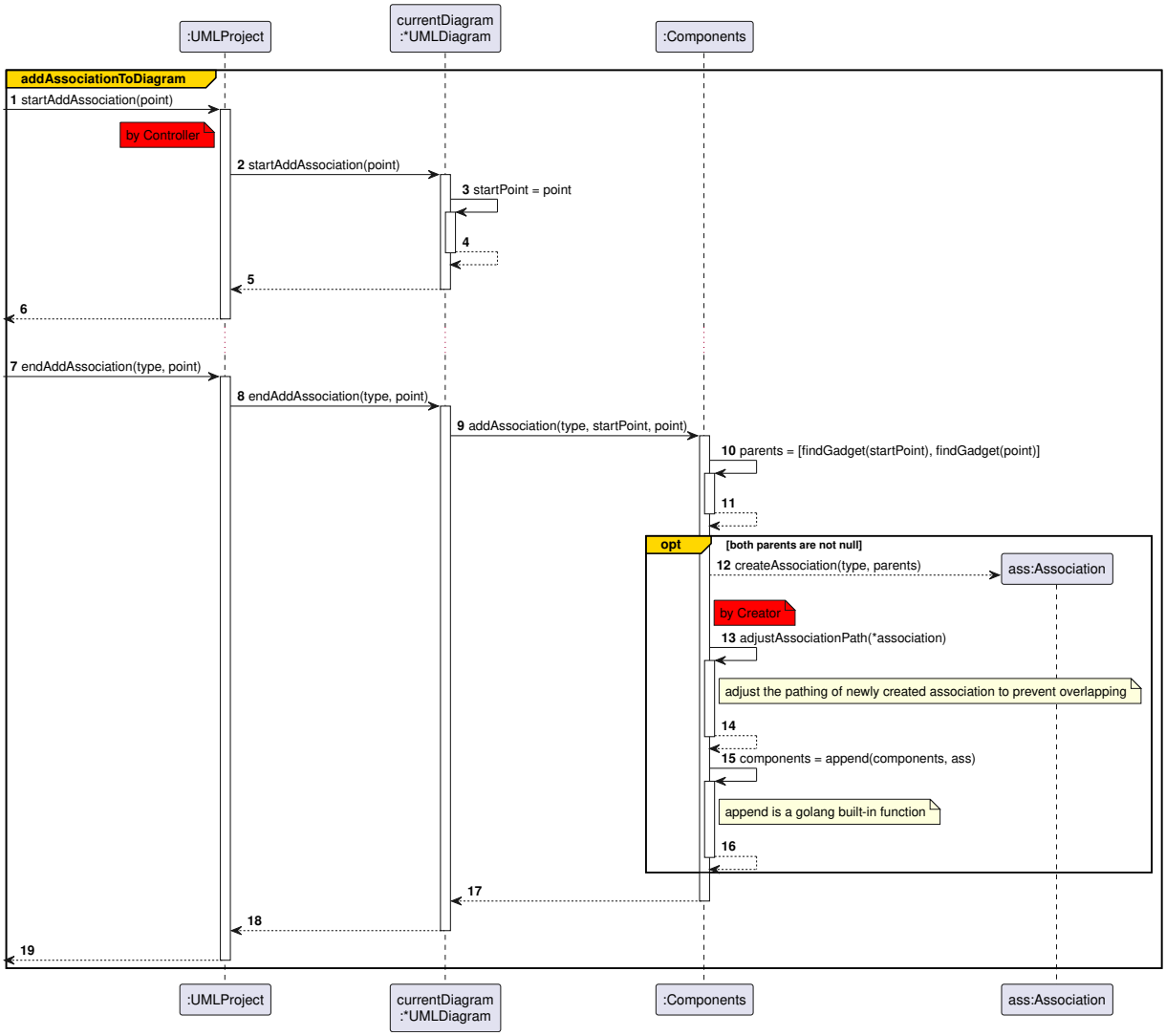


Figure 7: addAssociationToDiagram

11.3 addGadgetToDiagram

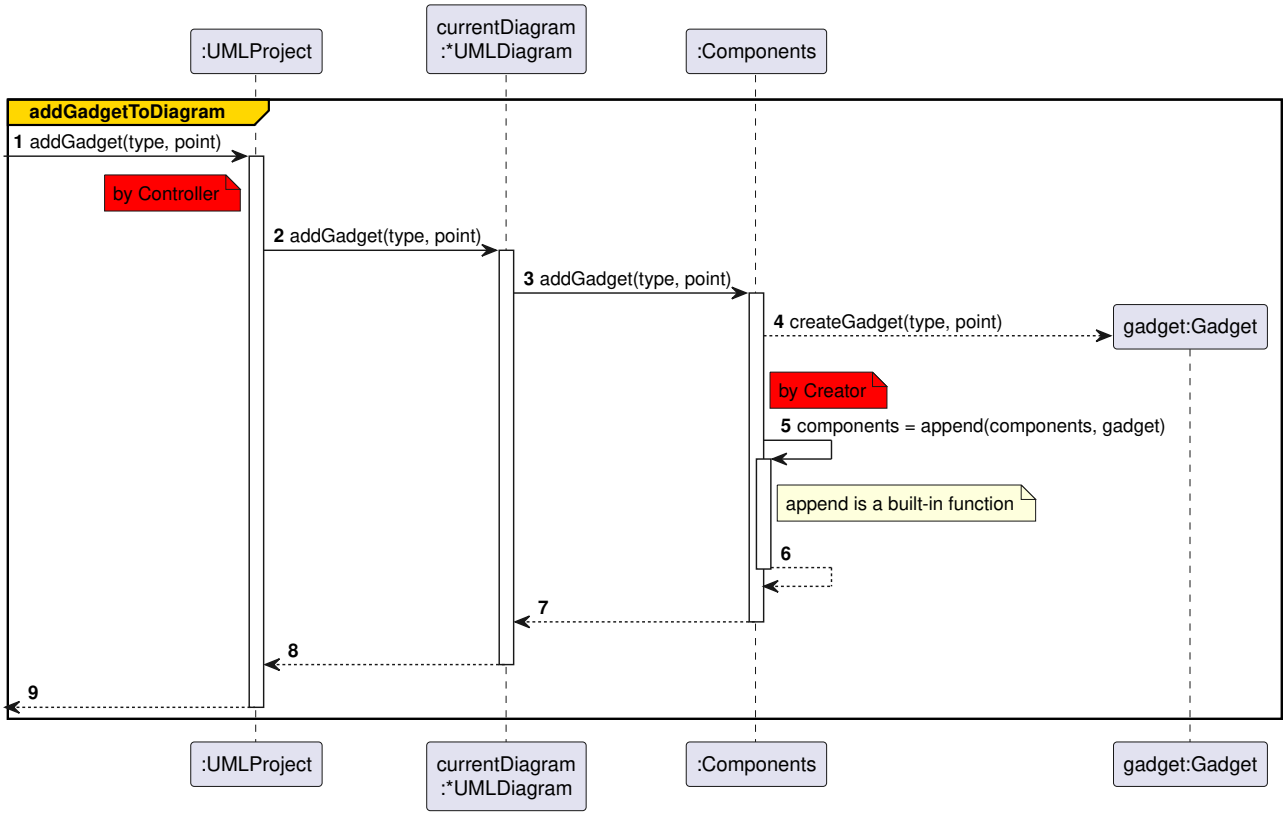


Figure 8: addGadgetToDiagram

11.4 copyComponents

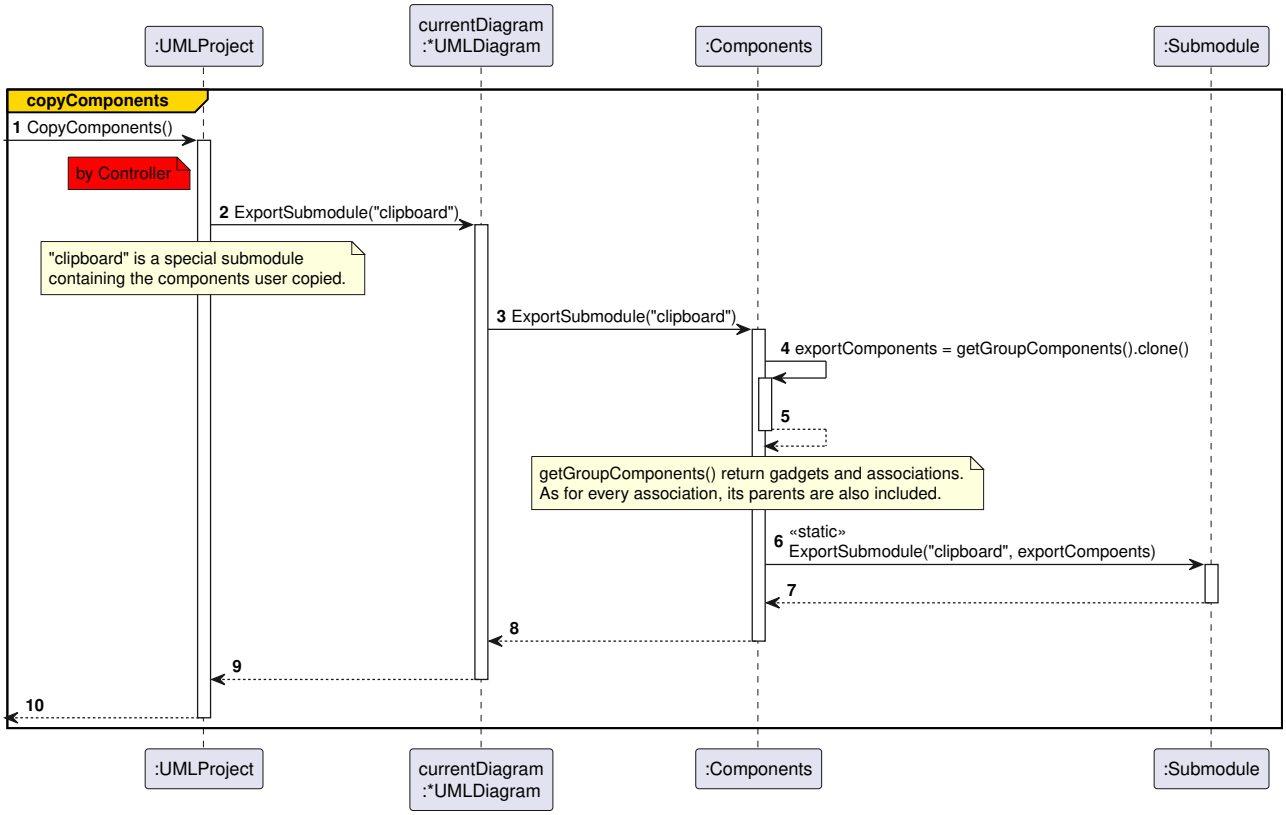


Figure 9: copyComponents

11.5 deleteComponent

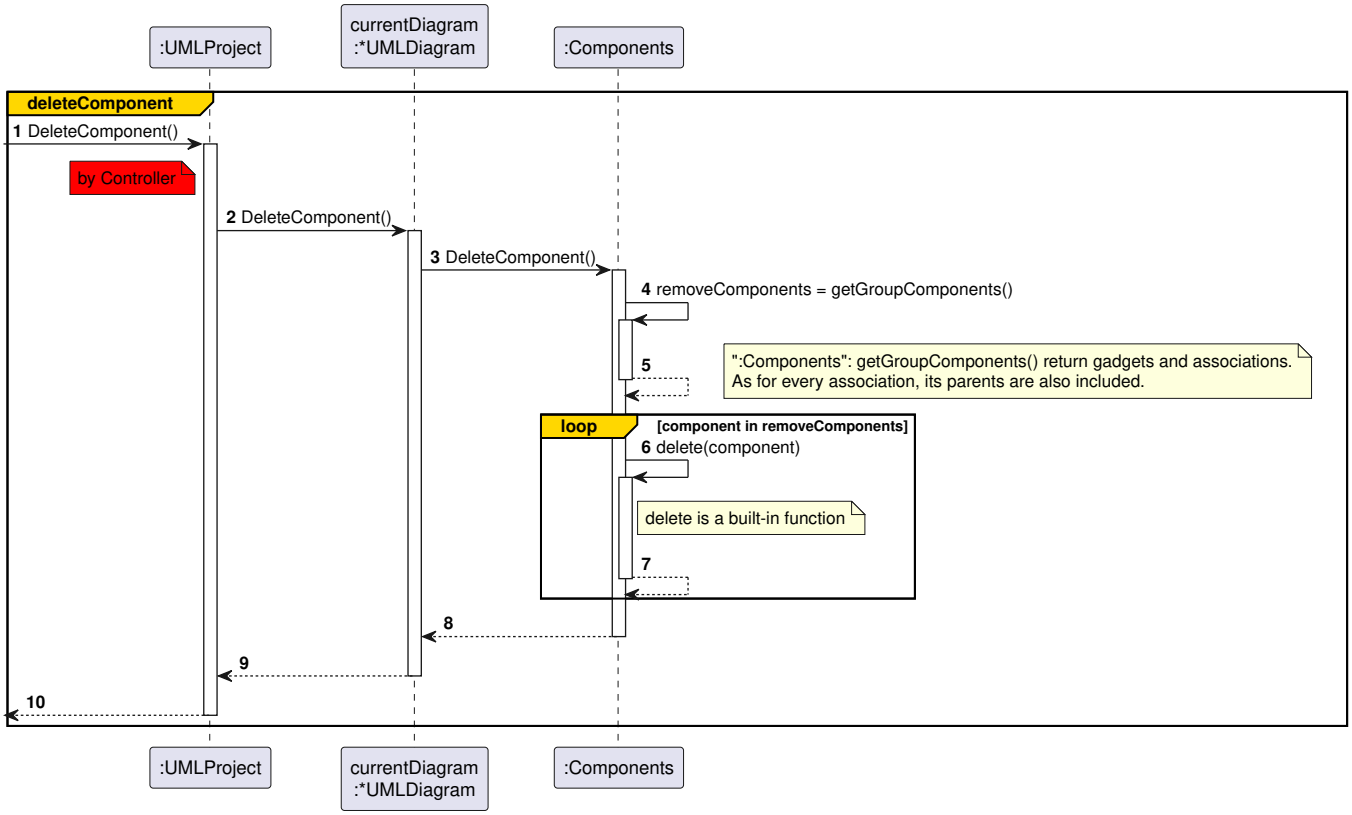


Figure 10: deleteComponent

11.6 drawAll

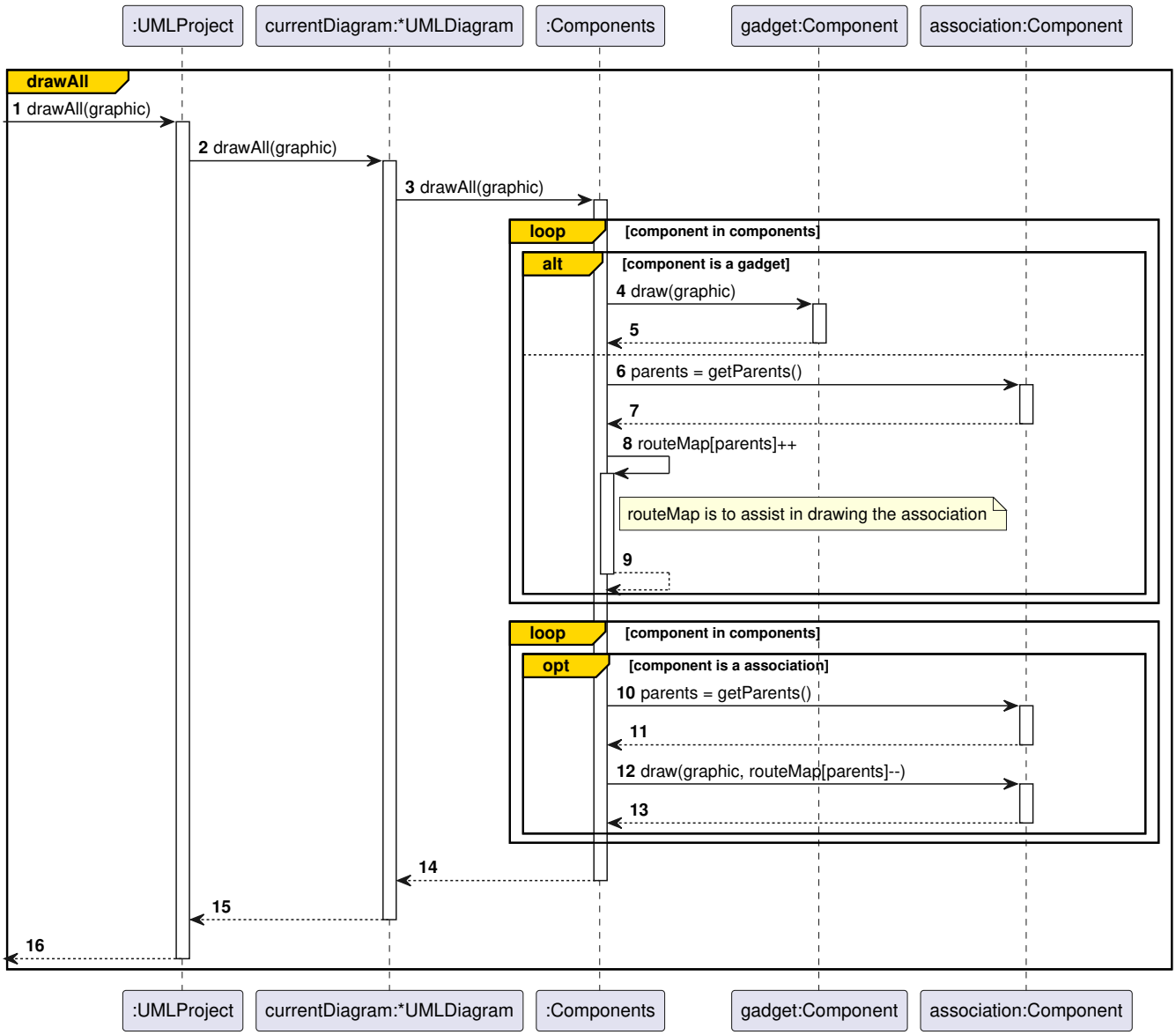


Figure 11: drawAll

11.7 importSubmodule

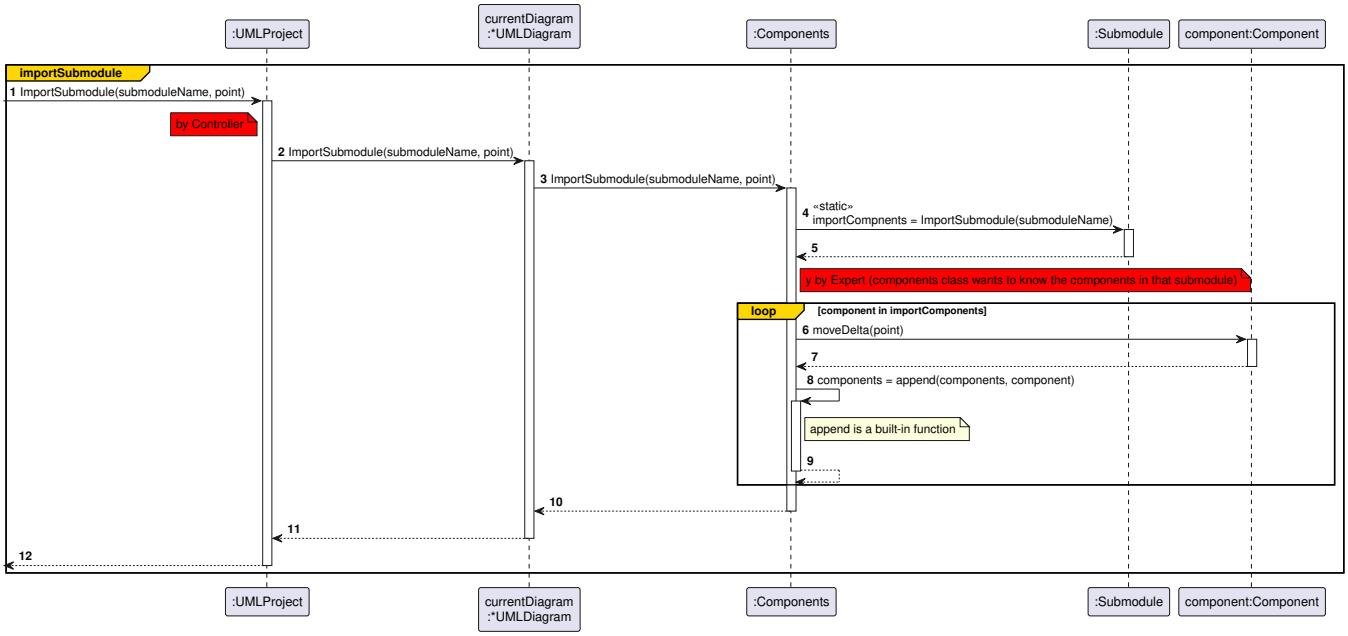


Figure 12: importSubmodule

11.8 moveComponent

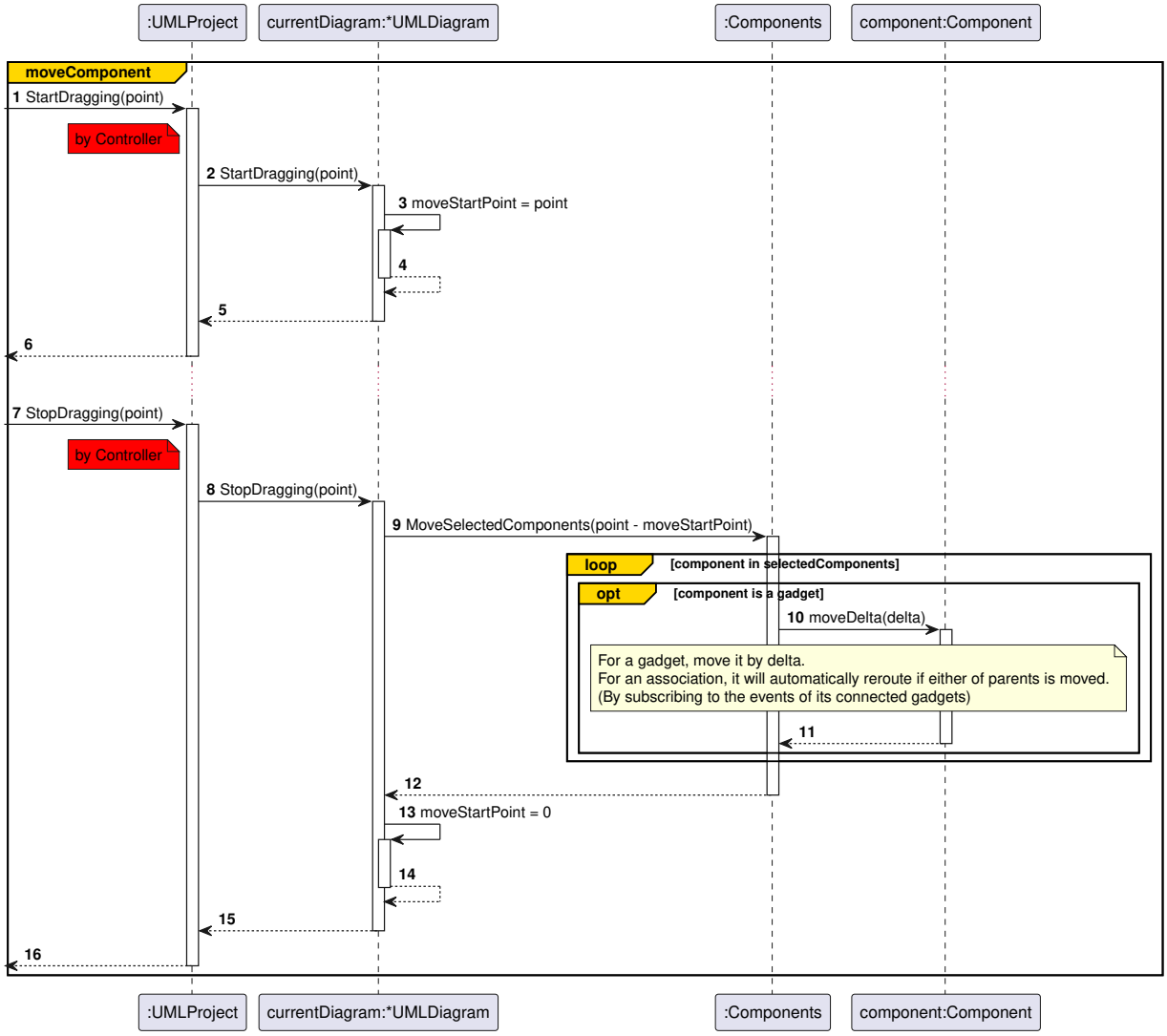


Figure 13: moveComponent

11.9 openProject

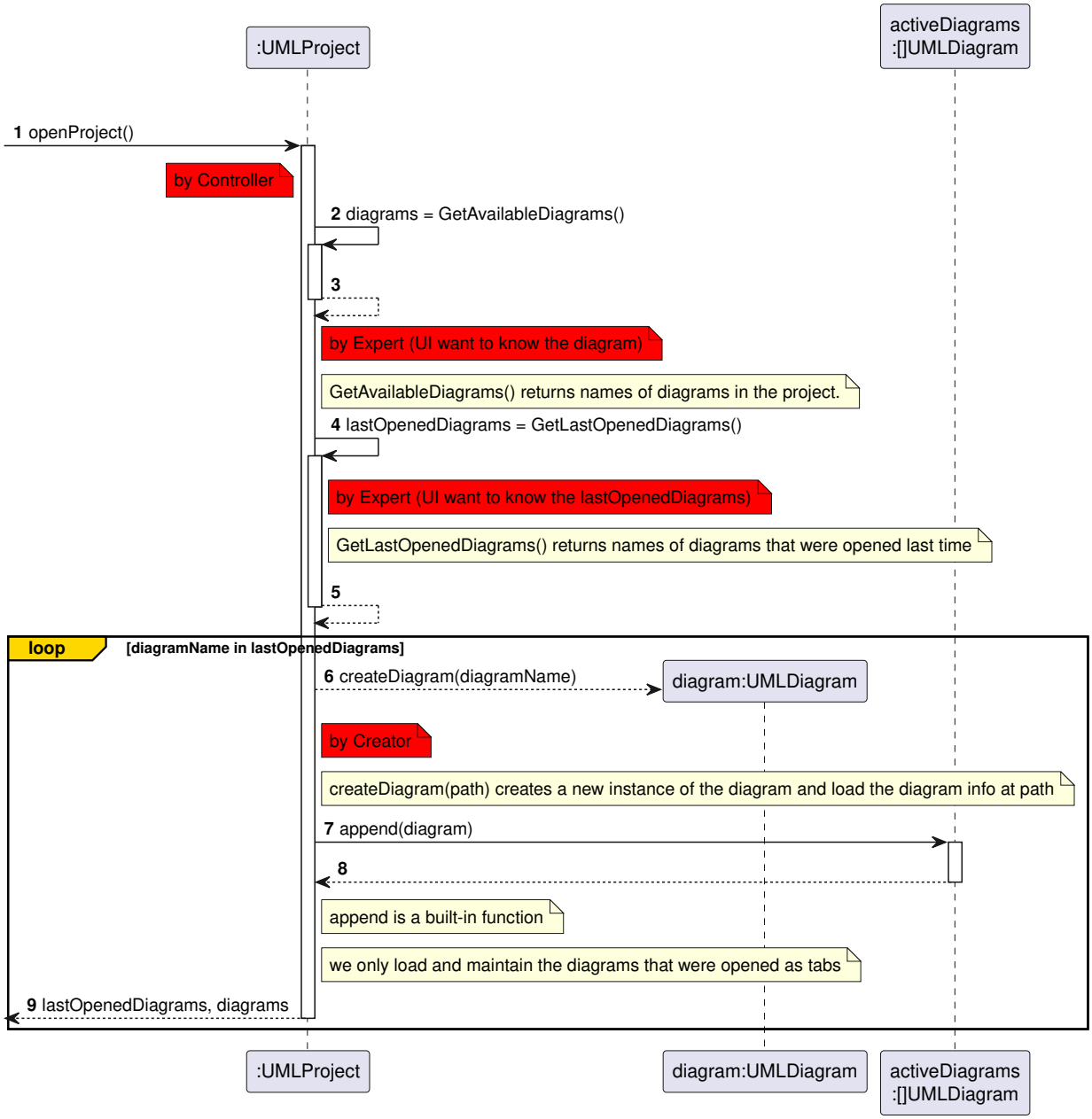


Figure 14: openProject

11.10 pasteComponents

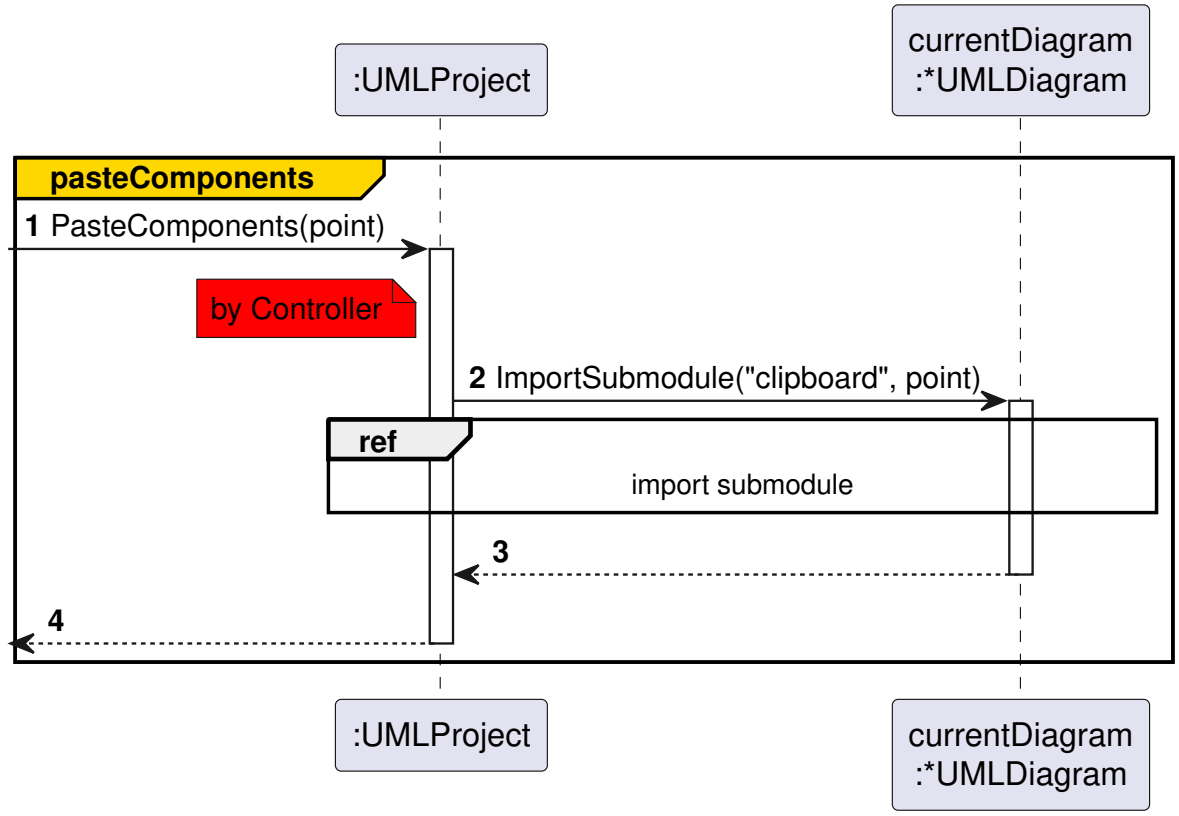


Figure 15: pasteComponents

11.11 redo

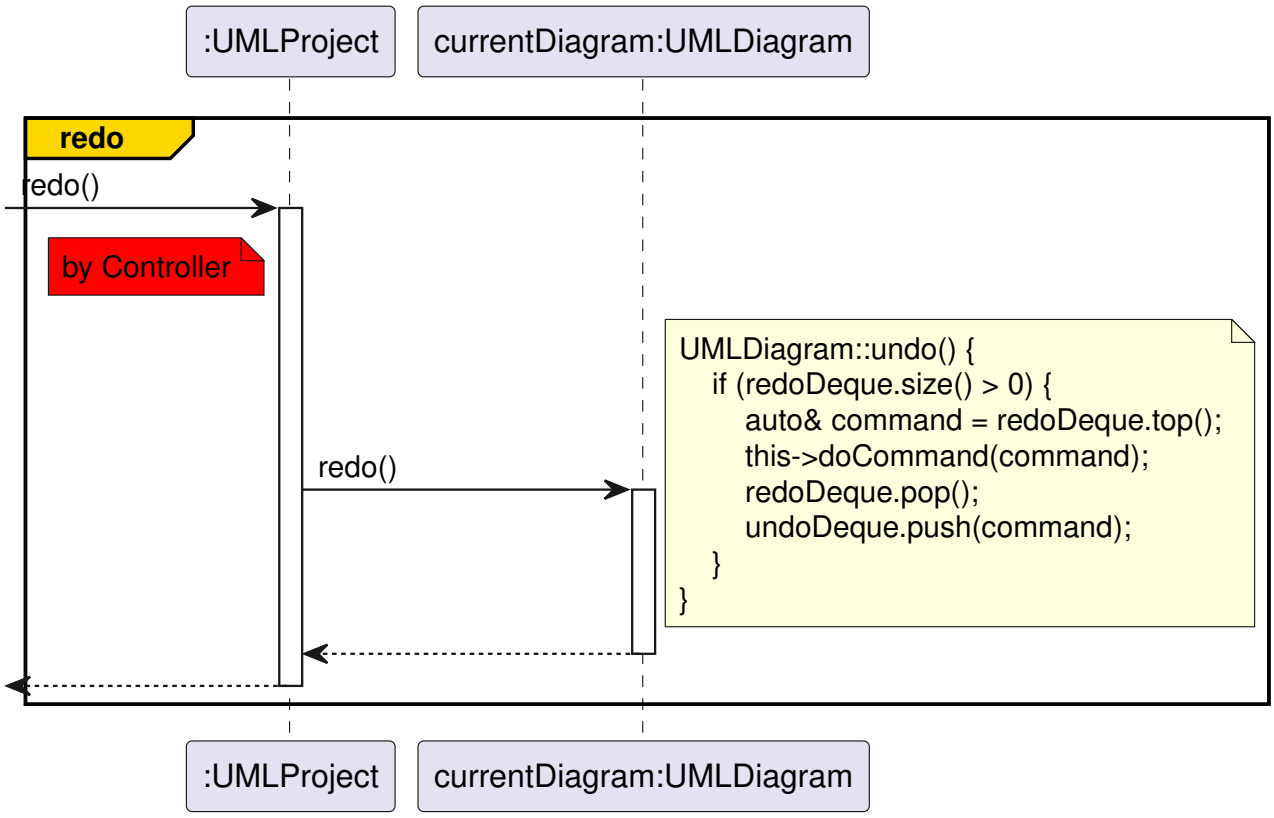


Figure 16: redo

11.12 undo

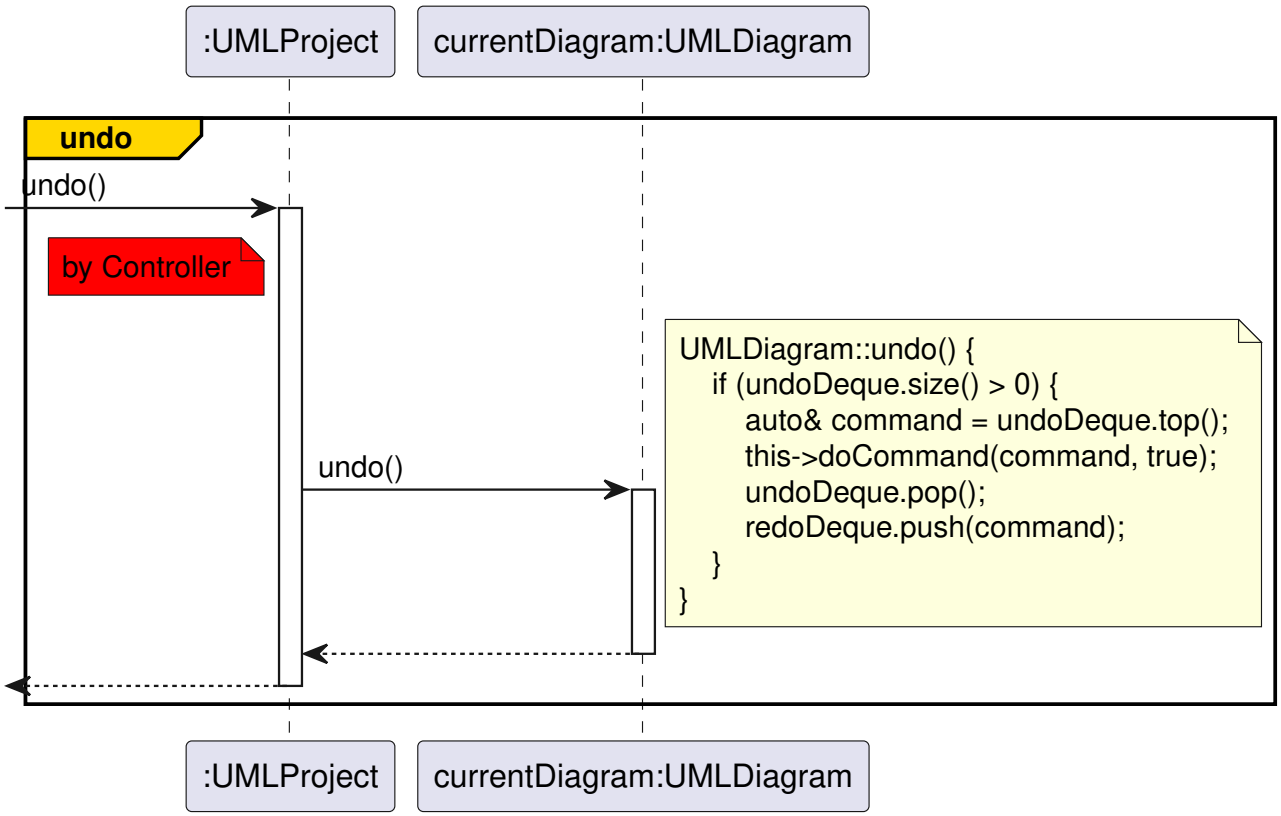


Figure 17: undo

11.13 selectComponent

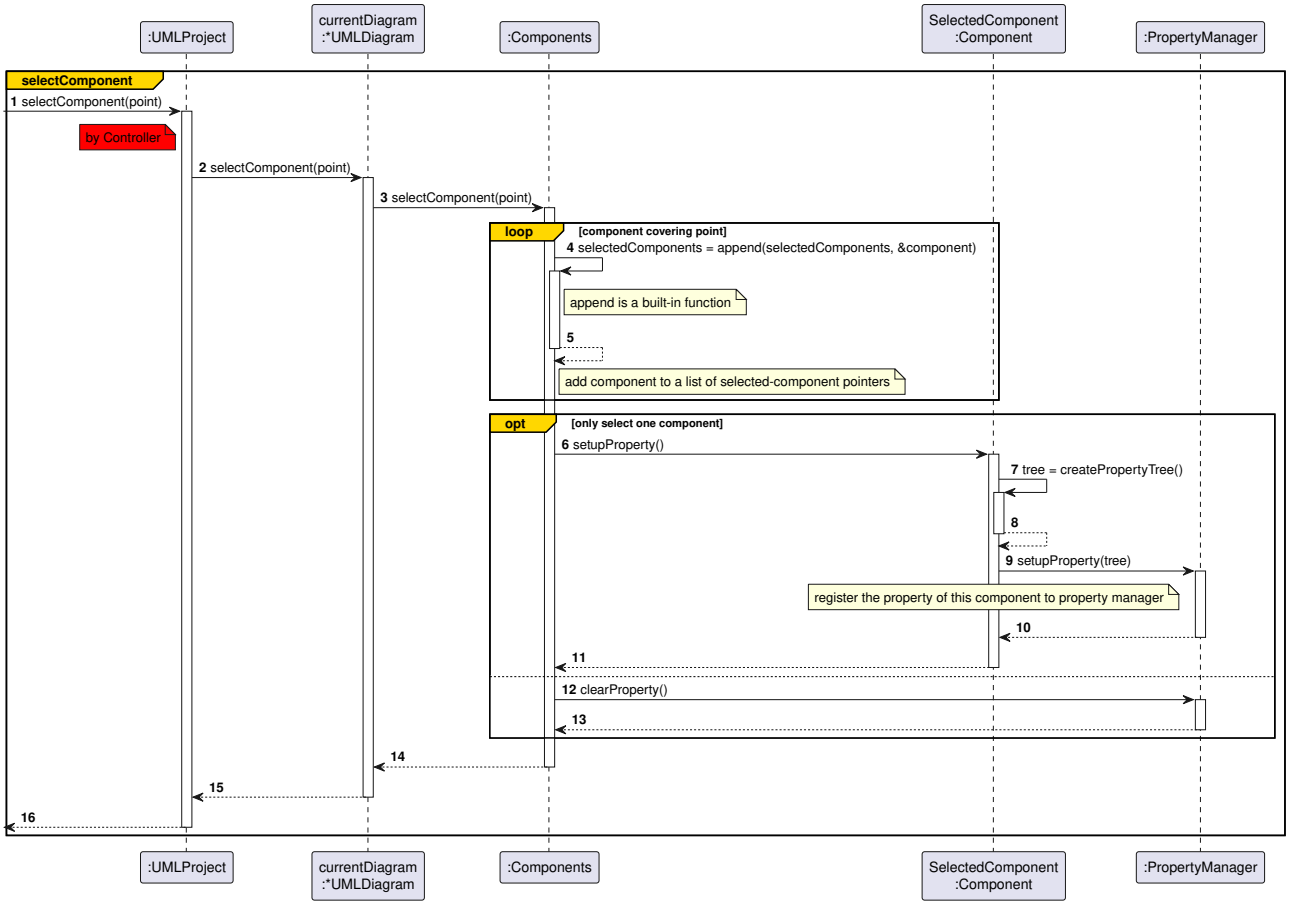


Figure 18: selectComponent

11.14 selectDiagram

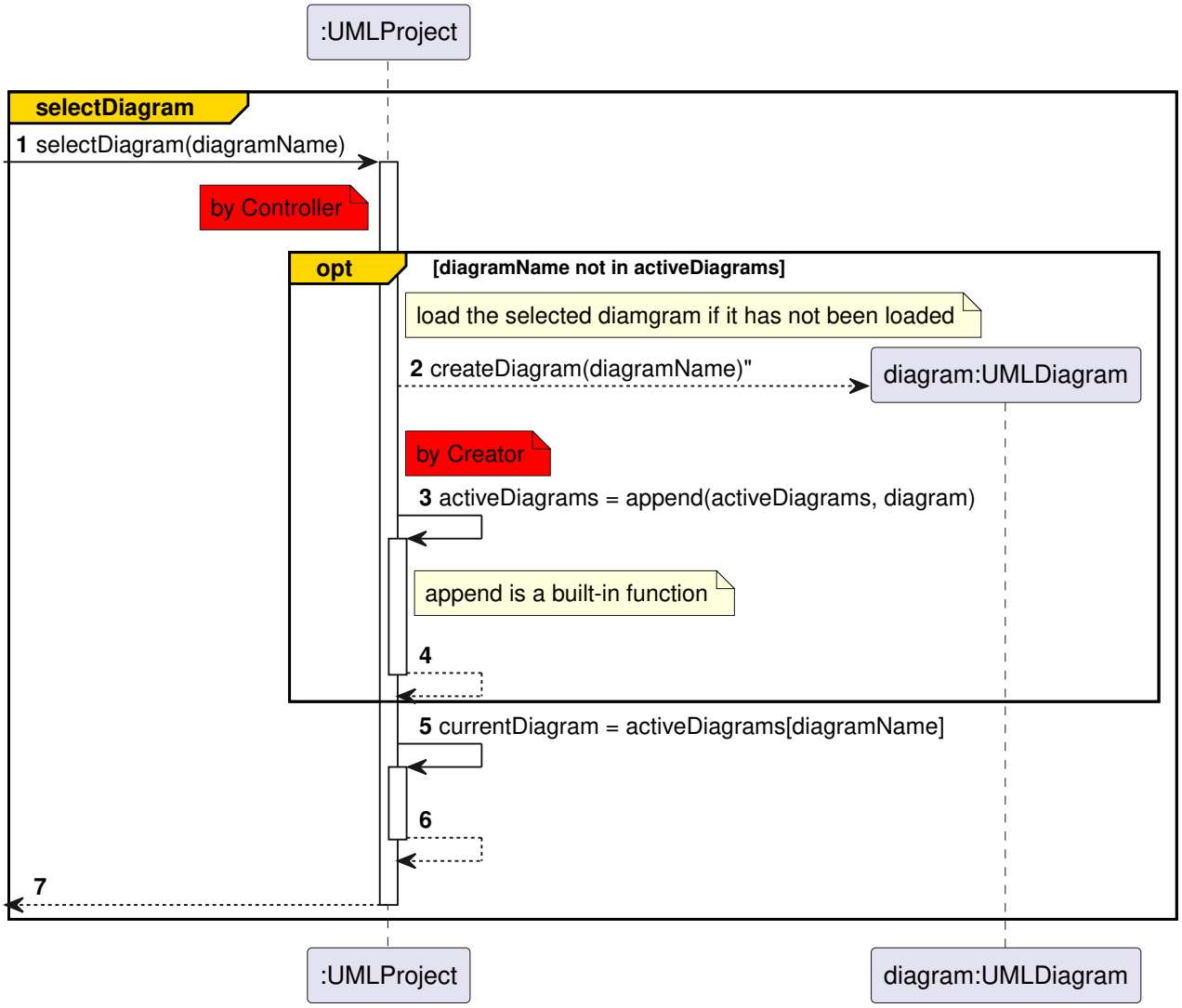


Figure 19: selectDiagram

11.15 unselectAllComponents

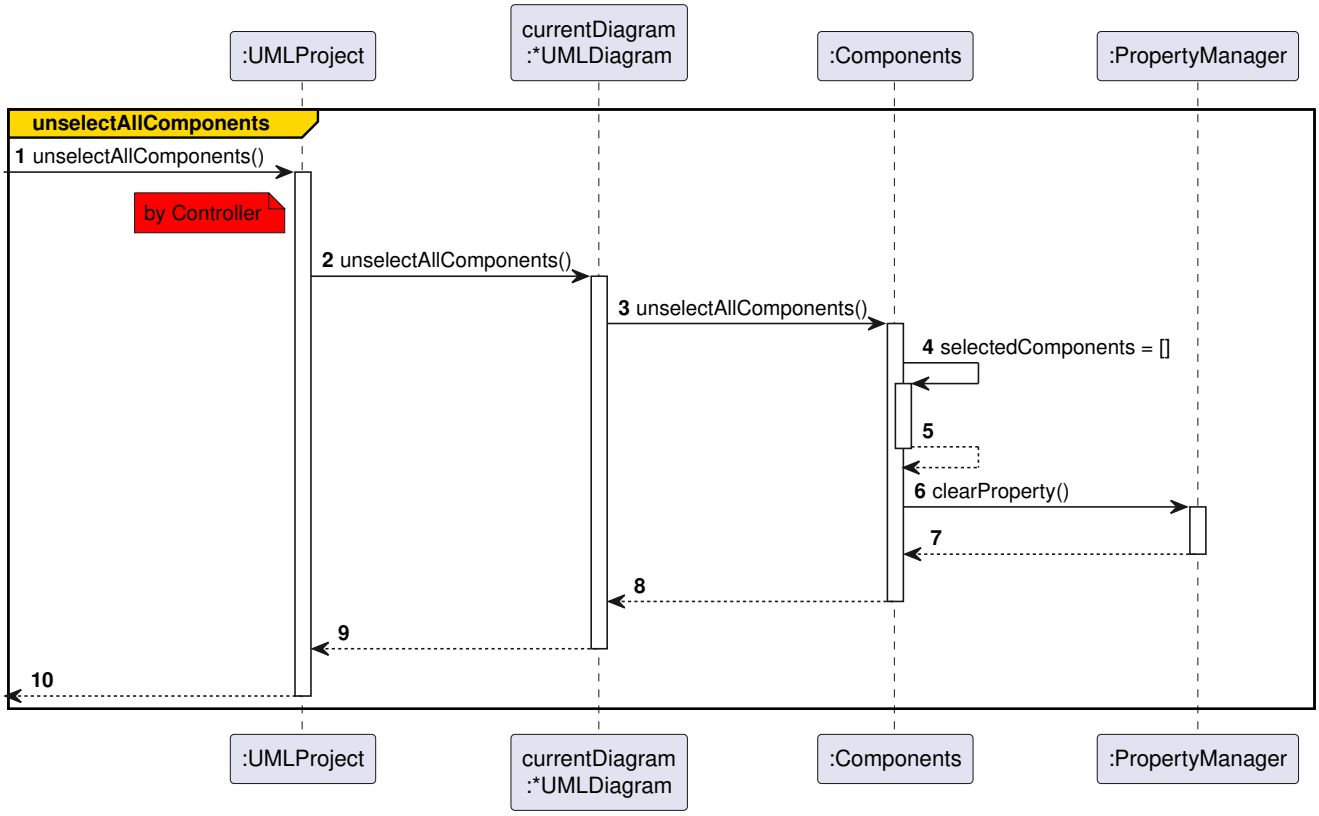


Figure 20: unselectAllComponents

11.16 unselectComponent

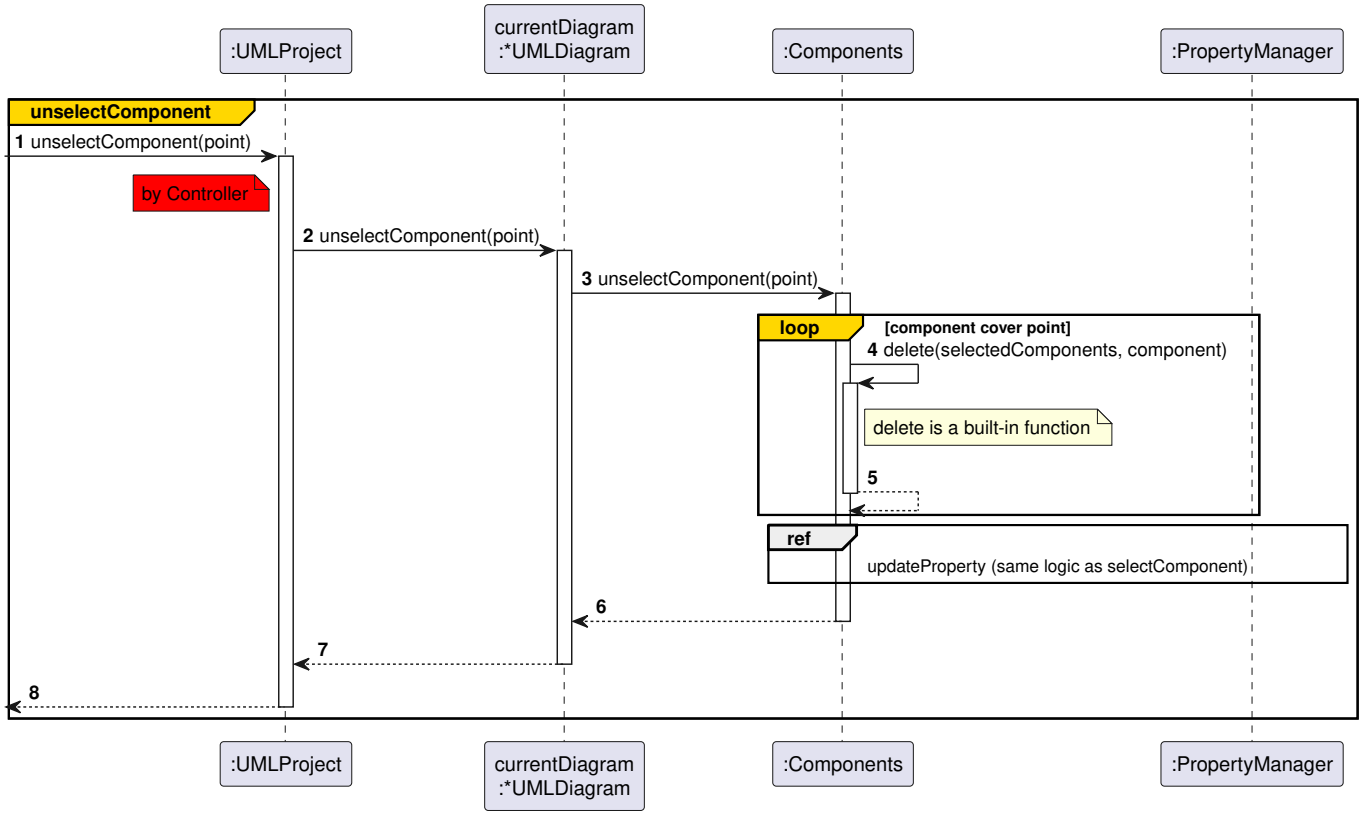


Figure 21: unselectComponent

11.17 updateProperty

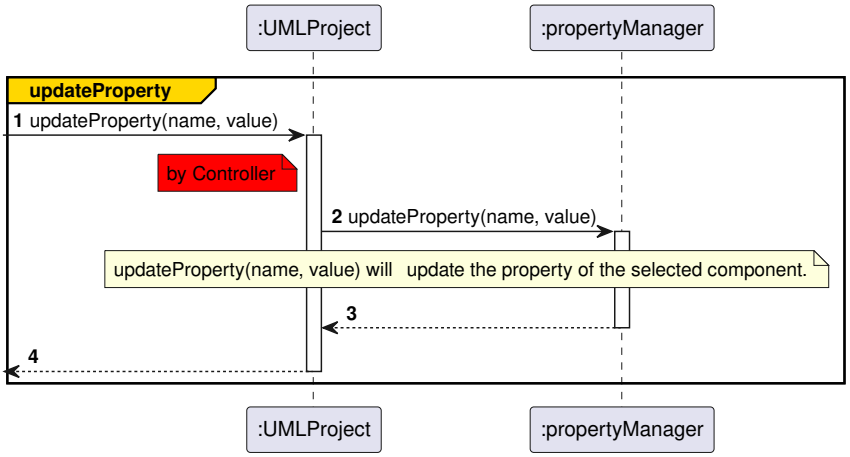


Figure 22: updateProperty

12 Design Class Model

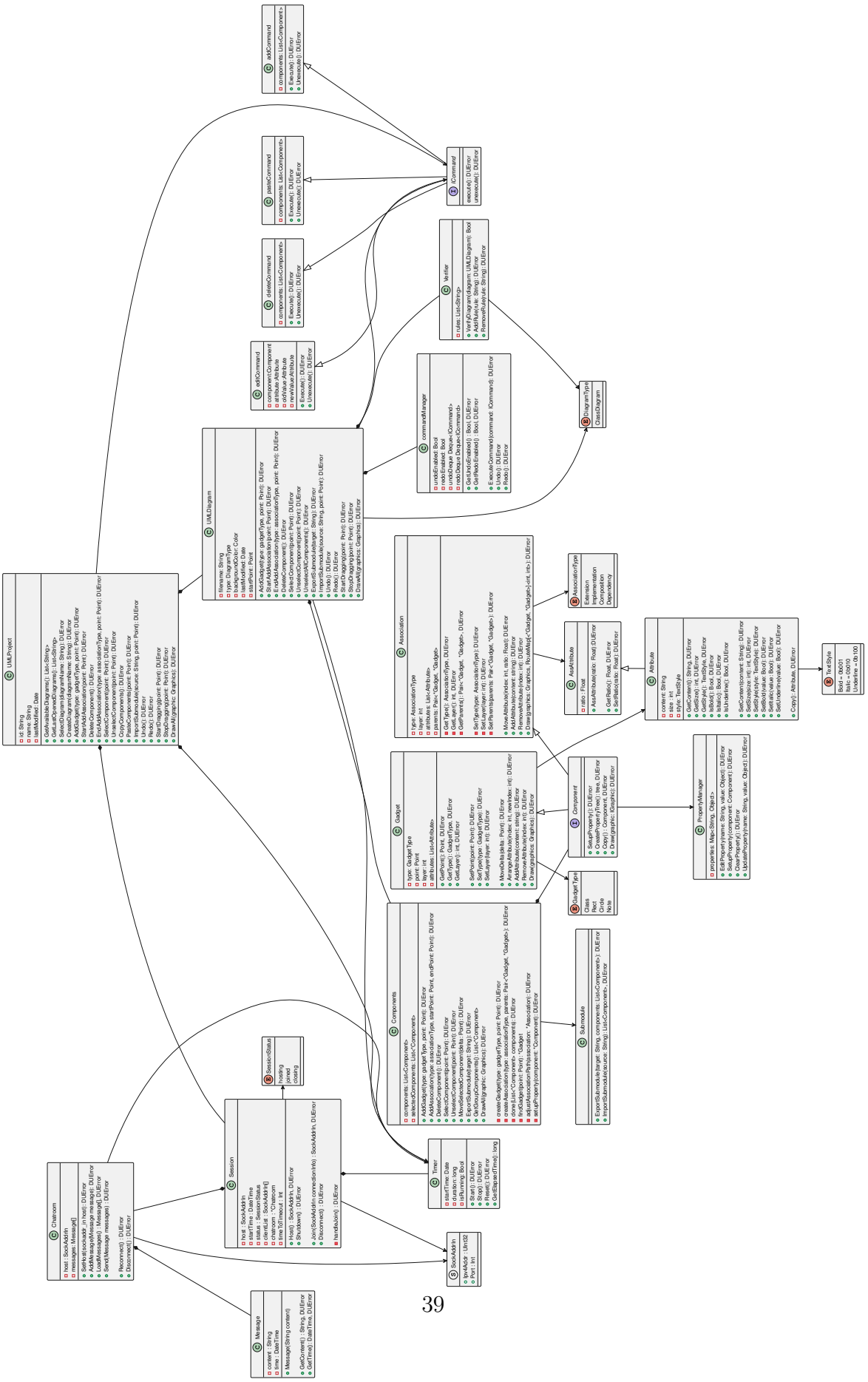


Figure 23: Design Class Diagram

13 Implementation Class Model

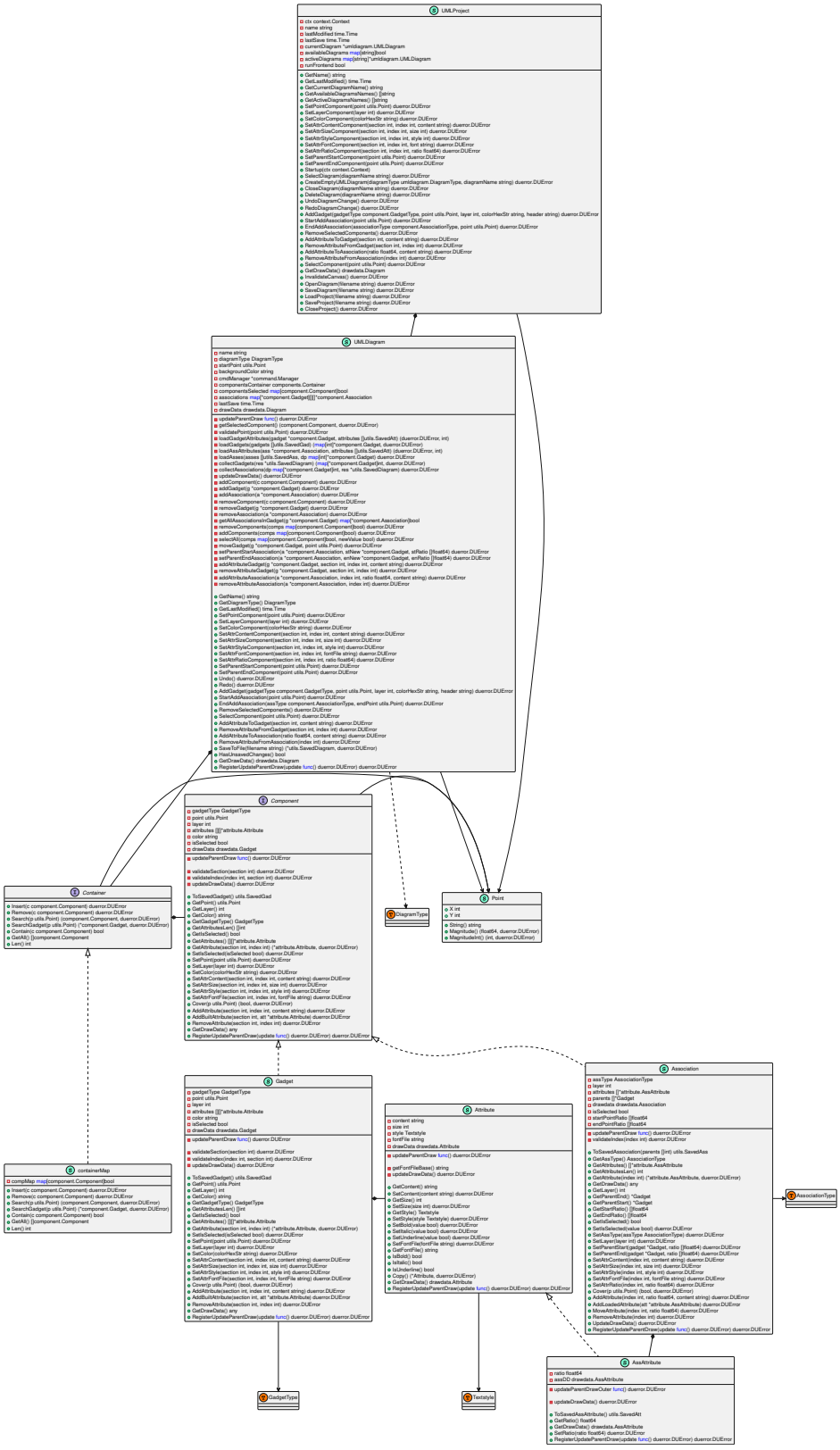


Figure 24: Implementation Design Class Diagram

Table 3: Comparison with design and implementation class

Class	**Method**	**Design**	**Imp.**
DUError	Error	Yes	Yes
	Error	Yes	Yes
AssAttribute	updateParentDrawOuter	No	Yes
	updateDrawData	No	Yes
	ToSavedAssAttribute	Yes	Yes
	GetRatio	Yes	Yes
	GetDrawData	No	Yes
	SetRatio	Yes	Yes
	RegisterUpdateParentDraw	Yes	Yes
Association	updateParentDraw	No	Yes
	validateIndex	No	Yes
	ToSavedAssociation	No	Yes
	GetAssType	Yes	Yes
	GetAttributes	No	Yes
	GetAttributesLen	No	Yes
	GetAttribute	No	Yes
	GetDrawData	No	Yes
	GetLayer	Yes	Yes
	GetParentEnd	Yes	Yes
	GetParentStart	Yes	Yes
	GetStartRatio	No	Yes
	GetEndRatio	No	Yes
	GetIsSelected	No	Yes
	SetIsSelected	No	Yes
	SetAssType	Yes	Yes
	SetLayer	Yes	Yes
	SetParentStart	Yes	Yes
	SetParentEnd	Yes	Yes
	SetAttrContent	No	Yes
	SetAttrSize	No	Yes
	SetAttrStyle	No	Yes
	SetAttrFontFile	No	Yes
	SetAttrRatio	No	Yes
	Cover	No	Yes
	AddAttribute	Yes	Yes
	AddLoadedAttribute	No	Yes
	MoveAttribute	Yes	Yes
	RemoveAttribute	Yes	Yes
	UpdateDrawData	No	Yes
	RegisterUpdateParentDraw	No	Yes
Attribute	updateParentDraw	No	Yes
	getFontFileBase	No	Yes

	updateDrawData	No	Yes
	GetContent	Yes	Yes
	SetContent	Yes	Yes
	GetSize	Yes	Yes
	SetSize	Yes	Yes
	GetStyle	Yes	Yes
	SetStyle	Yes	Yes
	SetBold	Yes	Yes
	SetItalic	Yes	Yes
	SetUnderline	Yes	Yes
	SetFontFile	No	Yes
	GetFontFile	No	Yes
	IsBold	Yes	Yes
	IsItalic	Yes	Yes
	IsUnderline	Yes	Yes
	RegisterUpdateParentDraw	No	Yes
	GetDrawData	No	Yes
	Copy	No	Yes
Commands (not in dcd)	Execute	Yes	No
	Unexecute	No	Yes
	GetBefore	No	Yes
	GetAfter	No	Yes
Component	SetupProperty	Yes	No
	CreatePropertyTree	Yes	No
	Copy	Yes	No
	Draw	Yes	No
	GetLayer	Yes	Yes
	GetPoint	Yes	No
	updateParentDraw	Yes	No
	validateSection	Yes	No
	validateIndex	Yes	No
	updateDrawData	Yes	No
	ToSavedGadget	Yes	No
	GetColor	Yes	No
	GetGadgetType	Yes	No
	GetAttributesLen	Yes	No
	GetIsSelected	Yes	No
	GetAttributes	Yes	No
	GetAttribute	Yes	No
	SetIsSelected	Yes	Yes
	SetPoint	Yes	Yes
	SetLayer	Yes	No
	SetColor	Yes	No
	SetAttrContent	Yes	No
	SetAttrSize	Yes	No
	SetAttrStyle	Yes	Yes

	SetAttrFontFile	Yes	No
	Cover	Yes	Yes
	AddAttribute	Yes	No
	AddBuiltAttribute	Yes	No
	RemoveAttribute	Yes	No
	GetDrawData	Yes	Yes
	RegisterUpdateParentDraw	Yes	Yes
ConnectionError	Error	Yes	Yes
Container	Insert	No	Yes
	Remove	No	Yes
	Search	No	Yes
	SearchGadget	No	Yes
	Contain	No s	Yes
	GetAll	No	Yes
	Len	No	Yes
ContainerMap (not in dcd)	Insert	Yes	Yes
	Insert	No	Yes
	Insert	No	Yes
	Remove	No	Yes
	Search	No	Yes
	SearchGadget	No	Yes
	Contain	No	Yes
	GetAll	No	Yes
	Len	Yes	Yes
CorruptedFileError	Error	Yes	Yes
FileIOError	Error	Yes	Yes
Gadget	updateParentDraw	No	Yes
	validateSection	No	Yes
	validateIndex	No	Yes
	updateDrawData	No	Yes
	ToSavedGadget	No	Yes
	GetPoint	Yes	Yes
	GetLayer	Yes	Yes
	GetColor	No	Yes
	GetGadgetType	No	Yes
	GetAttributesLen	No	Yes
	GetIsSelected	No	Yes
	GetAttributes	No	Yes
	GetAttribute	No	Yes
	SetIsSelected	No	Yes
	SetPoint	Yes	Yes
	SetLayer	Yes	Yes
	SetColor	No	Yes
	SetAttrContent	No	Yes
	SetAttrSize	No	Yes
	SetAttrStyle	No	Yes

	SetAttrFontFile	No	Yes
	Cover	No	Yes
	AddAttribute	Yes	Yes
	AddBuiltAttribute	Yes	Yes
	RemoveAttribute	No	Yes
	GetDrawData	No	Yes
	RegisterUpdateParentDraw	No	Yes
	SetPoint	Yes	No
	MoveDelta	Yes	No
	ArrangeAttribute	Yes	No
	Draw	Yes	No
InvalidArgumentError	Error	Yes	Yes
Manager (not in dcd)	GetLastModified	No	Yes
	Execute	No	Yes
	Undo	No	Yes
	Redo	No	Yes
MemoryFullError	Error	Yes	Yes
ParseError	Error	Yes	Yes
Point (not in dcd)	String	No	Yes
	Magnitude	No	Yes
	MagnitudeInt	No	Yes
SendError	Error	Yes	Yes
Umldiagram	getSelectedComponent	No	Yes
	validatePoint	No	Yes
	loadGadgetAttributes	No	Yes
	loadGadgets	No	Yes
	loadAssAttributes	No	Yes
	loadAsses	No	Yes
	collectGadgets	No	Yes
	collectAssociations	No	Yes
	updateDrawData	No	Yes
	addComponent	No	Yes
	addGadget	No	Yes
	addAssociation	No	Yes
	removeComponent	No	Yes
	removeGadget	No	Yes
	removeAssociation	No	Yes
	getAllAssociationsInGadget	No	Yes
	removeComponents	No	Yes
	addComponents	No	Yes
	selectAll	No	Yes
	moveGadget	No	Yes
	setParentStartAssociation	No	Yes
	setParentEndAssociation	No	Yes
	addAttributeGadget	No	Yes
	removeAttributeGadget	No	Yes

	addAttributeAssociation	No	Yes
	removeAttributeAssociation	No	Yes
	GetName	No	Yes
	GetDiagramType	No	Yes
	GetLastModified	No	Yes
	SetPointComponent	No	Yes
	SetLayerComponent	No	Yes
	SetColorComponent	No	Yes
	SetAttrContentComponent	No	Yes
	SetAttrSizeComponent	No	Yes
	SetAttrStyleComponent	No	Yes
	SetAttrFontComponent	No	Yes
	SetAttrRatioComponent	No	Yes
	SetParentStartComponent	No	Yes
	SetParentEndComponent	No	Yes
	Undo	Yes	Yes
	Redo	Yes	Yes
	AddGadget	Yes	Yes
	StartAddAssociation	Yes	Yes
	EndAddAssociation	Yes	Yes
	RemoveSelectedComponents	Yes	Yes
	SelectComponent	Yes	Yes
	AddAttributeToGadget	No	Yes
	RemoveAttributeFromGadget	No	Yes
	AddAttributeToAssociation	No	Yes
	RemoveAttributeFromAssociation	No	Yes
	SaveToFile	No	Yes
	HasUnsavedChanges	No	Yes
	GetDrawData	No	Yes
	RegisterUpdateParentDraw	No	Yes
	UnselectComponent	No	No
	UnselectAllComponents	No	No
	ExportSubmodule	No	No
	ImportSubmodule	No	No
	StartDragging	No	No
	StopDragging	No	No
	createAssociation	No	No
	findGadget	No	No
	adjustAssociationPath	No	No
	setupProperty	No	No
Umlproject	GetName	No	Yes
	GetLastModified	No	Yes
	GetCurrentDiagramName	No	Yes
	GetAvailableDiagramsNames	Yes	Yes
	GetActiveDiagramsNames	No	Yes
	SetPointComponent	No	Yes

	SetLayerComponent	No	Yes
	SetColorComponent	No	Yes
	SetAttrContentComponent	No	Yes
	SetAttrSizeComponent	No	Yes
	SetAttrStyleComponent	No	Yes
	SetAttrFontComponent	No	Yes
	SetAttrRatioComponent	No	Yes
	SetParentStartComponent	No	Yes
	SetParentEndComponent	No	Yes
	Startup	No	Yes
	SelectDiagram	Yes	Yes
	CreateEmptyUMLDiagram	Yes	Yes
	CloseDiagram	No	Yes
	DeleteDiagram	No	Yes
	UndoDiagramChange	Yes	Yes
	RedoDiagramChange	Yes	Yes
	AddGadget	Yes	Yes
	StartAddAssociation	Yes	Yes
	EndAddAssociation	Yes	Yes
	RemoveSelectedComponents	Yes	Yes
	AddAttributeToGadget	No	Yes
	RemoveAttributeFromGadget	No	Yes
	AddAttributeToAssociation	No	Yes
	RemoveAttributeFromAssociation	No	Yes
	SelectComponent	Yes	Yes
	GetDrawData	No	Yes
	InvalidateCanvas	No	Yes
	OpenDiagram	No	Yes
	SaveDiagram	No	Yes
	LoadProject	No	Yes
	SaveProject	No	Yes
	CloseProject	No	Yes
	GetLastOpenedDiagrams	Yes	No
	UnselectComponent	Yes	No
	CopyComponents	Yes	No
	PasteComponents	Yes	No
	ImportSubmodule	Yes	No
	StartDragging	Yes	No
	StopDragging	Yes	No
	DrawAll	Yes	No

Table 5: Line of Code of classes

No	Class Name	Number of methods	Line of Code in Class
1	DUErrorError	1	5
2	assAttribute	6	108

3	association	32	557
4	attribute	19	215
5	commands	24	247
6	component	7	19
7	connectionError	1	13
8	container	7	17
9	containerMap	7	92
10	corruptedFileError	1	13
11	fileIOError	1	13
12	gadget	29	424
13	invalidArgumentError	1	13
14	manager	4	80
15	memoryFullError	1	13
16	parseError	1	13
17	point	3	48
18	sendError	1	13
19	umldiagram	55	1256
20	umlproject	42	657
21	total	242	3816

14 Programming

14.1 Snapshots of System Execution

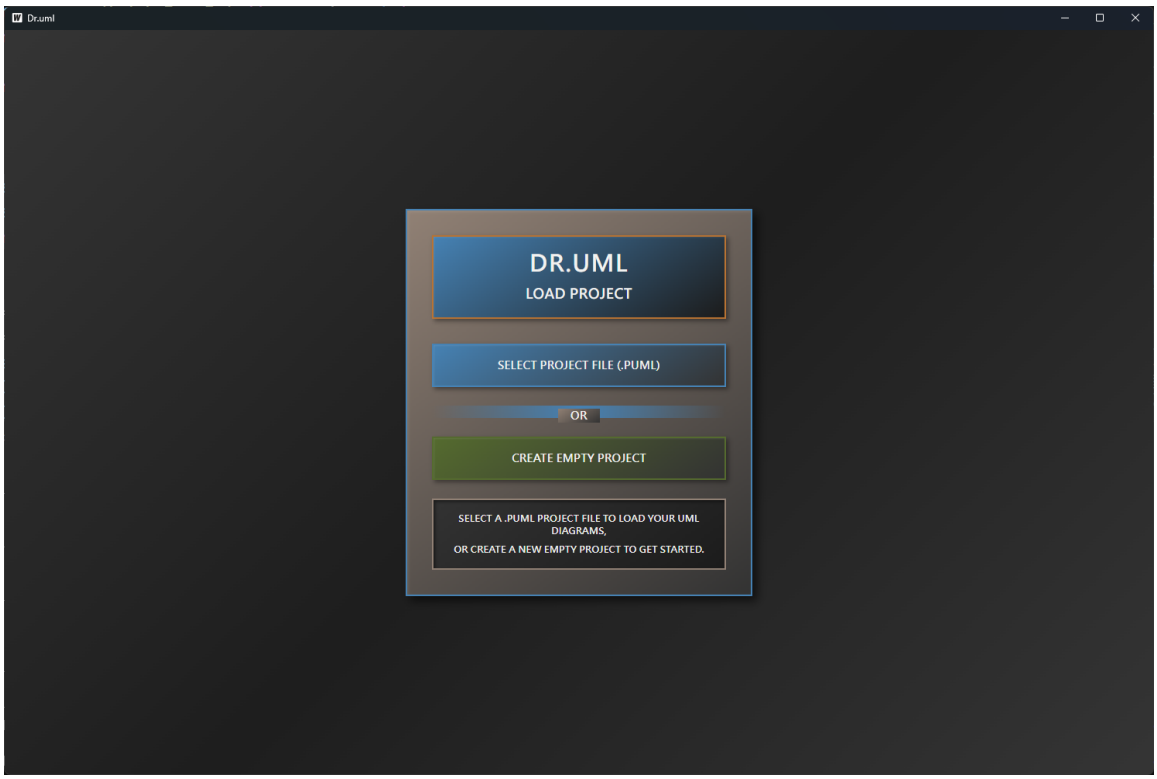
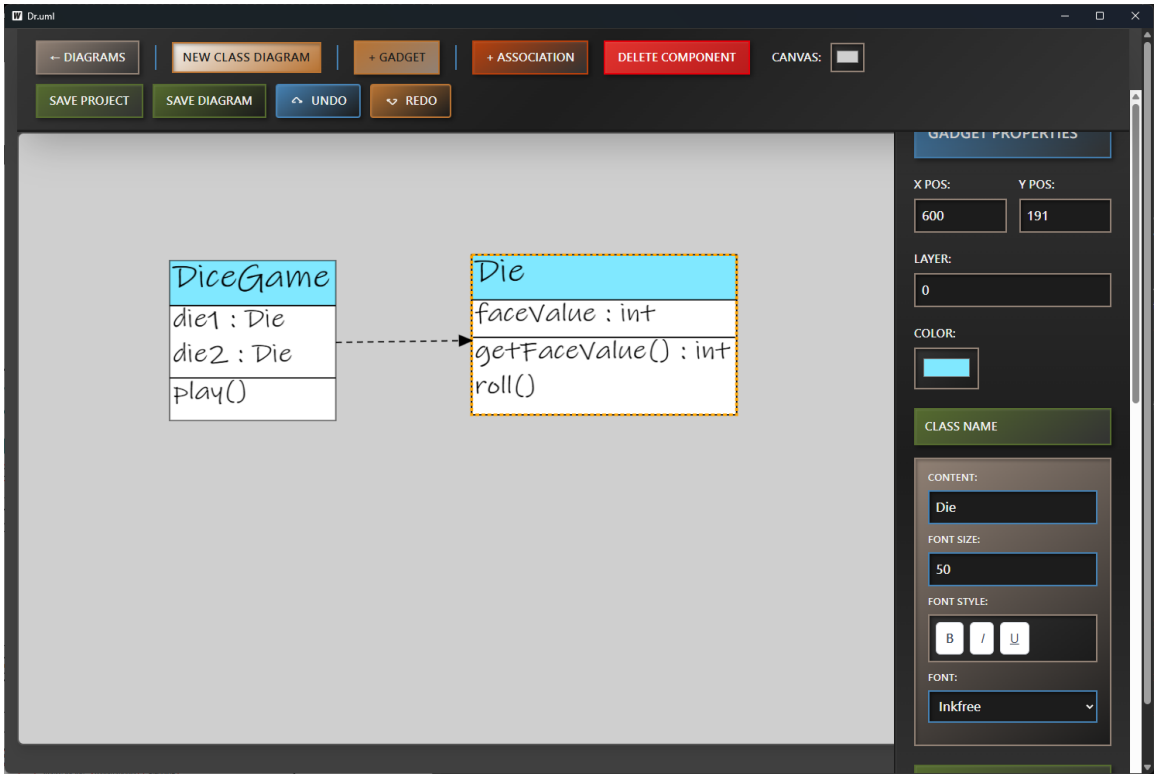


Table 4: Summary of implementation class/method changed

	Number of Added	Number of Removed	Number of Modified
Class	10	13	10
Methods	119	37	47




```go
func (m *Manager) Execute(cmd Command) duerror.DUError {
    if cmd == nil {
        return duerror.NewInvalidArgumentError("command is nil")
    }
    if err := cmd.Execute(); err != nil {
        return err
    }
    if len(m.undoStack) == m.limit {
        m.undoStack = m.undoStack[1:]
    }
    m.undoStack = append(m.undoStack, cmd)
    m.redoStack = nil
    m.lastModified = cmd.GetAfter()
    return nil
}
```

```go
func (ud *UMLDiagram) AddGadget(
    gadgetType component.GadgetType,
    point utils.Point,
    layer int,
    colorHexStr string,
    header string) duerror.DUError {

    g, err := component.NewGadget(gadgetType, point, layer, colorHexStr, header)
    if err != nil {
        return err
    }
    if err = g.RegisterUpdateParentDraw(ud.updateDrawData); err != nil {
        return err
    }
    cmd := &addComponentCommand{
        baseCommand: baseCommand{
            diagram: ud,
            before:  ud.GetLastModified(),
            after:   time.Now(),
        },
        component: g,
    }
    if err := ud.cmdManager.Execute(cmd); err != nil {
        return err
    }
    return nil
}
```

```go
func (ud *UMLDiagram) RemoveSelectedComponents() duerror.DUError {
    comps := map[component.Component]bool{}
    for c := range ud.componentsSelected {
        comps[c] = true
        switch g := c.(type) {
        case *component.Gadget:
            for a := range ud.getAllAssociationsInGadget(g) {
                comps[a] = true
            }
        }
    }
    cmd := &removeSelectedComponentCommand{
        baseCommand: baseCommand{
            diagram: ud,
            before:  ud.GetLastModified(),
            after:   time.Now(),
        },
        components: comps,
    }
    if err := ud.cmdManager.Execute(cmd); err != nil {
        return err
    }
    return nil
}
```

```go
func (g *Gadget) SetAttrContent(section int, index int, content string) duerror.DUError {
    if err := g.validateSection(section); err != nil {
        return err
    }
    if err := g.validateIndex(index, section); err != nil {
        return err
    }
    if err := g.attributes[section][index].SetContent(content); err != nil {
        return err
    }
    return g.updateDrawData()
}
```

```go
func (ass *Association) Cover(p utils.Point) (bool, duerror.DUError) {
    if ass.parents[0] == nil || ass.parents[1] == nil {
        return false, duerror.NewInvalidArgumentError("parents are nil")
    }

    st := utils.Point{X: ass.drawdata.StartX, Y: ass.drawdata.StartY}
    en := utils.Point{X: ass.drawdata.EndX, Y: ass.drawdata.EndY}
    delta := utils.Point{X: ass.drawdata.DeltaX, Y: ass.drawdata.DeltaY}
    stDelta := utils.AddPoints(st, delta)
    enDelta := utils.AddPoints(en, delta)

    threshold := float64(4)
    return dist(stDelta, enDelta, p) <= threshold ||
        dist(st, stDelta, p) <= threshold ||
        dist(en, enDelta, p) <= threshold, nil
}
```

14.2 Source Code Listing

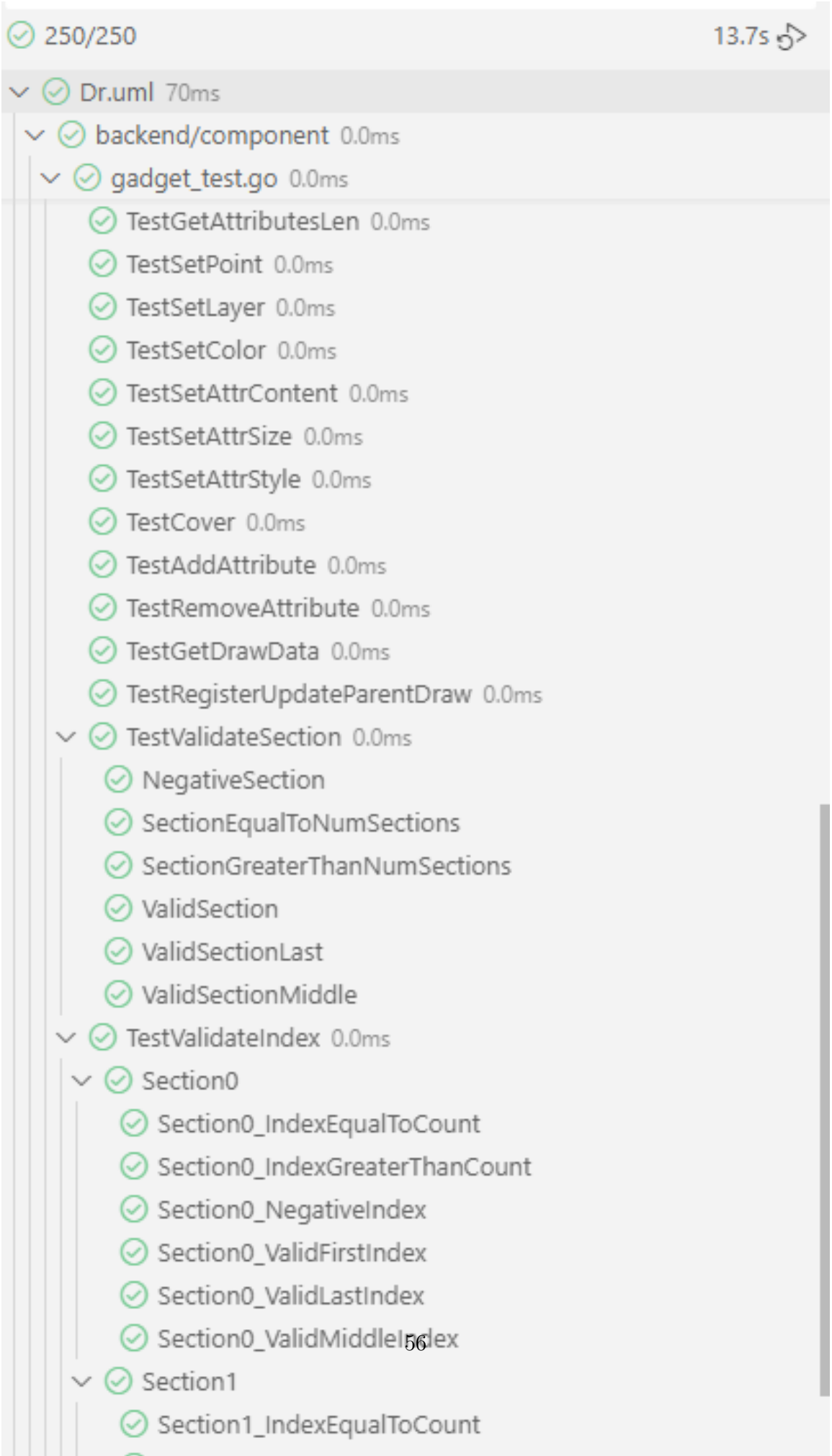
```go
package component

import (
    "Dr.uml/backend/component/attribute"
    "Dr.uml/backend/drawdata"
    "Dr.uml/backend/utils"
    "Dr.uml/backend/utils/duerror"
)

type GadgetType int

const (
    Class               GadgetType = 1 << iota // 0x01
    supportedGadgetType            = Class
)

type Gadget struct {
    gadgetType       GadgetType
    point            utils.Point
    layer            int
    attributes       [][]attribute.Attribute // Gadget have multiple sections, each section have multiple attr
    color            utils.Color
    drawData         drawdata.Gadget
    updateParentDraw func() duerror.DUError
}

/*
component interface
*/

func (g *Gadget) Cover(p utils.Point) (bool, duerror.DUError) {
    tl := g.point                                                          // top-left
    br := utils.AddPoints(g.point, utils.Point{X: g.drawData.Width, Y: g.drawData.Height}) // bottom-right
    return p.X >= tl.X && p.X <= br.X && p.Y >= tl.Y && p.Y <= br.Y, nil
}

func (g *Gadget) GetLayer() (int, duerror.DUError) {
    return g.layer, nil
}
```

```go
package component

import (
    "Dr.uml/backend/component/attribute"
    "Dr.uml/backend/drawdata"
    "Dr.uml/backend/utils"
    "Dr.uml/backend/utils/duerror"
)

type AssociationType int

const (
    Extension               = 1 << iota // 0x01
    Implementation          = 1 << iota // 0x02
    Composition             = 1 << iota // 0x04
    Dependency              = 1 << iota // 0x08
    supportedAssociationType = Extension | Implementation | Composition | Dependency
)

type Association struct {
    assType          AssociationType
    layer            int
    attributes       []*attribute.AssAttribute
    parents          [2]*Gadget
    drawdata         drawdata.Association
    updateParentDraw func() duerror.DUError
}

// Constructor
func NewAssociation(parents [2]*Gadget, assType AssociationType) (*Association, duerror.DUError) {
    if assType&supportedAssociationType != assType || assType == 0 {
        return nil, duerror.NewInvalidArgumentError("unsupported association type")
    }
    if parents[0] == nil || parents[1] == nil {
        return nil, duerror.NewInvalidArgumentError("parents are nil")
    }
    a := &Association{
        parents: [2]*Gadget{parents[0], parents[1]},
    }
    a.updateDrawData()
    return a, nil
}

// Getters
func (this *Association) GetAssType() AssociationType {
    return this.assType
}

```

15 Unit Testing

15.1 Snapshot


Filter (e.g. text, !exclude, @tag)
250/250
13.7s
Dr.uml 70ms
backend/component 0.0ms
association_test.go 0.0ms
Test_Association_RemoveAttribute 0.0ms
Remove_valid_attribute
Test_Association_MoveAttribute 0.0ms
Move_invalid_index
Move_valid_attribute
Test_Association_Cover 0.0ms
Point_inside_threshold
Point_outside_threshold
Test_Association_UpdateDrawData 0.0ms
Update_with_valid_data
Test_Association_RegisterUpdateParentDraw 0.0ms
Register_nil_function
Register_valid_function
Test_Association_ToSavedAssociation 0.0ms
gadget_test.go 0.0ms
TestNewGadget 0.0ms
TestGetPoint 0.0ms
TestGetLayer 0.0ms
TestGetColor 0.0ms
TestGetGadgetType 0.0ms
TestGetAttributesLen 0.0ms
TestSetPoint 0.0ms
TestSetLayer 0.0ms
TestSetColor 0.0ms
TestSetAttrContent 0.0ms
TestSetAttrSize 0.0ms
TestSetAttrStyle 0.0ms


250/250 13.7s
Dr.uml 70ms
backend/component 0.0ms
gadget_test.go 0.0ms
TestGetAttributesLen 0.0ms
TestSetPoint 0.0ms
TestSetLayer 0.0ms
TestSetColor 0.0ms
TestSetAttrContent 0.0ms
TestSetAttrSize 0.0ms
TestSetAttrStyle 0.0ms
TestCover 0.0ms
TestAddAttribute 0.0ms
TestRemoveAttribute 0.0ms
TestGetDrawData 0.0ms
TestRegisterUpdateParentDraw 0.0ms
TestValidateSection 0.0ms
NegativeSection
SectionEqualToNumSections
SectionGreaterThanNumSections
ValidSection
ValidSectionLast
ValidSectionMiddle
TestValidateIndex 0.0ms
Section0
Section0_IndexEqualToCount
Section0_IndexGreaterThanCount
Section0_NegativeIndex
Section0_ValidFirstIndex
Section0_ValidLastIndex
Section0_ValidMiddleIndex
Section1
Section1_IndexEqualToCount

```
Dr.uml/backend/umlproject/umlproject.go:187:          SetAssociationType                83.3%
Dr.uml/backend/umlproject/umlproject.go:199:          Startup                           100.0%
Dr.uml/backend/umlproject/umlproject.go:207:          SelectDiagram                     87.5%
Dr.uml/backend/umlproject/umlproject.go:223:          CreateEmptyUMLDiagram             100.0%
Dr.uml/backend/umlproject/umlproject.go:237:          CloseDiagram                      90.0%
Dr.uml/backend/umlproject/umlproject.go:255:          DeleteDiagram                     100.0%
Dr.uml/backend/umlproject/umlproject.go:260:          UndoDiagramChange                 83.3%
Dr.uml/backend/umlproject/umlproject.go:271:          RedoDiagramChange                 83.3%
Dr.uml/backend/umlproject/umlproject.go:282:          AddGadget                         100.0%
Dr.uml/backend/umlproject/umlproject.go:293:          StartAddAssociation               100.0%
Dr.uml/backend/umlproject/umlproject.go:300:          EndAddAssociation                 83.3%
Dr.uml/backend/umlproject/umlproject.go:311:          RemoveSelectedComponents          83.3%
Dr.uml/backend/umlproject/umlproject.go:322:          AddAttributeToGadget              83.3%
Dr.uml/backend/umlproject/umlproject.go:333:          RemoveAttributeFromGadget         83.3%
Dr.uml/backend/umlproject/umlproject.go:344:          AddAttributeToAssociation         66.7%
Dr.uml/backend/umlproject/umlproject.go:355:          RemoveAttributeFromAssociation    83.3%
Dr.uml/backend/umlproject/umlproject.go:366:          SelectComponent                   83.3%
Dr.uml/backend/umlproject/umlproject.go:378:          GetDrawData                       100.0%
Dr.uml/backend/umlproject/umlproject.go:385:          InvalidateCanvas                  66.7%
Dr.uml/backend/umlproject/umlproject.go:397:          OpenDiagram                       82.8%
Dr.uml/backend/umlproject/umlproject.go:450:          SaveDiagram                       84.0%
Dr.uml/backend/umlproject/umlproject.go:489:          LoadProject                       95.2%
Dr.uml/backend/umlproject/umlproject.go:522:          SaveProject                       82.6%
Dr.uml/backend/umlproject/umlproject.go:563:          CloseProject                      80.0%
Dr.uml/backend/umlproject/umlproject.go:574:          OpenFileDialog                    28.6%
Dr.uml/backend/umlproject/umlproject.go:602:          SaveFileDialog                    28.6%
Dr.uml/backend/umlproject/umlproject.go:631:          SaveDiagramFileDialog             28.6%
Dr.uml/backend/utils/checker.go:13:                   ValidateFilePath                  100.0%
Dr.uml/backend/utils/duerror/connection.go:7:         Error                             0.0%
Dr.uml/backend/utils/duerror/connection.go:11:        NewConnectionError                0.0%
Dr.uml/backend/utils/duerror/corruptedFile.go:7:      NewCorruptedFile                  0.0%
Dr.uml/backend/utils/duerror/corruptedFile.go:11:     Error                             0.0%
Dr.uml/backend/utils/duerror/fileIO.go:7:             Error                             0.0%
Dr.uml/backend/utils/duerror/fileIO.go:11:            NewFileIOError                    100.0%
Dr.uml/backend/utils/duerror/invalidArgument.go:7:    Error                             100.0%
Dr.uml/backend/utils/duerror/invalidArgument.go:11:   NewInvalidArgumentError          100.0%
Dr.uml/backend/utils/duerror/memoryFull.go:7:         Error                             0.0%
Dr.uml/backend/utils/duerror/memoryFull.go:11:        NewMemoryFullError                0.0%
Dr.uml/backend/utils/duerror/parse.go:7:              Error                             100.0%
Dr.uml/backend/utils/duerror/parse.go:11:             NewParsingError                   100.0%
Dr.uml/backend/utils/duerror/send.go:7:               Error                             0.0%
Dr.uml/backend/utils/duerror/send.go:11:              NewSendError                      0.0%
Dr.uml/backend/utils/math.go:3:                       AbsInt                            0.0%
Dr.uml/backend/utils/point.go:16:                     FromString                        80.0%
Dr.uml/backend/utils/point.go:25:                     String                            100.0%
Dr.uml/backend/utils/point.go:29:                     Magnitude                         100.0%
Dr.uml/backend/utils/point.go:33:                     MagnitudeInt                      100.0%
Dr.uml/backend/utils/point.go:38:                     EqualPoints                       100.0%
Dr.uml/backend/utils/point.go:42:                     AddPoints                         100.0%
Dr.uml/backend/utils/point.go:46:                     SubPoints                         100.0%
Dr.uml/backend/utils/textlength.go:13:                loadFont                          71.4%
Dr.uml/backend/utils/textlength.go:25:                GetTextSize                       81.8%
Dr.uml/main.go:18:                                    main                              0.0%
total:                                                (statements)                      71.2%
```

Figure 28: Unit test snapshot

15.2 Code listing

```go
// Test lastModified updates
run test | debug test
func TestLastModifiedUpdates(t *testing.T) {
    p, err := CreateEmptyUMLProject("TestProject")
    assert.NoError(t, err)
    err = p.CreateEmptyUMLDiagram(umldiagram.ClassDiagram, "TestDiagram")
    assert.NoError(t, err)
    err = p.SelectDiagram("TestDiagram")
    assert.NoError(t, err)

    initialTime := p.GetLastModified()

    // Sleep to ensure time difference
    time.Sleep(10 * time.Millisecond)

    // Operations that should update lastModified
    operations := []func() error{
        func() error {
            return p.AddGadget(component.Class, utils.Point{X: 10, Y: 10}, 0, drawdata.DefaultGadgetColor, "test")
        },
        func() error { return p.SelectComponent(utils.Point{X: 15, Y: 15}) },
        func() error { return p.SetLayerComponent(1) },
        func() error { return p.SetColorComponent("#FF0000") },
    }

    for i, operation := range operations {
        time.Sleep(10 * time.Millisecond) // Ensure time difference
        err := operation()
        assert.NoError(t, err, "Operation %d failed", i)

        newTime := p.GetLastModified()
        assert.True(t, newTime.After(initialTime), "LastModified should be updated after operation %d", i)
        initialTime = newTime
    }
}
```

Figure 29: Code listing snapshot

```go
// Test multiple diagram management
run test | debug test
func TestMultipleDiagramManagement(t *testing.T) {
    p, err := CreateEmptyUMLProject("TestProject")
    assert.NoError(t, err)

    // Create multiple diagrams
    err = p.CreateEmptyUMLDiagram(umldiagram.ClassDiagram, "Diagram1")
    assert.NoError(t, err)
    err = p.CreateEmptyUMLDiagram(umldiagram.ClassDiagram, "Diagram2")
    assert.NoError(t, err)
    err = p.CreateEmptyUMLDiagram(umldiagram.ClassDiagram, "Diagram3")
    assert.NoError(t, err)

    // Select different diagrams
    err = p.SelectDiagram("Diagram1")
    assert.NoError(t, err)
    assert.Equal(t, "Diagram1", p.GetCurrentDiagramName())

    err = p.SelectDiagram("Diagram2")
    assert.NoError(t, err)
    assert.Equal(t, "Diagram2", p.GetCurrentDiagramName())

    // Close one diagram
    err = p.CloseDiagram("Diagram1")
    assert.NoError(t, err)

    activeDiagrams := p.GetActiveDiagramsNames()
    assert.NotContains(t, activeDiagrams, "Diagram1")
    assert.Contains(t, activeDiagrams, "Diagram2")
    assert.Contains(t, activeDiagrams, "Diagram3")

    // Available diagrams should still contain all
    availableDiagrams := p.GetAvailableDiagramsNames()
    assert.Contains(t, availableDiagrams, "Diagram1")
    assert.Contains(t, availableDiagrams, "Diagram2")
    assert.Contains(t, availableDiagrams, "Diagram3")
}
```

Figure 30: Code listing snapshot

```go
// Test edge cases for component selection and modification
run test | debug test
func TestComponentSelectionEdgeCases(t *testing.T) {
    p, err := CreateEmptyUMLProject("TestProject")
    assert.NoError(t, err)
    err = p.CreateEmptyUMLDiagram(umldiagram.ClassDiagram, "TestDiagram")
    assert.NoError(t, err)
    err = p.SelectDiagram("TestDiagram")
    assert.NoError(t, err)

    // Test selecting component when no components exist - should not error but not select anything
    err = p.SelectComponent(utils.Point{X: 50, Y: 50})
    assert.NoError(t, err) // No error but nothing selected

    // Add a component and test selecting outside its bounds - should not error
    err = p.AddGadget(component.Class, utils.Point{X: 10, Y: 10}, 0, drawdata.DefaultGadgetColor, "test")
    assert.NoError(t, err)

    err = p.SelectComponent(utils.Point{X: 500, Y: 500})
    assert.NoError(t, err) // No error but nothing selected

    // Test that we can successfully select the component at its correct location
    err = p.SelectComponent(utils.Point{X: 15, Y: 15})
    assert.NoError(t, err)
}
```

Figure 31: Code listing snapshot

```go
// Test file dialog methods without context
run test | debug test
func TestFileDialogsWithoutContext(t *testing.T) {
    p, err := CreateEmptyUMLProject("TestProject")
    assert.NoError(t, err)

    // Test OpenFileDialog without context
    _, err = p.OpenFileDialog()
    assert.Error(t, err)
    assert.Contains(t, err.Error(), "application context not available")

    // Test SaveFileDialog without context
    _, err = p.SaveFileDialog()
    assert.Error(t, err)
    assert.Contains(t, err.Error(), "application context not available")

    // Test SaveDiagramFileDialog without context
    _, err = p.SaveDiagramFileDialog()
    assert.Error(t, err)
    assert.Contains(t, err.Error(), "application context not available")
}
```

Figure 32: Code listing snapshot

```go
// Test various diagram types
run test | debug test
func TestCreateDiagramDifferentTypes(t *testing.T) {
    p, err := CreateEmptyUMLProject("TestProject")
    assert.NoError(t, err)

    // Test creating class diagram (supported)
    err = p.CreateEmptyUMLDiagram(umldiagram.ClassDiagram, "ClassDiagram")
    assert.NoError(t, err)

    // Test creating use case diagram (not supported - should fail)
    err = p.CreateEmptyUMLDiagram(umldiagram.UseCaseDiagram, "UseCaseDiagram")
    assert.Error(t, err)
    assert.Contains(t, err.Error(), "Invalid diagram type")

    // Verify only supported diagram exists
    diagrams := p.GetAvailableDiagramsNames()
    assert.Contains(t, diagrams, "ClassDiagram")
    assert.NotContains(t, diagrams, "UseCaseDiagram")
}
```

Figure 33: Code listing snapshot

16 Misc

16.1 View online

Since the report contains many images, we suggest visiting the GitHub repository to view higher-resolution versions.

蕭耕宏	張庭瑋	黃冠鈞	吳宥駒
25/06/11 21:00 - 23:59	25/06/11 21:00 - 23:59	25/06/11 21:00 - 23:59	25/06/11 21:00 - 23:59
3 hr	3 hr	3 hr	3 hr

References