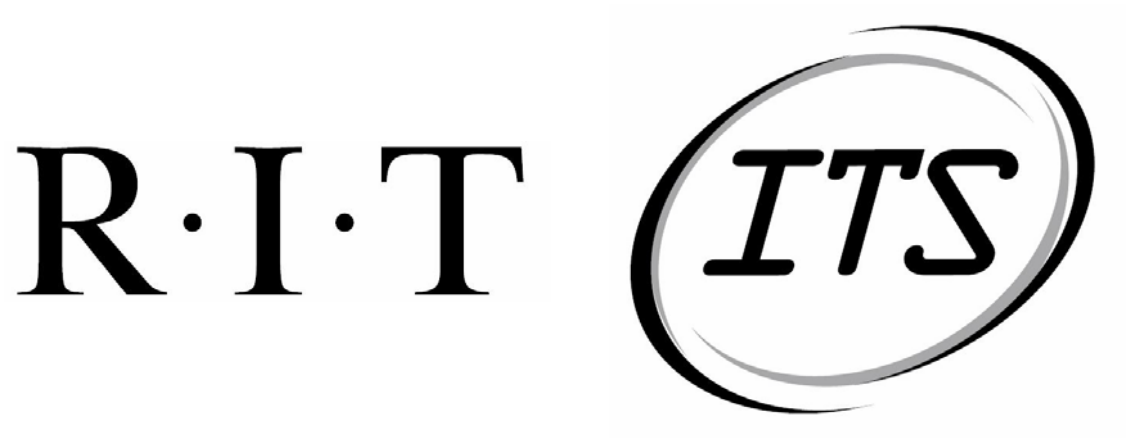


ITS Graphical Report Maker

Detailed-Level Design

07 April 2004

Team JACT Software
RIT Software Engineering Department

Version 1.4.1

Revision History

Revision	Date	Author	Section	Comments/Changes
1.0.0	1 April 04	Train	All	Initial Revision.
1.0.1	2 April 04	Train	2	Added Diagram and possible major components.
1.1.0	2 April 04	Train, Adam	All	Added detailed-level design process. Added component descriptions and services.
1.2.0	4 April 04	Train, Cesario	DL Design	Insert UML Class Diagrams
1.3.0	4 April 04	Train	All	Edit all sections; Add Interaction diagrams.
1.3.1	5 April 04	Myers	Section 2	Typographical Errors
1.4.0	5 April 04	All	DL Design	Update diagrams and descriptions.
1.4.1	7 April 04	Train	Diagrams	Some minor formatting errors.

Table of Contents

1 Document Overview

1.1 Purpose

The purpose of this document is to specify the detailed-level design for the ITS Graphical Report Maker (GRM). This document will act as a map for the implementation of the system. It also provides descriptions for the major components of the GRM system. Sequence diagrams are used to ensure that the design is capable of carrying out the functional requirements of the system.

1.2 Audience

This detailed-level design is intended to be used by members of the development team that will implement the functionality of the GRM. This document will also be used to communicate the detailed-level design and design considerations to the ITS staff members.

- Emilio DiLorenzo, ITS, Director of Technical Support Services
- Mark J. Kimble, ITS, System Management and Tools Technical Support Services
- Patrick Saeva, ITS, Program Manager
- Dr. James Vallino, Faculty Advisor
- Dr. Stephanie Ludi, Assistant Faculty Advisor
- Adam Buehler, Development Team
- Cesario Tam, Development Team
- John Myers, Development Team
- Cheng-Train Chiou, Development Team

1.3 Detailed-Design Process

Once the high-level design was been completed, we focused on developing a detailed-level design. First a general UML module diagram for the server side system was generated. Then the team discussed what design patterns could be used to enhance the general design. Then team decided that factory, strategy and command design patterns were appropriate for use indifferent portions of the general design. A more thorough version of the design was generated including the chosen design patterns. Then the team went through each class in the design and decided upon major functions and attributes. Various parameterized objects were added for intersystem communication. Finally the team used sequence diagrams to verify that the design is capable of satisfying each use case, as specified in the Software Requirements Specification. The process of generating sequence diagrams lead to discovery of design flaws and modifications to the design were made as needed.

2 Detailed-Level Design

2.1 UML Diagram

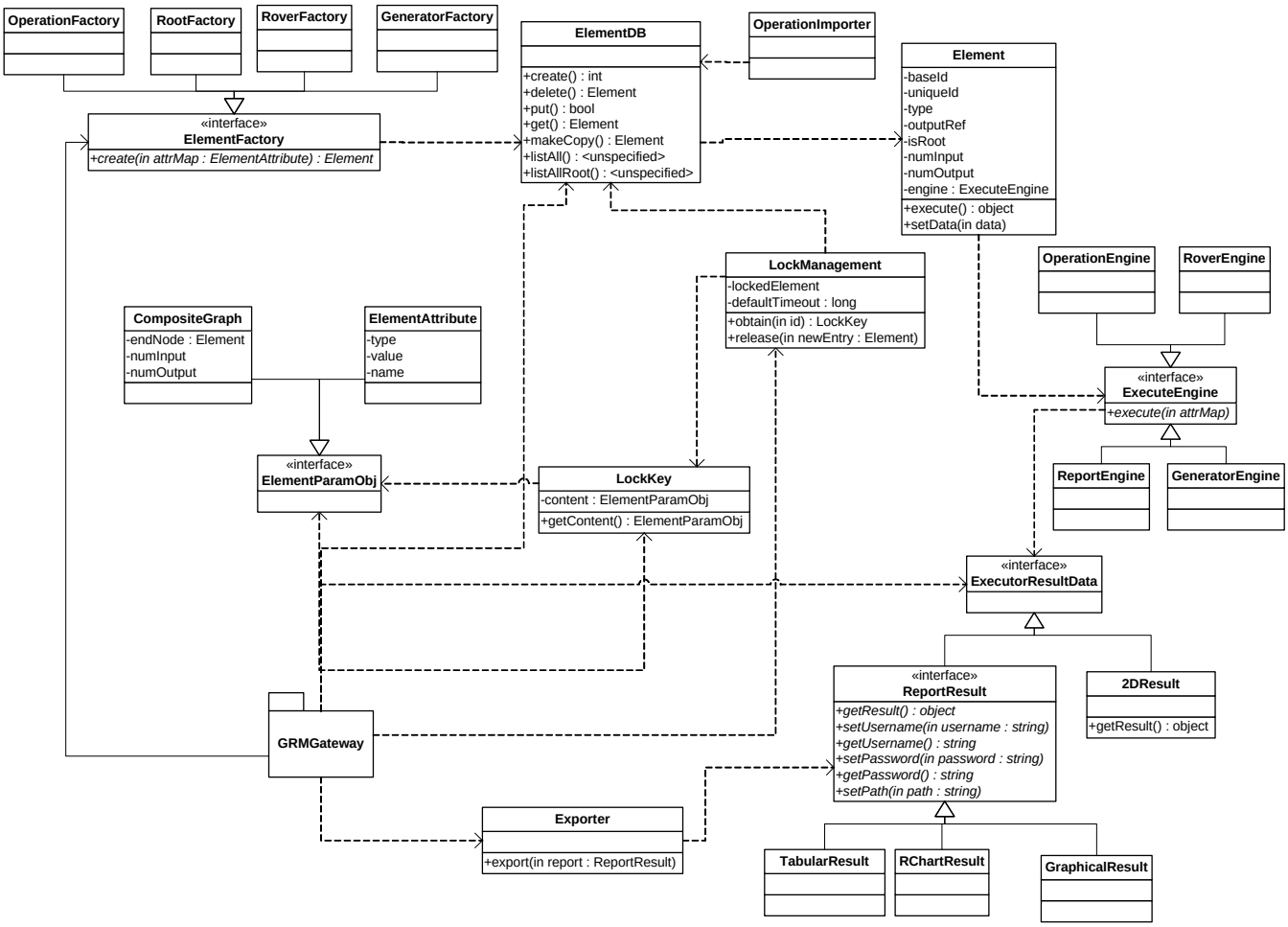


Figure 2.1 – GRM Server Side UML Class Diagram

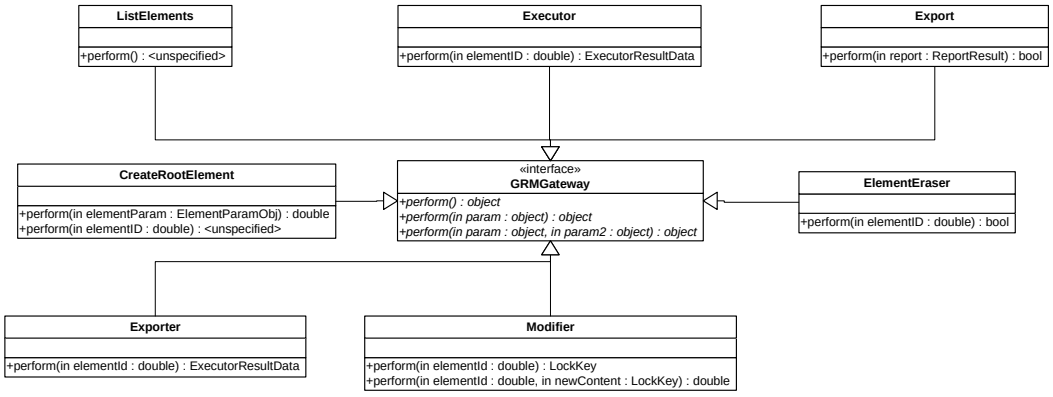


Figure 2.2 – GRM Gateway UML Class Diagram

2.2 Component Descriptions

2.2.1 GRM Gateway

Refer to Figure 2.2

GRM Gateway processes invocations from the client side. The invocations are passed on to one of three primary subcomponents, Creator, Modifier or Element Executor. The Creator is responsible for element creation invocations. The Modifier is responsible for invocations that make modifications to existing elements. The Element Executor is responsible for executing elements and exporting the results of the execution. Command design pattern is used in this component to encapsulate requests as an object, parameterize clients with different requests, queue, and support error checking.

Services:

- Gateway for Thin-Client

2.2.2 Element Persistency

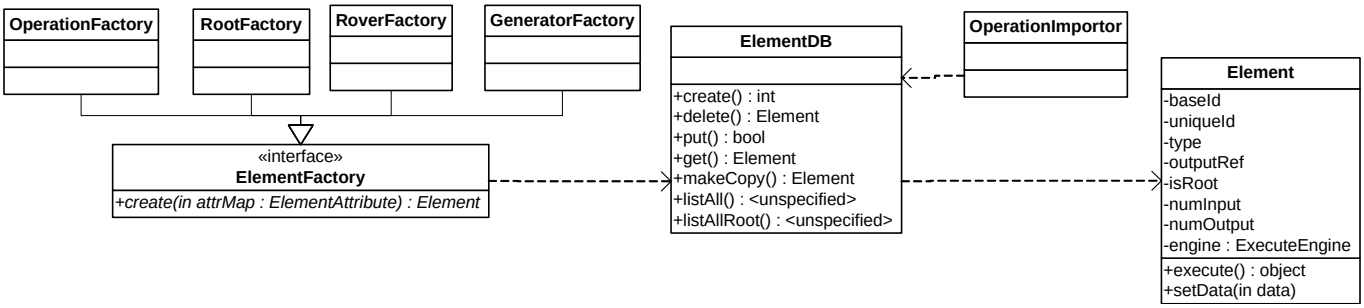


Figure 2.3 – Element Persistency UML Class Diagram

Element Persistency provides services to save, retrieve, and remove elements from the Element DB. The factory design pattern is used to create elements based on the attributes passed from the thin-client. The operation

importer is a dynamic class loader and imports operations into the database. Element DB is a subcomponent that manages exiting elements and provides methods for saving new elements which have already been created. Lock Management is a subcomponent that ensures concurrency within the Element DB by locking an element when it is being modified.

Services:

- Create New Element
- Retrieve Existing Element
- Save Element
- Remove Existing Element
- Import Operation Element Dynamically

2.2.3 Execute Engine

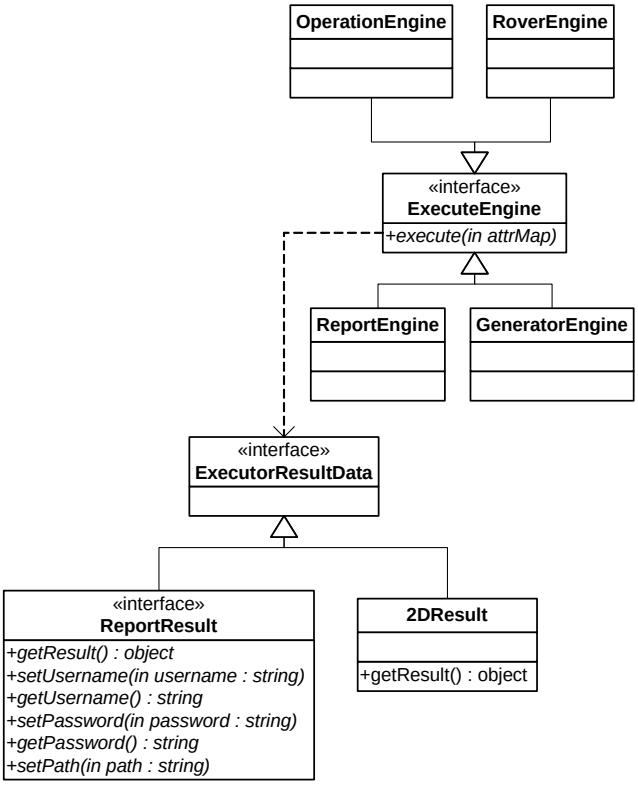


Figure 2.4 – Execute Engine UML Class Diagram

Execute Engine contains execution logic of elements and provides services to execute each type of element. An ExecutorResultData object is generated after each engine's execution. The ExecutorResultData encapsulates the data representation of the result from executions. If the element is a Report Element, the GeneratorEngine produces a ReportResult, while the rest of the engines will produce a 2DResult. The strategy design pattern is used in this component to encapsulate all the engines' algorithms and allow them to be interchangeable, varying depending on the type of element to be executed.

Services:

- Execute Element

2.2.4 Exporter

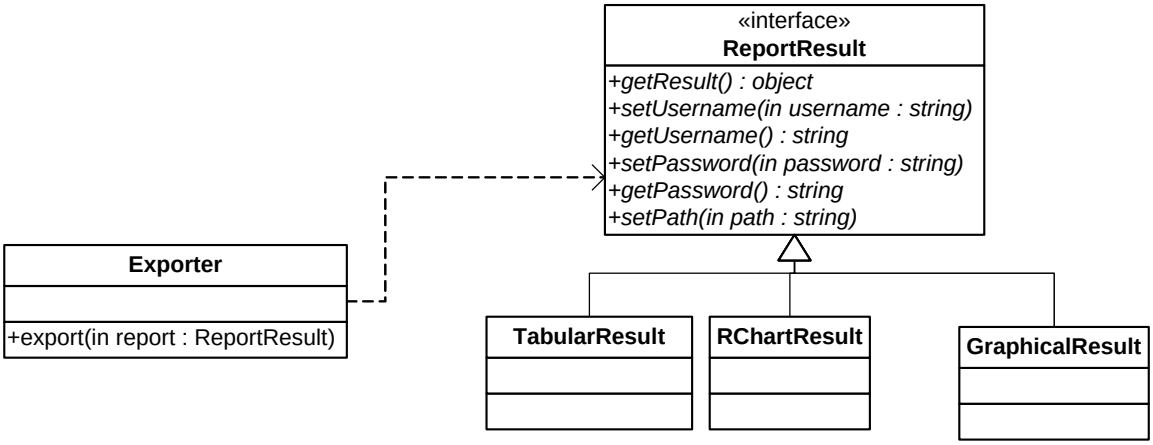


Figure 2.5 – Exporter UML Class Diagram

Exporter provides services to export report data into an external file system. A ReportResult object is used to define the report data, path to the external file system, and the necessary authentication information needed for accessing the file system. Concrete implementations of the ReportResult are used to encapsulate the format of the report data.

Services:

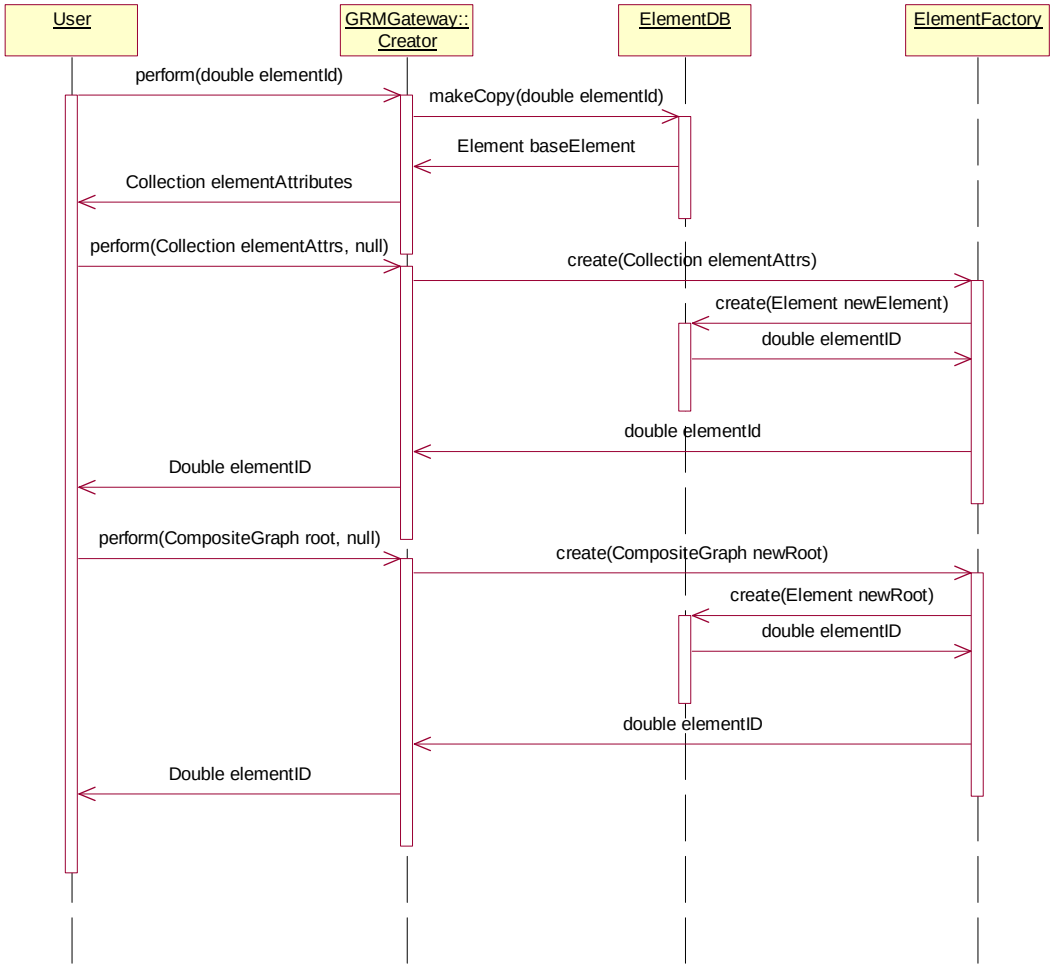
- Export Result

2.2.5 Thin-Client

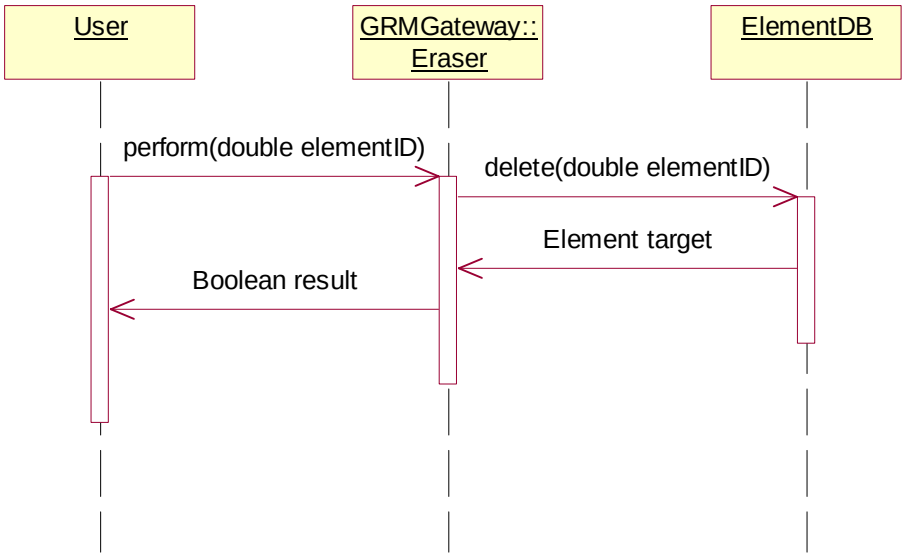
The Thin-Client serves as the presentation layer of the GRM system that also manages user inputs.

3 Sequence Diagrams

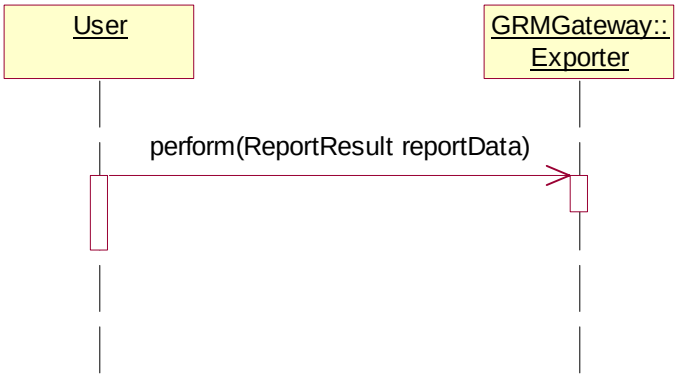
3.1 Create Element



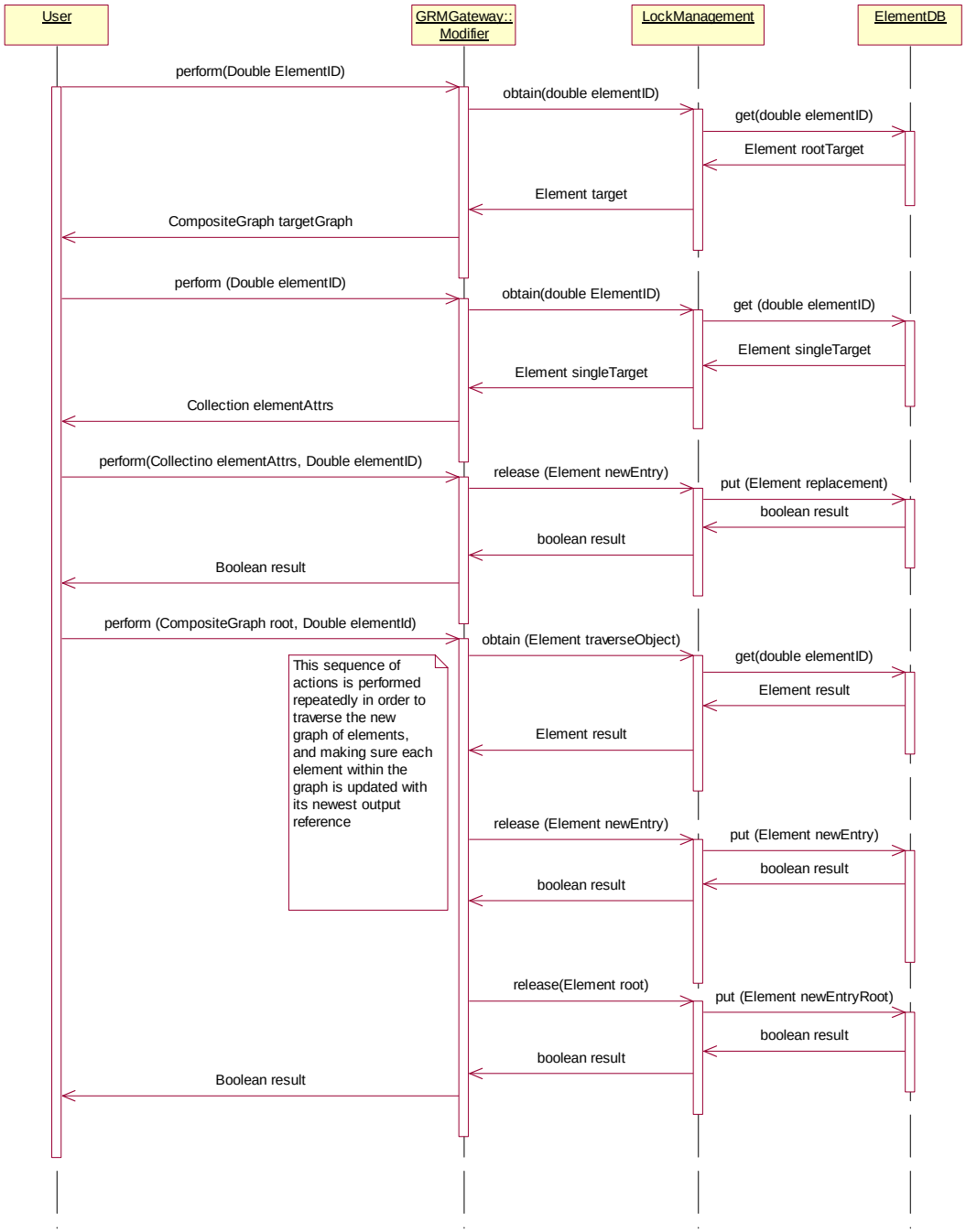
3.2 Delete Element



3.3 Export Report



3.4 Modify Element



3.5 Preview Element