

Three Quarters Native

Grain Elevator System

Project Map

Thomas Slade - Anthony Koester - Johannes Astner - Steven Melcher

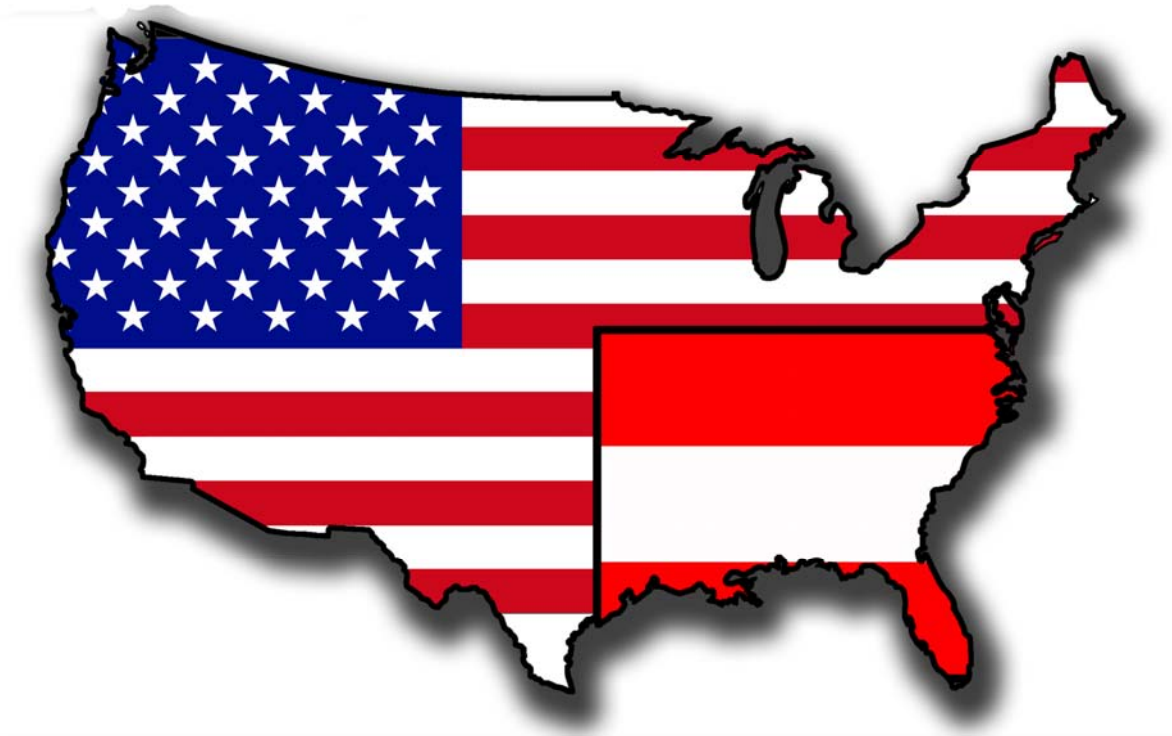


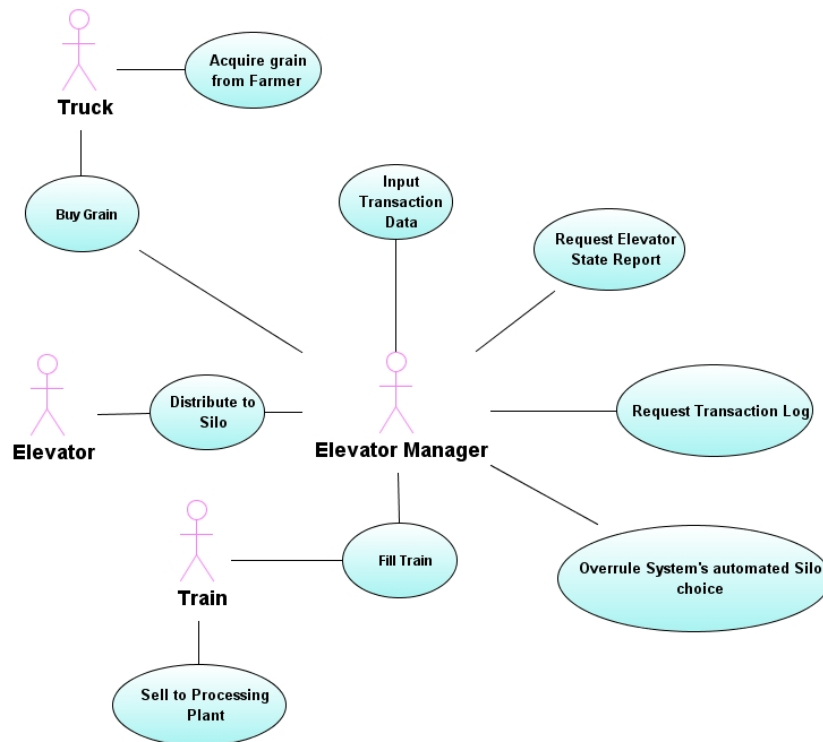
Table of Content

UCM – Use Case Model.....	4
Use Case Diagram	4
Actor Briefs	4
Use Case Descriptions	5
Use Case 1: Buy Grain	5
Use Case 2: Input Transaction Data	6
Use Case 3: Request Elevator State Report	7
Use Case 4: Distribute to Silo	7
Use Case 5: Fill Train	9
Use Case 6: Request Transaction Log	10
Design Rationale	11
SAD – Software Architectural Design.....	12
Grain Elevator – Conceptual Model.....	12
Grain Elevator – Architecture One Decomposition	13
Grain Elevator – Architecture One Class Model	14
Grain Elevator – Architecture Two Decomposition	15
Grain Elevator – Architecture Two Class Model	16
Utility Tree	17
Scenarios	18
Reliability.....	18
Usage.....	18
Scenarios - continued.....	18

Hardware Adaptability	18
Maintainability	19
Performance	19
Scoring Matrix	20
Scenario Evaluation.....	21
Never lose data	21
System distributes correctly	21
Design Rationale	22
DDD – Detailed Design Document	23
Class Diagrams and Module Responsibilities	23
User Interface	23
System Class.....	25
Elevator Interface.....	26
Central Database Class.....	28
Sequence Diagrams.....	29
setData	29
getReport	30
getTransactionLog.....	30
State Diagrams	31
Elevator Interface State Diagram.....	31
User Interface State Diagram.....	32
Prototype	33

UCM – Use Case Model

Use Case Diagram



Actor Briefs

Truck: The person who delivers/sells grain to the Elevator.

Elevator: The hardware that distributes the grain into the silos.

Elevator Manager: The supervisor over the Elevator, and the transactions between buying and selling grain.

Train: The person who buys grain from the Elevator Manager, and delivers/sells it to the Processing Plants.

Use Case Descriptions

Use Case 1: Buy Grain

Actors: Elevator Manager, Truck

Stakeholders and Needs:

Elevator Manager—Buy Grain and sell for profit

Truck—Sell Grain for a set price

Preconditions: Space is available in the correct silo. Grain is of the correct type and grain for what the Elevator Manager needs to buy.

Postconditions: Grain is in the Elevator.

Trigger: Truck arrives with grain.

Basic Flow:

1. Truck arrives with grain.
2. Elevator manager buys grain.
3. Grain is loaded onto elevator.

Extensions: NONE

Use Case 2: Input Transaction Data

Actors: Elevator Manager

Stakeholders and Needs:

Elevator Manager—to input data concerning transactions

Farmers—wants consistent data between elevator manager and truck

Train—wants consistent data between elevator manager and train

Preconditions: Grain has arrived. Grain has been bought or sold.

Postconditions: Correct data has been entered and stored in database.

Trigger: Grain has been bought or sold by elevator manager.

Basic Flow:

1. Elevator manager buys or sells grain.
2. Data concerning transaction is entered and stored into database.
3. Correct data has been entered and stored in the database.

Extensions:

2a Elevator manager inputs wrong data.

2a1. System alerts elevator manager of inconsistent data, then the use case continues at point 2.

Use Case 3: Request Elevator State Report

Actors: Elevator Manager

Stakeholders and Needs:

Elevator Manager—wants to view current state report

Preconditions: System is in ready state

Postconditions: Current state report is outputted. The system is in ready state once again.

Trigger: Elevator Manager requests state report

Basic Flow:

1. Elevator manager requests state report
2. Correct report is generated

Extensions:

1a Report is not generated.

1a1. Database failure message.

1a2. Report generation aborted.

2a Report is faulty.

2a1. Incorrect data is inputted.

2a2. Report generation aborted.

Use Case 4: Distribute to Silo

Actors: Elevator Manager

Stakeholders and Needs:

Elevator Manager—wants to distribute the incoming grain in an appropriate silo

Preconditions:

Grain is loaded onto the elevator.

System is in ready state.

Postconditions:

Grain is loaded into silos.

Trigger: Elevator manager informs the system of the type of grain, its grade, and its quantity.

Basic Flow:

1. Elevator manager informs the system of the type of grain, its grade, and its quantity.
2. The system automatically is looking for appropriate silos.
3. The system notifies the elevator manager of its choice.
4. The elevator manager accepts the choice.
5. The elevator distributes the grain in the chosen silo(s).
6. Grain is successfully loaded into the silo(s).\

Extensions:

1a The grain type is unknown

1a1. Stop the distributing process.

1a2. No grain is loaded into silo(s).

2a The system does not find an appropriate silo.

2a1. Continue with point 3a1.

3a The elevator manager overrules the system's choice.

3a1. The elevator inputs his choice.

3a2. Continue with point 5.

5a Silo is full.

5a1. Continue with point 3a1.

5b All silos are full or no space for the grain.

5b1. The system alarms the elevator manager.

5b2. Use case ends.

Use Case 5: Fill Train

Actors: Elevator Manager, Train

Stakeholders and Needs:

Elevator Manager- To ensure a recorded and successful transfer of the purchased grain to the train

Train- Authorizes the purchase of the grain and the filling of the train with the grain from the silo

Preconditions:

There is grain to be sold.

Postconditions: The train is filled with grain from the elevator system.

Trigger: The elevator manager initiates the filling of the train with grain from the elevator system.

Basic Flow:

1. The elevator manager sells the grain to the train owner.
2. The train owner authorizes the filling of the train with the purchased grain and waits for the train to be filled.
3. The elevator manager tells the system how much grain is removed from the silo, which rail cars have been loaded, who the train conductor is, and who the buyer is.
4. The system acknowledges that the data has been successfully stored.
5. Grain is transported from the silo to the train's rail cars via the elevator system.

Extensions:

3a The input to the system is invalid .

3a1. The system notifies the Elevator manager that his/her input was invalid

or missing data and the use case resumes at step 3.

5a The rail cars are full, but there is more grain that was supposed to be transported to the train.

5a1. The elevator manager is alerted to the amount of grain left in the silo and how much to refund based on the remaining amount.

Use Case 6: Request Transaction Log

Actors: Elevator Manager

Stakeholders and Needs:

Elevator Manager- Needs a chronological listing of the transaction log

Preconditions: System is in ready state. Grain has been bought or sold.

Postconditions: System outputs a chronological listing of the transaction log.

Trigger: Elevator Manager requests transaction log.

Basic Flow:

1. Elevator manager requests transaction log.
2. Correct log is generated in chronological order.

Extensions:

1a Transaction log is not generated.

1a1. Database failure.

2a Transaction log is faulty.

2a1. Incorrect data is inputted.

Design Rationale

We chose to make a simple use case diagram to abstract unnecessary details that can hinder one from seeing how the Grain Elevator System works. Our design is easy and accommodates all requirements as discussed in the SRS.

We chose to lump together some of the “characters” that were described in the SRS. For instance, the actor *Truck* is actually the truck filled with the grain, the driver of the truck, and the seller on truck. This is also the same for the *Train*. The *Elevator* represents all of the systems computing functionality. The main actor in the system is the *Elevator Manager*.

When a *Truck* arrives with grain, the *Elevator Manager* will buy the grain from the *Truck*. The *Elevator Manager* then inputs the transaction data into the system and allows the *Elevator* to distribute to a silo at his/her consent. The *Elevator Manager* has the choice at any time to override the *Elevator’s* decision, to avoid unwanted failures. The silo that is chosen to fill with the grain is based upon many factors that are constantly monitored by the *Elevator* and the *Elevator Manager*. The decision is a joint decision that helps make the system full proof. The *Elevator Manager* can also request a transaction log or request the elevator’s state report at any time.

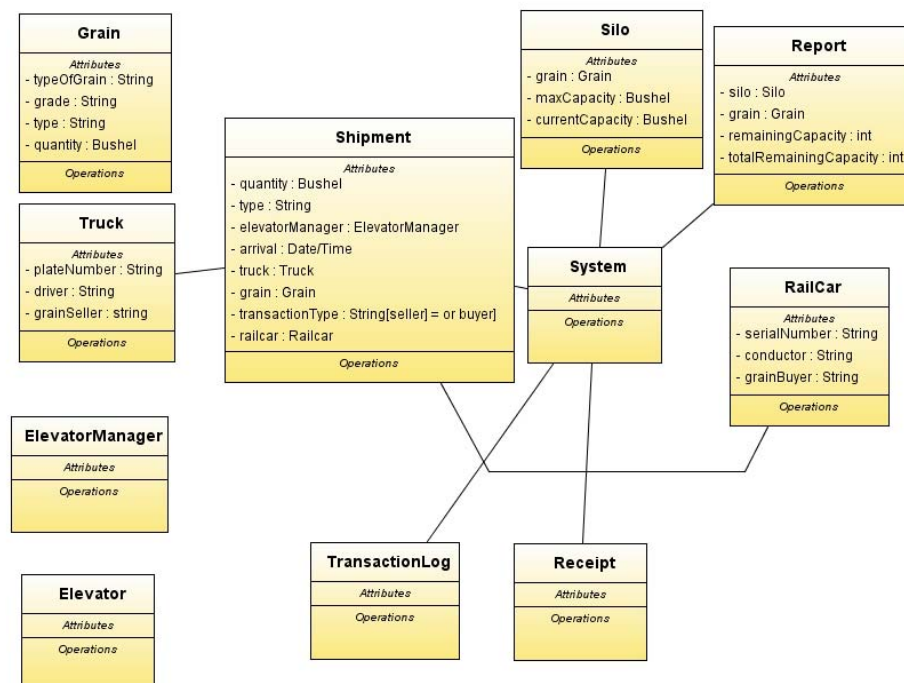
When the *Train* arrives, the *Elevator Manager* will authorize the transaction, input that transaction data, and fill the train with the agreed upon amount and type of grain.

This sums up how our Grain Elevator System works; we chose to implement a very simple use case diagram as to not clutter ourselves with unnecessary details that will be dealt with in the near future.

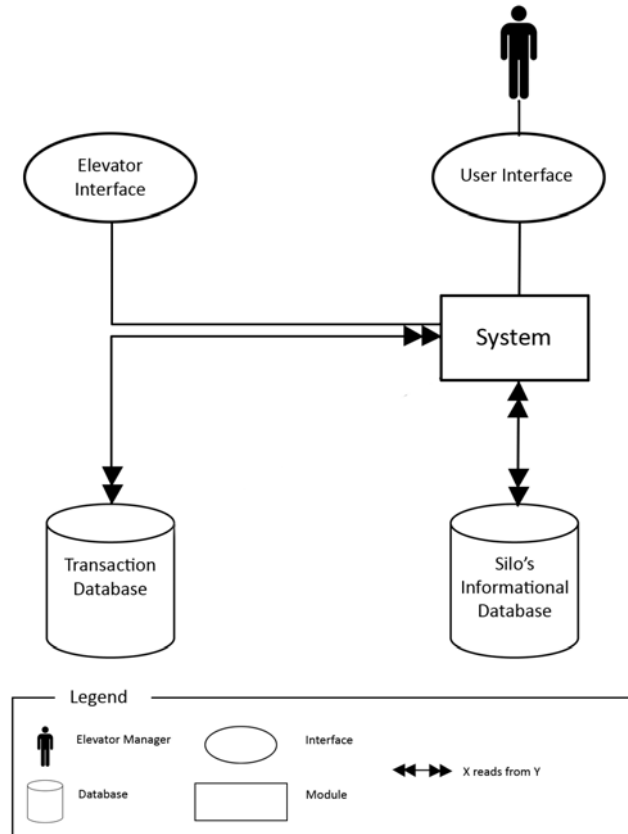
SAD – Software Architectural Design

Grain Elevator – Conceptual Model

Noun Phrases	Comment
Central storage elevator, rail car, truck, transaction log, shipment, elevator manager, silo, receipt, report, grain	Concepts
Type of grain, grade, quantity	Attributes of grain
Quantity (bushel), type of grain, grade, time and date, elevator manager, arrival or departure, truck or rail car identifier, driver or conductor, seller or buyer	Attributes of shipment
Silo, type of grain, amount, remaining capacity, total remaining capacity	Attributes of report
Plate number, driver, grain seller	Attributes of truck
Serial number, conductor, grain buyer	Attributes of rail car
Farms, processing plants, SW system, accounting, place, room	Irrelevant noun phrases (parts of explanations, examples, etc.)
Grower, load, system's choice, wheat, barley, rice, oats, hop	Indirect connection to program

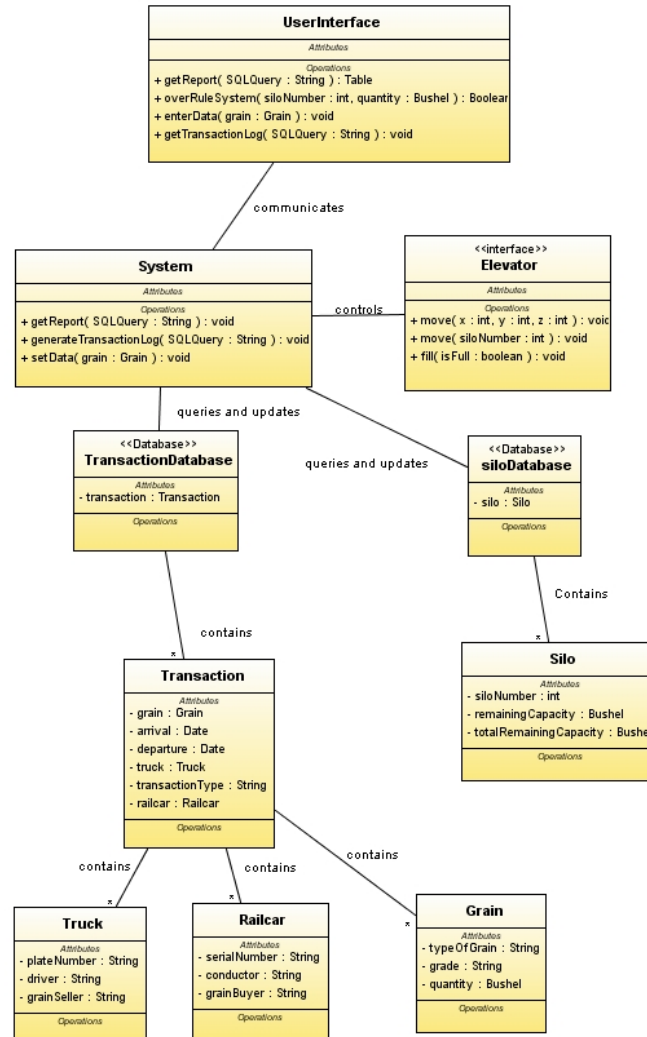


Grain Elevator – Architecture One Decomposition

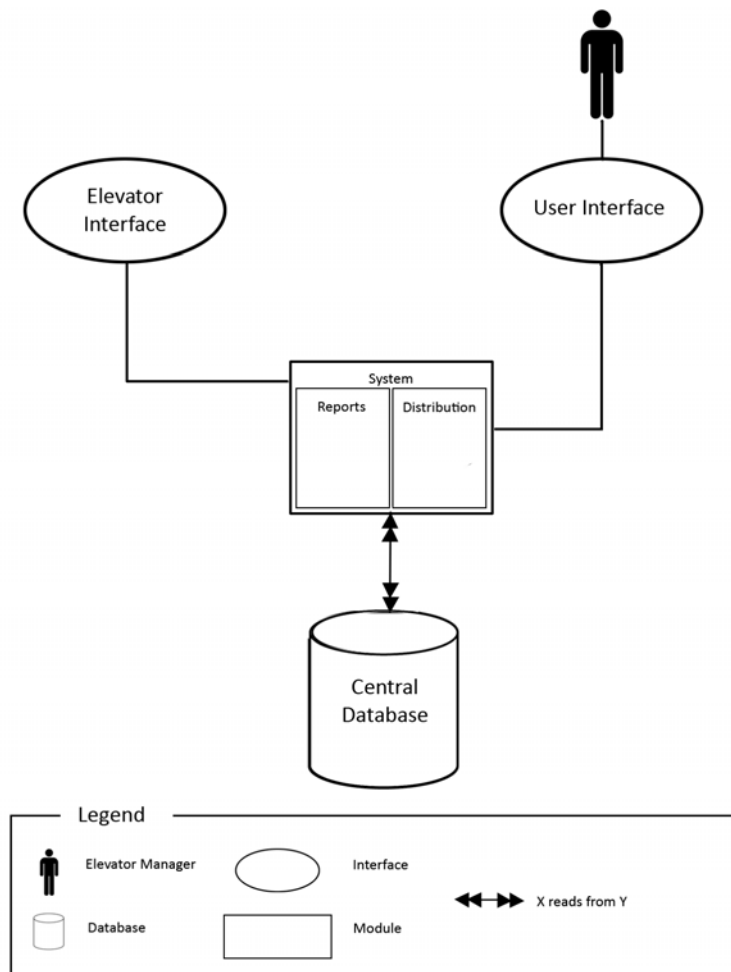


Component	Responsibilities
Elevator Interface	<ul style="list-style-type: none"> Provides an interface between the elevator and the system
User Interface	<ul style="list-style-type: none"> Provides an interface between the user and the system
Transaction Database	<ul style="list-style-type: none"> Stores the data about transactions
Silo's Informational Database	<ul style="list-style-type: none"> Stores the data about storage silos, grain information, and errors
System	<ul style="list-style-type: none"> Distributing the grain Generates reports and transaction logs

Grain Elevator – Architecture One Class Model

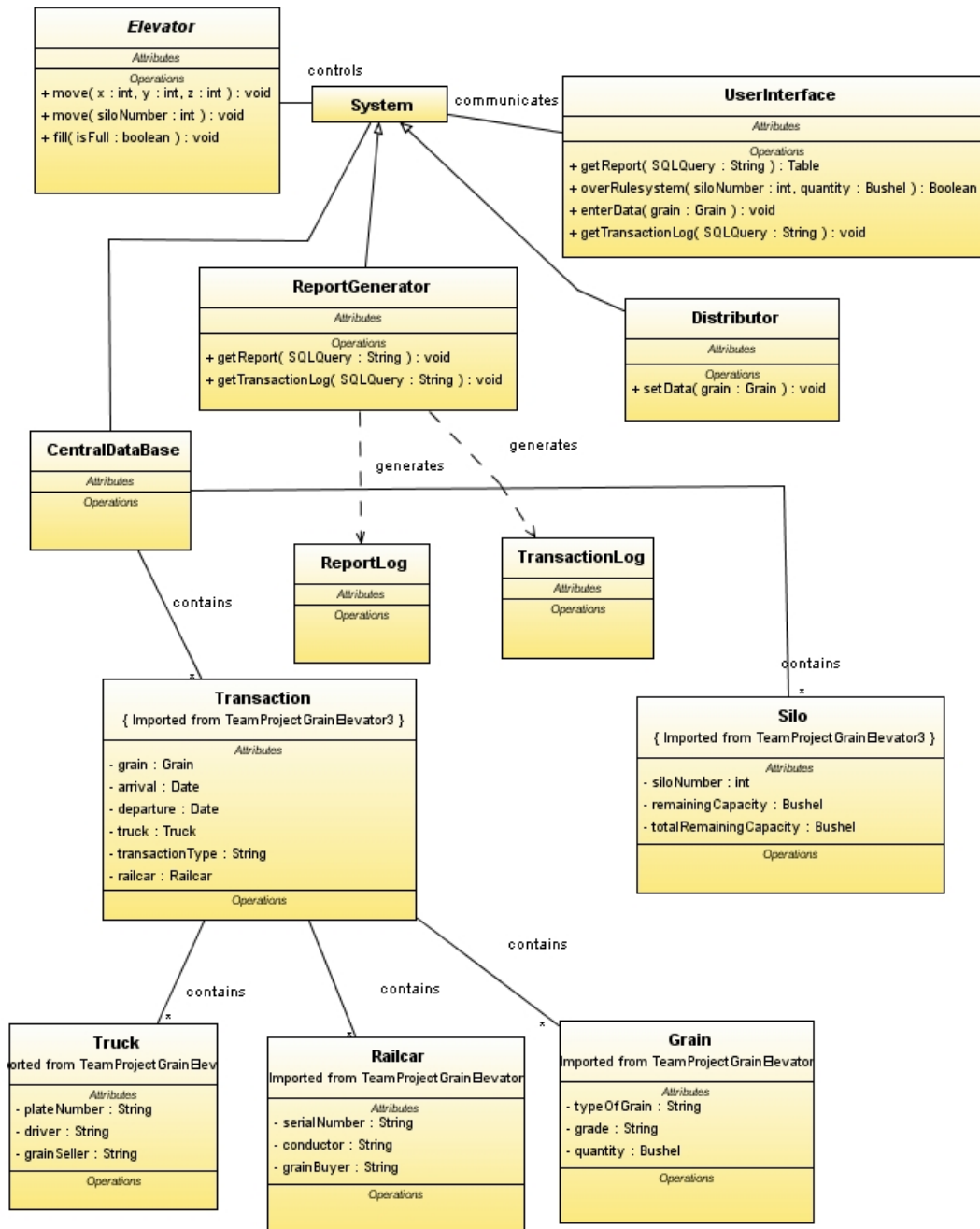


Grain Elevator – Architecture Two Decomposition



Component	Responsibilities
Elevator Interface	<ul style="list-style-type: none"> Provides an interface between the elevator and the system
User Interface	<ul style="list-style-type: none"> Provides an interface between the user and the system
Central Database	<ul style="list-style-type: none"> Stores the data about transactions, storage silos, grain information, and errors
Reports	<ul style="list-style-type: none"> Generates reports and transaction logs
Distribution	<ul style="list-style-type: none"> Distributing the grain

Grain Elevator – Architecture Two Class Model



Utility Tree

- Utility Tree
 - Reliability
 - Never lose data saved to the DB (H)
 - 24 hour access (L)
 - System distributes correctly (H)
 - Usage
 - Easy to operate (M)
 - Distribute manually (H)
 - Distribute automatically (H)
 - Hardware Adaptability
 - Add new silo (M)
 - Change elevator (L)
 - Store a different type of material (L)
 - Maintainability
 - Query an error log (M)
 - Performance
 - System can automatically sell (distribute) to rail car in 10 minutes (M)

Scenarios

Reliability

Never Lose Data saved in the Database—The elevator manager enters a new transaction into the system via the user interface, when suddenly the power goes out. The elevator manager remains calm, because he knows that the database utilizes an “auto back-up” mechanism that periodically backs up the database. He also knows that the appending information on the database is done via an “update” query that automatically hard codes the information into the database table. The database is stored on a local machine and will be accessible when the power goes back on.

24 Hour Access—The elevator manager gets a call from his boss at three o’clock in the morning. His boss says that a truck has some grain that he wants to “get rid of” as soon as possible. The grain will be cheaper if the elevator manager is willing to accept the offer. The elevator manager decides to take the offer and arrives at the elevator. He starts up the system, coordinates the transaction, and distributes the grain.

System distributes correctly – The elevator manager enters the data and the system starts to automatically distribute 3000 bushels of wheat of grade high in silo #3. After completion the system notifies the elevator manager.

Usage

Easy to operate – The elevator manager just inputs the data about the grain and the system automatically starts the distribution process. The elevator manager only has to enter the type of grain, quantity in bushels, and the grade of grain.

Distribute automatically—A truck arrives with a shipment of grain. The elevator manager tells the system about the grain’s type, grade, and quantity. Based on this information, the system distributes the grain into the appropriate silo(s).

Distribute manually— A truck arrives with a shipment of grain. The elevator manager tells the system about the grain’s type, grade, and quantity. The elevator manager decides that there is a specific silo that he wants to store the grain in. The system is overruled by the elevator manager as he inputs into the system which silo(s) he wants to store the grain in.

Scenarios - continued

Hardware Adaptability

Add a new silo- A new silo is added to the Grain Elevator System. The software allows the elevator manager to add the new silo to the system so it can be used. This takes the elevator manager less than an hour to configure in the system.

Change elevator- With larger quantities of grain being bought and sold, it was decided that the elevator needed to be replaced with a larger and faster model. It took about 24 hours to install the new elevator and configure it with the Grain Elevator System which supports multiple types of elevators.

Store a different type of material- The elevator manager is now buying and selling bird feed in addition to grain. The information about the bird feed is added into the Grain Elevator System so that it can be stored in the silos.

Maintainability

Query an error log- The Elevator Manager receives an order to fill 1500 bushels of wheat into Railcar 108. When asked, the system tells him that 1900 bushels of wheat are available in Silo#3. When the Elevator Manager tries to fill this order there are only 400 bushels available. The Elevator Manager queries an error log from the system to find out why there is a discrepancy in the amount of wheat stored.

Performance

System can automatically sell (distribute) to rail car in 10 minutes- Railcar 232 arrives to pickup 2000 bushels of oats. The Elevator Manager should be able to tell the system how much grain is removed, from what silo, and into which railcar. The system should have the oats ready for the loading of the railcar in 10 minutes or less.

Scoring Matrix

Scenarios		Decomposition One		Decomposition Two	
Description	Weight	Rating	Score	Rating	Score
Reliability					
Never lose data saved to the DB (H)	0.130435	4	0.521739	5	0.652174
24 hour access (L)	0.043478	3	0.130435	3	0.130435
System distributes correctly (H)	0.130435	3	0.391304	5	0.652174
Usage					
Easy to operate (M)	0.086957	3	0.26087	3	0.26087
Distribute manually (H)	0.130435	3	0.391304	3	0.391304
Distribute automatically (H)	0.130435	3	0.391304	3	0.391304
Hardware Adaptability					
Add new silo (M)	0.086957	3	0.26087	3	0.26087
Change elevator (L)	0.043478	5	0.217391	5	0.217391
Store a different type of material (L)	0.043478	5	0.217391	5	0.217391
Maintainability					
Query an error log (M)	0.086957	2	0.173913	5	0.434783
Performance					
Sell (distribute) to rail car in 10 minutes (M)	0.086957	3	0.26087	3	0.26087
		3.217391		<u>3.869565</u>	

Scenario Evaluation

Never lose data

Architecture One- Has a transaction database and a silo's informational database. When the elevator manager enters a new transaction, the transaction is stored in the transactional database. Once the transaction has been completed, the grain is then distributed to the correct silo pending verification from the system and the elevator manager. The data is then updated in the silo informational database. Architecture One is more complicated because it has two databases and is thus a two-step process.

Architecture Two- Has only a central database that is a combination of the transaction database and the silo's informational database. When the elevator manager enters any transaction, the information concerning the silos is automatically updated using the transaction information pending elevator manager verification. This architecture is preferred over Architecture One.

System distributes correctly

Architecture One- The elevator interface provides a means of communication between the system and the elevator. Having two separate databases calls for two separate queries anytime the elevator is used to store or distribute grain.

Architecture Two- The elevator interface communicates with the system's distribution component that then queries the central database. The central database offers much faster and more accurate communication between the elevator interface and the system.

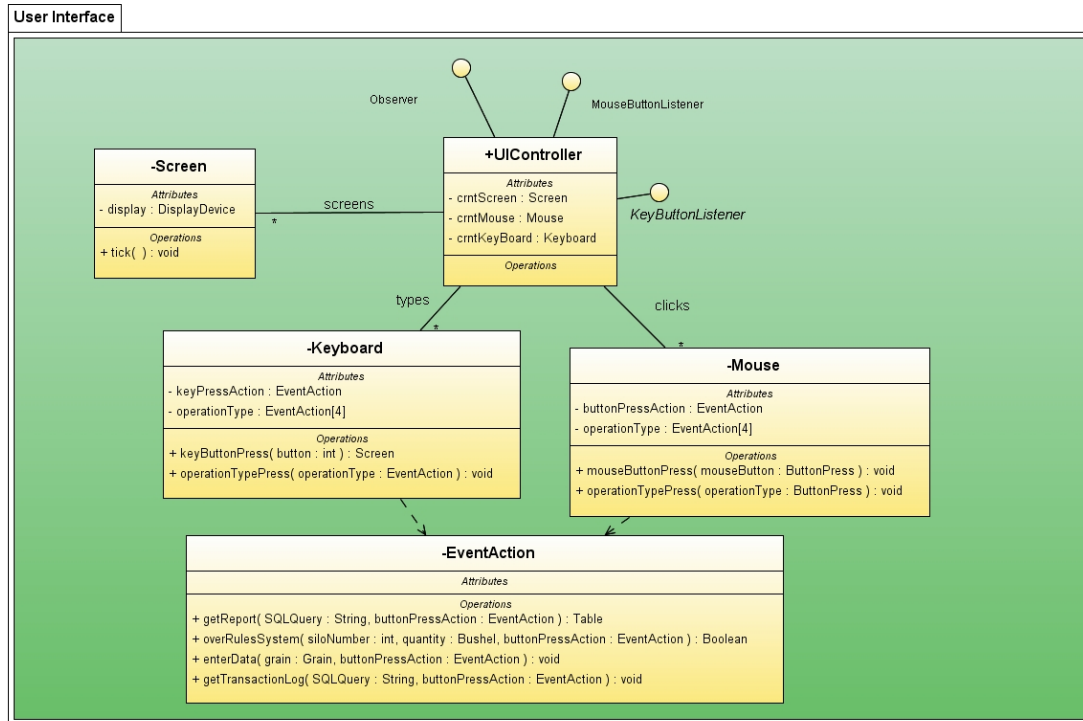
Design Rationale

We have chosen Architecture Two. Architecture Two provides fast and accurate communication between the elevator interface and the system. It is easier to change because we split the system component into two components each with their own responsibility. The architecture has a better score in the scoring matrix based on its reliability, adaptability, and maintainability.

DDD – Detailed Design Document

Class Diagrams and Module Responsibilities

User Interface



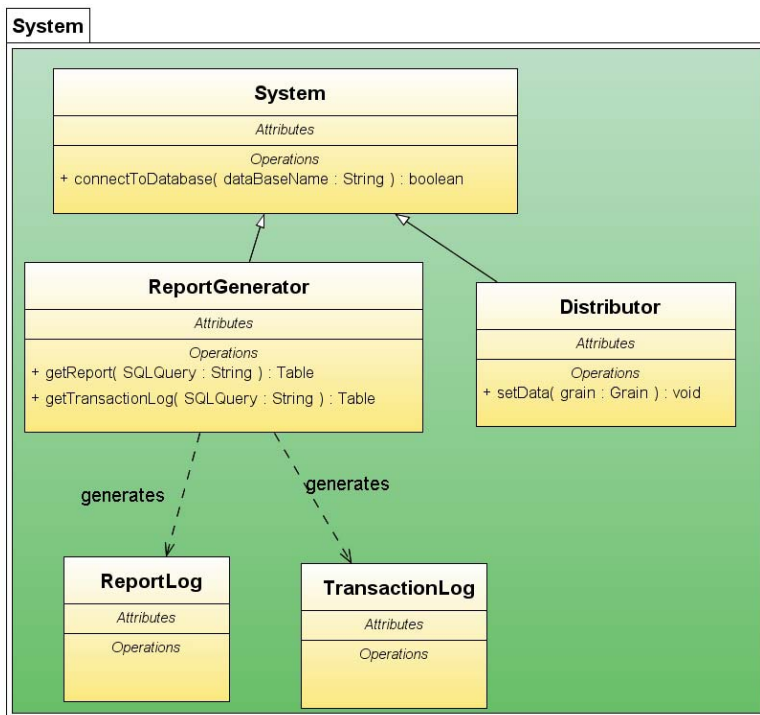
User Interface Layer Module Responsibilities

Module	Responsibilities
Screen	Displays user's input and clock ticks (routed to it from the UIController.)
Keyboard	Controls the user's input and handles keyButtonPress actions and operationTypePress actions.
Mouse	Controls the user's mouse constantly keeping the cursor on the screen. It handles mouseButtonPress actions and operationTypePress actions.
EventAction	Executes commands depending on operationType specified by user. Operations are then sent to the system to be processed.

User Interface Layer Module Interface Specifications

Notify Screen that a button has been pressed.	<i>Syntax:</i>	KeyButtonPress(button: int) : Screen
	<i>Pre:</i>	None.
	<i>Post:</i>	Screen refreshes with user input displayed.
Notify Screen that a button has been pressed and call EventAction.	<i>Syntax:</i>	operationTypePress(operationType : EventAction) : Screen
	<i>Pre:</i>	None.
	<i>Post:</i>	Screen refreshes with user's chosen option.
Notify Screen that a button has been pressed and call EventAction.	<i>Syntax:</i>	mouseButtonPress(mouseButton : ButtonPress)
	<i>Pre:</i>	None.
	<i>Post:</i>	Screen refreshed with cursor or option chosen.
Notify Screen that a button has been pressed and call EventAction.	<i>Syntax:</i>	operationTypePress(operationType : ButtonPress)
	<i>Pre:</i>	None.
	<i>Post:</i>	Screen refreshes with user's chosen option.
Activate Screen when is turned on.	<i>Syntax:</i>	activate()
	<i>Pre:</i>	None:
	<i>Post:</i>	Screen turns on.
Activate screen when a micro second has passed.	<i>Syntax:</i>	tick()
	<i>Pre:</i>	None:
	<i>Post:</i>	Screen is refreshed.

System Class



System Layer Module Responsibilities

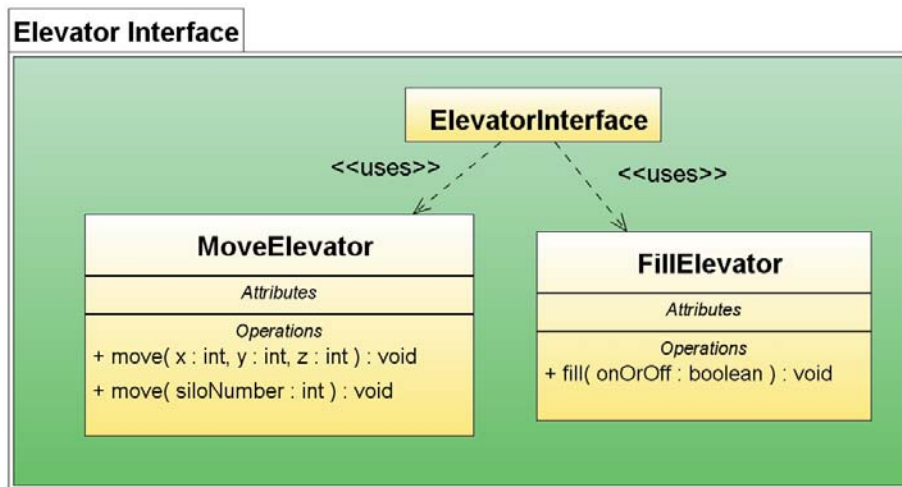
Module	Responsibilities
System	Connects to the database
Report Generator	Inherits from the System and gets Reports and Transaction Logs.
Distributor	Inherits from the System and sets data for Reports
ReportLog	No responsibilities. Only contains data.
TransactionLog	No responsibilities. Only contains data.

System Layer Module Specifications

Connects to database so that operations can be performed on it.	<i>Syntax:</i>	connectToDatabase (dataBaseName : String) Boolean
	<i>Pre:</i>	None.
	<i>Post:</i>	Either connected or not connected to database

Gets the report from the database.	<i>Syntax:</i>	getReport(SQL Query : String) : Table
	<i>Pre:</i>	None.
	<i>Post:</i>	Report (table) is returned.
Gets the transaction log from the database.	<i>Syntax:</i>	getTransactionLog(SQL Query : String) : Table
	<i>Pre:</i>	None.
	<i>Post:</i>	Transaction Log (table) is returned.
Appends/Modifies the data in the database.	<i>Syntax:</i>	setData(grain : Grain) : void
	<i>Pre:</i>	Transaction has occurred.
	<i>Post:</i>	Data has been stored in database.

Elevator Interface



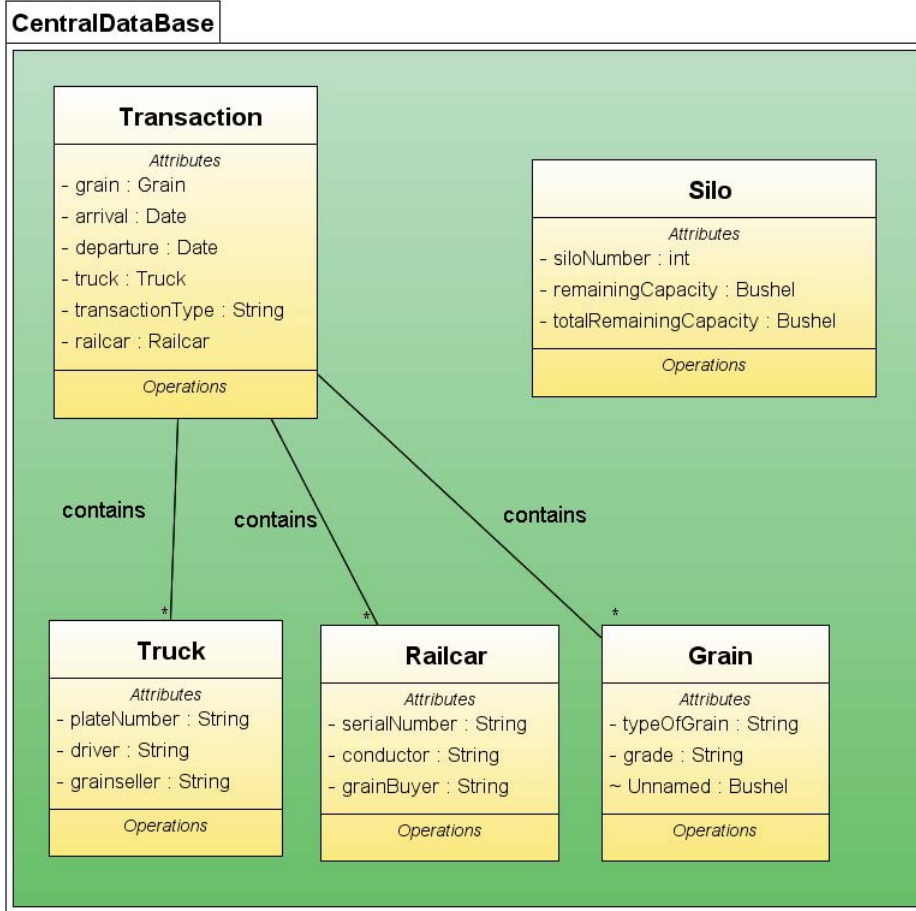
Elevator Interface Layer Module Responsibilities

Module	Responsibilities
MoveElevator	
Fill Elevator	Turn elevator on or off when fill is called.

Elevator Interface Layer Module Interface Specifications

Moves elevator when move is called using three parameters: x, y, and z.	<i>Syntax:</i>	Move(x : int, y: int, z: int)
	<i>Pre:</i>	None.
	<i>Post:</i>	Elevator moves to desired location.
Moves elevator when move is called using siloNumber integer.	<i>Syntax:</i>	Move(siloNumber: int)
	<i>Pre:</i>	None.
	<i>Post:</i>	Elevator moves to desired silo.
Turns elevator on when onOrOff is set to true; turns elevator off when onOrOff is set to false.	<i>Syntax:</i>	Fill(onOrOff : boolean)
	<i>Pre:</i>	None.
	<i>Post:</i>	Elevator is turned on or off.

Central Database Class

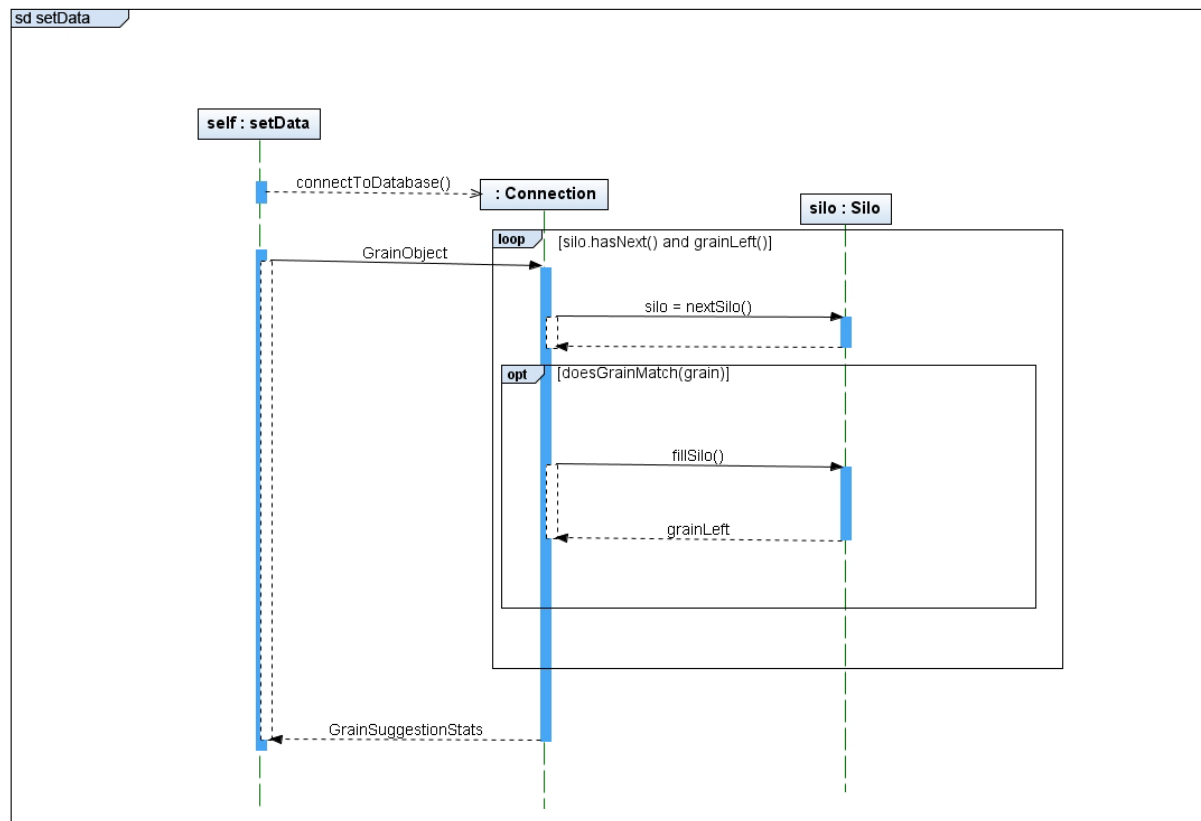


Central Database Layer Module Responsibilities

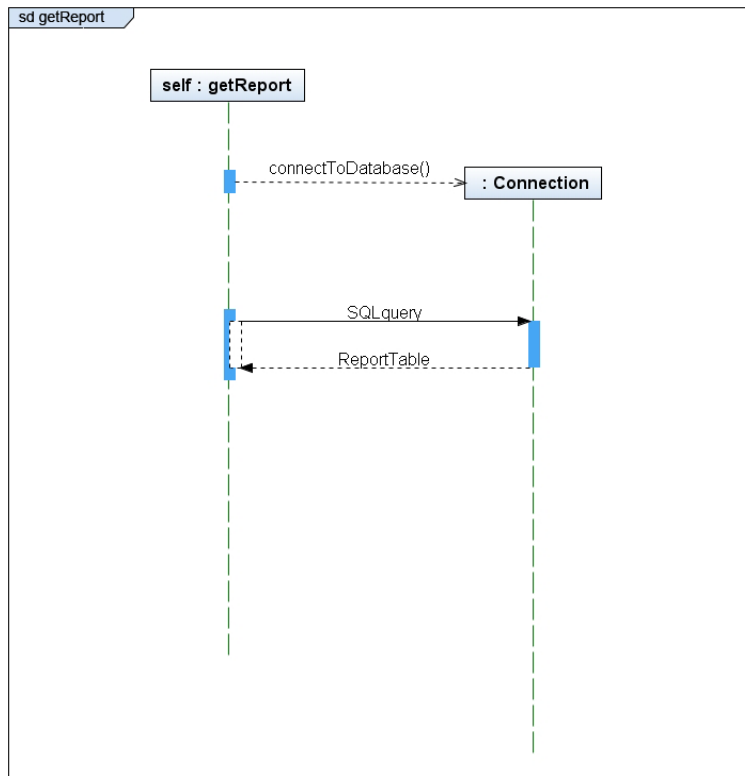
Module	Responsibilities
Transaction	A table that contains the information about each transaction.
Silo	A table that contains the information about each silo.
Truck	Contains information for the transaction including: driver, grainseller, and plateNumber.
Railcar	Contains information for the transaction including serialNumber, conductor, and grainBuyer.
Grain	Contains information for the transaction including typeOfGrain, grade, and Unnamed.

Sequence Diagrams

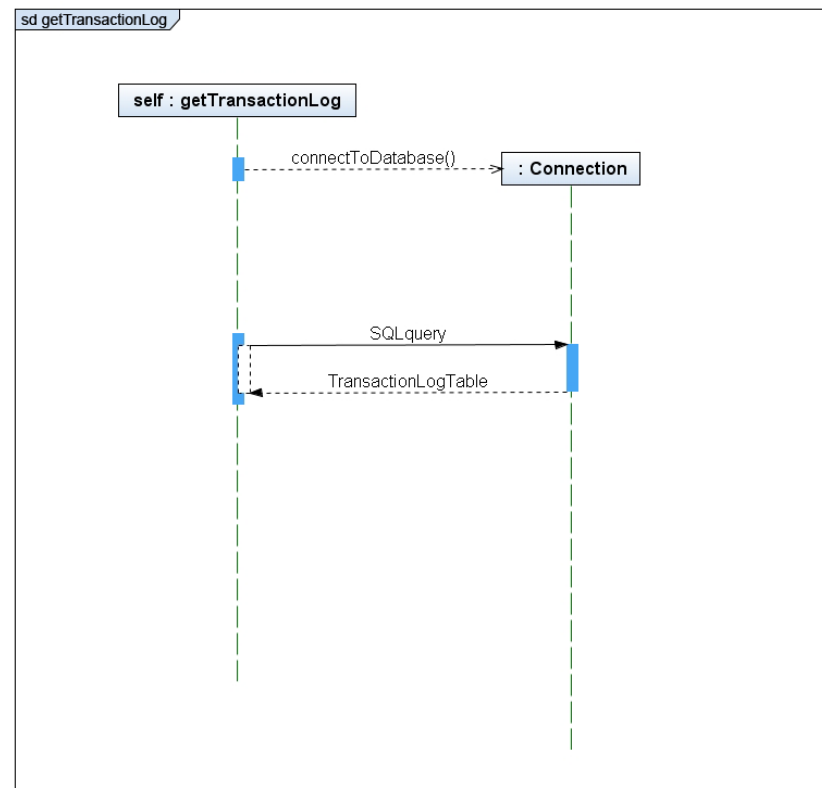
setData



getReport

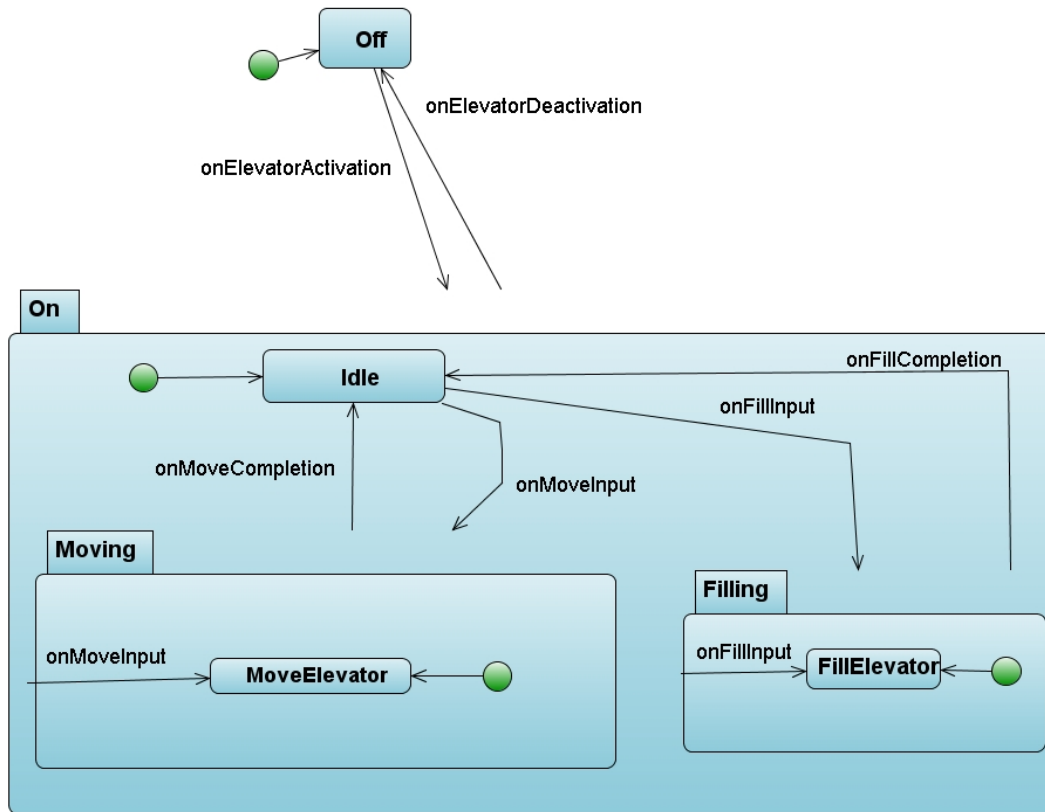


getTransactionLog

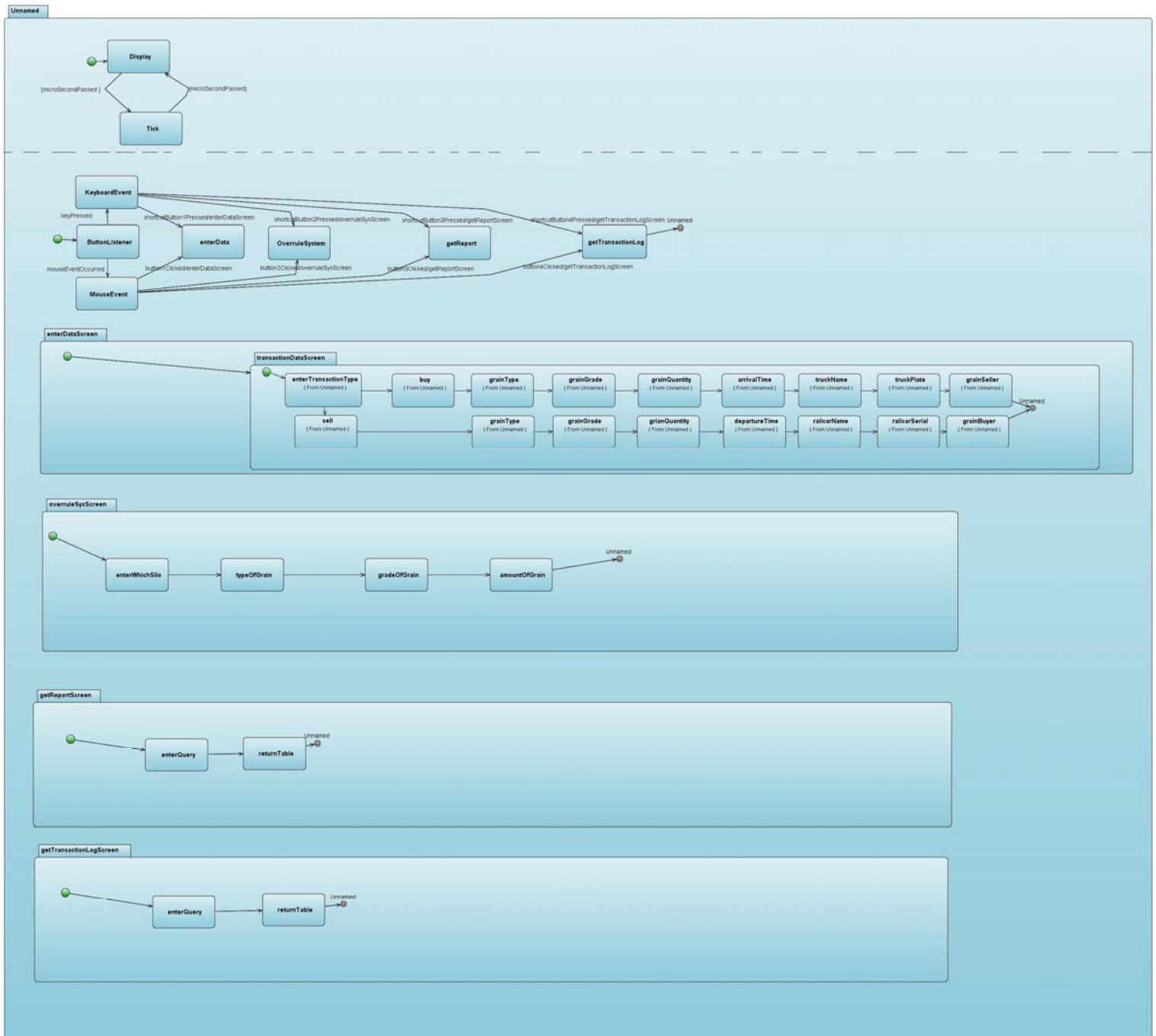


State Diagrams

Elevator Interface State Diagram



User Interface State Diagram



Prototype

Design Preview [NewJFrame]

New Window LogOut About

Distribution Area

Type Grade Quantity

Rice High 1000


Buy Sell

OverRule Accept

Get Data Logs

Get Report

Get Transaction Log



Silo Number	Grain Type	Grain Grade	Remaining Capac...	Total Capacity	Amount Stored
1	Rice	High	5000	8000	3000
2	Rice	Low	2000	8000	6000
3	Hops	High	1000	8000	7000
4	Hops	Low	0	8000	8000
5	Hops	Low	5000	8000	3000
6	Oats	High	7000	8000	1000
7	Oats	Low	8000	12000	4000
8	Wheat	Low	1000	12000	11000
9	Wheat	High	10000	12000	2000
10	-	-	-	12000	-
11	-	-	-	12000	-
12	-	-	-	12000	-