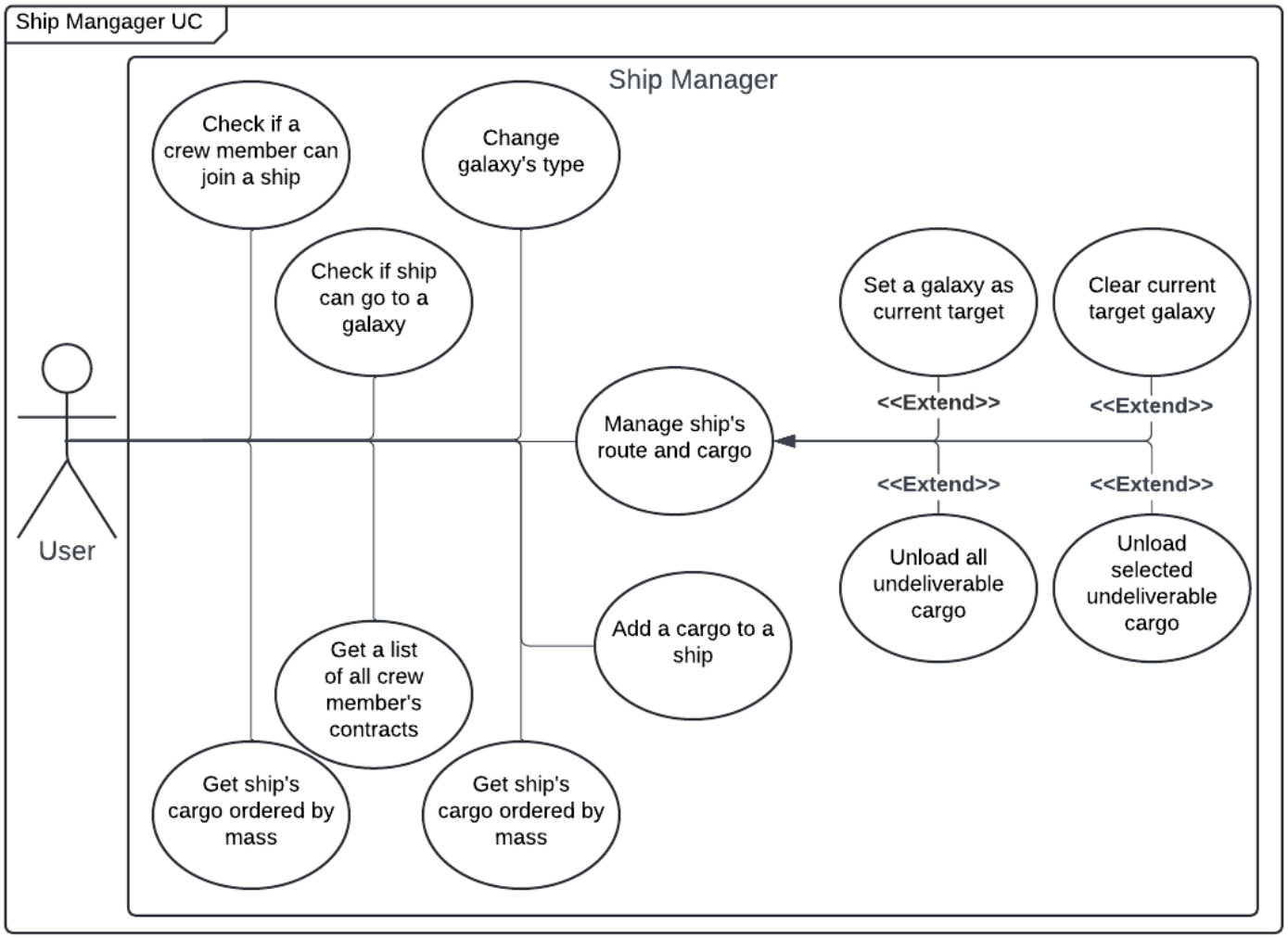
1. User requirements

Space ship manager app

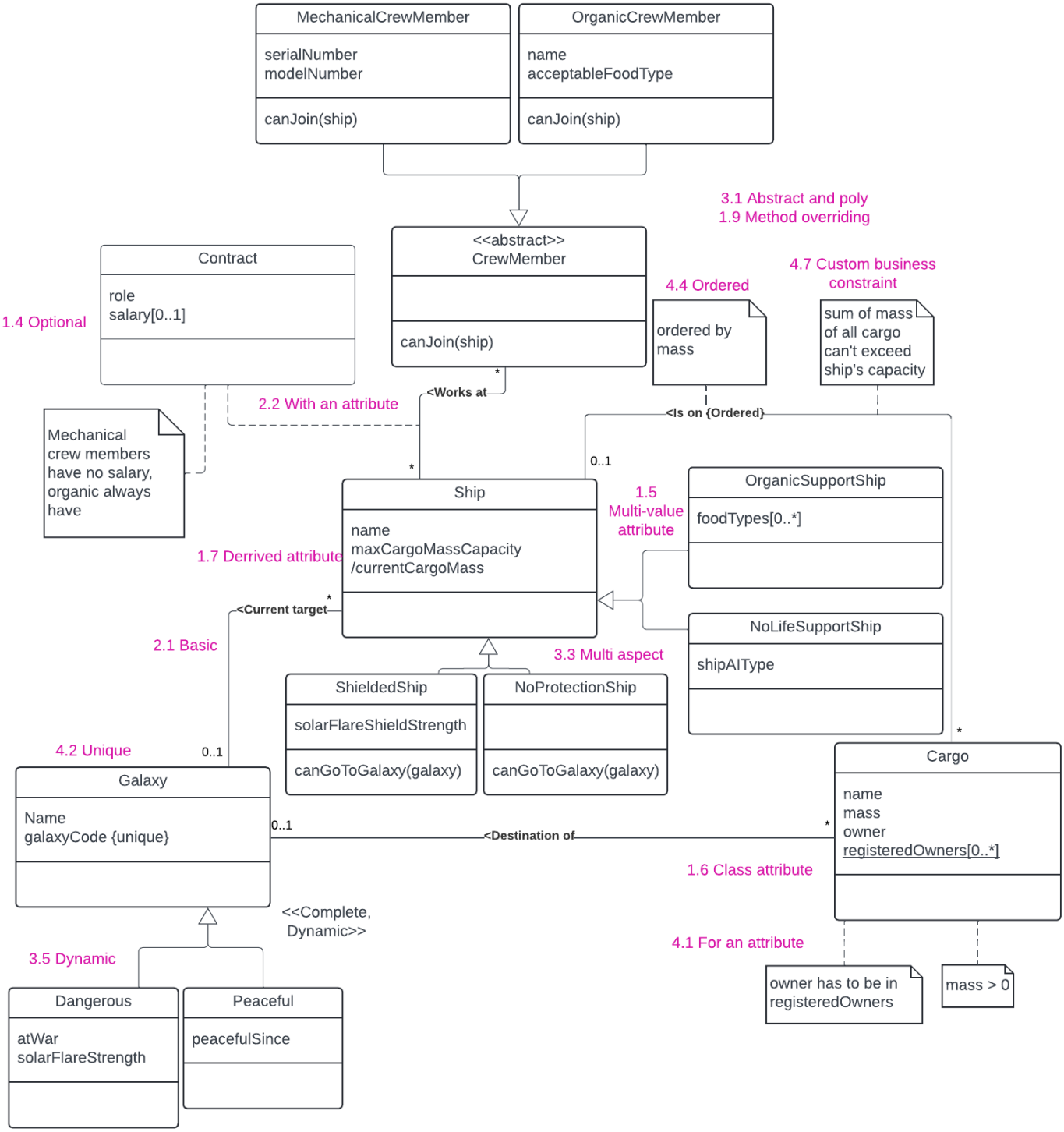
A system which allows managing different parts related to ships.

1. Crew members can be either organic or mechanical.. Organic ones have a name and acceptable food type. They require their food type to be present on a ship. Mechanical crew members have serial and model number.
2. All crew members can have contracts for many ships. Each contract has a role. Organic crew members contracts have a salary. List of crew member's contracts can be retrieved.
3. Each ship has a name and max cargo mass capacity. A ship can support organic life (has its food types) or not (then it has a ship AI type). Ships can be shielded (it has a value of solar flare strength then). Those two traits can be combined in any way.
4. Galaxies have a name and a code which is unique among them. They are split into two categories. Dangerous galaxies can be at war and have a solar flare strength. If a galaxy is at war it cannot be entered by any ship. Otherwise it can be only entered by a ship with solar flare shield strength greater than solar flare strength of that galaxy. There is a way to check if a ship can go to a given galaxy. Peaceful galaxies have a date since they have been peaceful.
5. Cargo have a name, mass, owner. Owner has to be present in the list of registered owners. It can also have a destination galaxy, but it doesn't have to be set initially. Cargo can be loaded on a ship. A ship's cargo list which is ordered by mass can be retrieved. Cargo on a ship cannot exceed its max capacity.
6. A list of all galaxies to be visited based on its cargo can be retrieved. Based on it we can set a new current target for a ship. Cargo which can't be delivered is highlighted and can be unloaded.

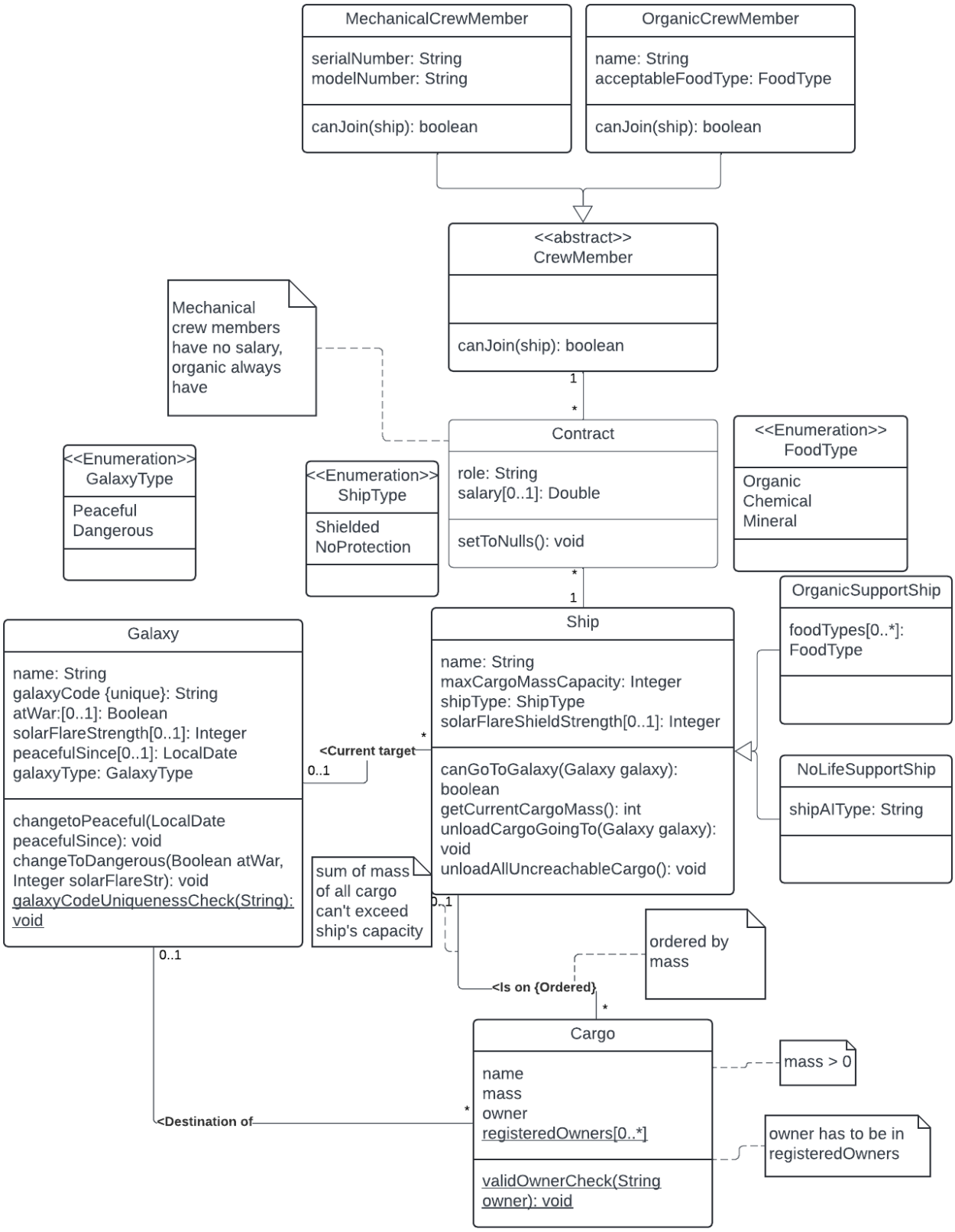
2. The use case diagram



3. The class diagram – analytical



4. The class diagram – design



5. The scenario of selected use case (as text)

[UC] Manage ship's route and cargo

Allows the user to see a list of galaxies given ship needs to travel due to its cargo destinations. It also

allows the user to set one of those galaxies as current target, remove current target or unload cargo which cannot be delivered.

Actors

User.

Purpose and context

User wants see the list of all galaxies which ship has deliveries for. Allows to change current target galaxy and unload undeliverable cargo.

Assumptions and pre-conditions

There is already at least one ship. Existence of at least one galaxy and at least one cargo aboard the ship is needed for lists to not be empty.

Initiating business event

User clicks the "Manage route and cargo" button

Flow of Events

1. User is shown the lists of all galaxies which are destinations of cargo aboard the chosen ship. One list has reachable galaxies and the other has unreachable ones. All of those galaxies have total mass and cargo count shown as well. User is also show the current target.
2. User can choose to remove impossible to deliver cargo or set new target galaxy or remove current target.
3. User clicks " target" next to a chosen galaxy.
4. System changes current target and shows new value
5. User can go to any of alternative flows, step 3 or skip this step
6. User closes the window.

Alternative Flow of Events

3.a. User wants to remove current target

3.a.1 User clicks "Remove Current Target"
3.a.2 System changes the target and shows the new value
3.a.3 Go to 5

3.b User wants to remove all undeliverable cargo

3.b.1 User clicks "Unload all undeliverable cargo."
3.b.2 System removes all undeliverable cargo and updates the list
3.b.3 Go to 5

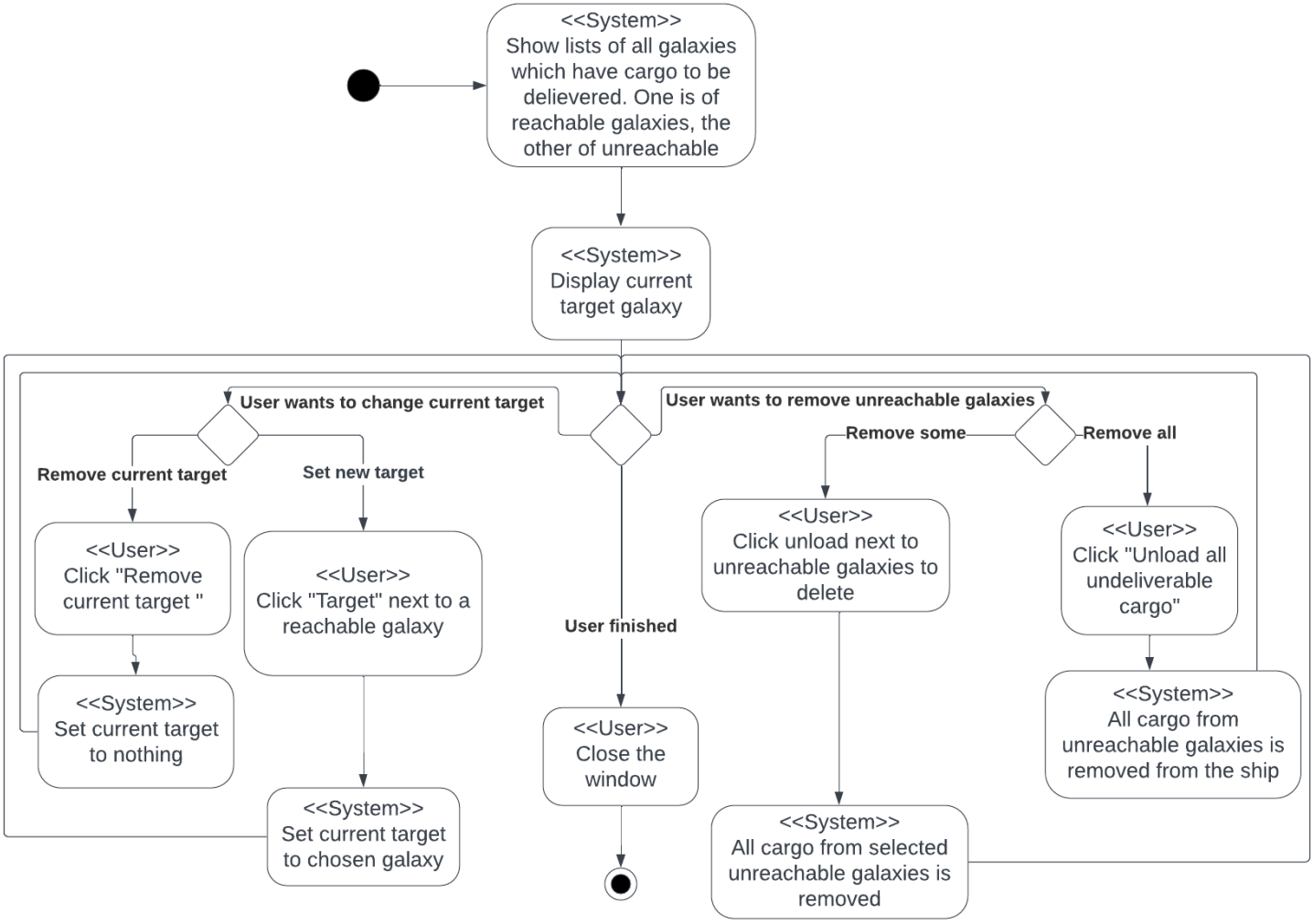
3.c User wants to remove some of undeliverable cargo

3.c.1 User clicks "Unload” next to a galaxy
3.c.2 System removes cargo from that galaxy and updates the list
3.c.3 Go to 5

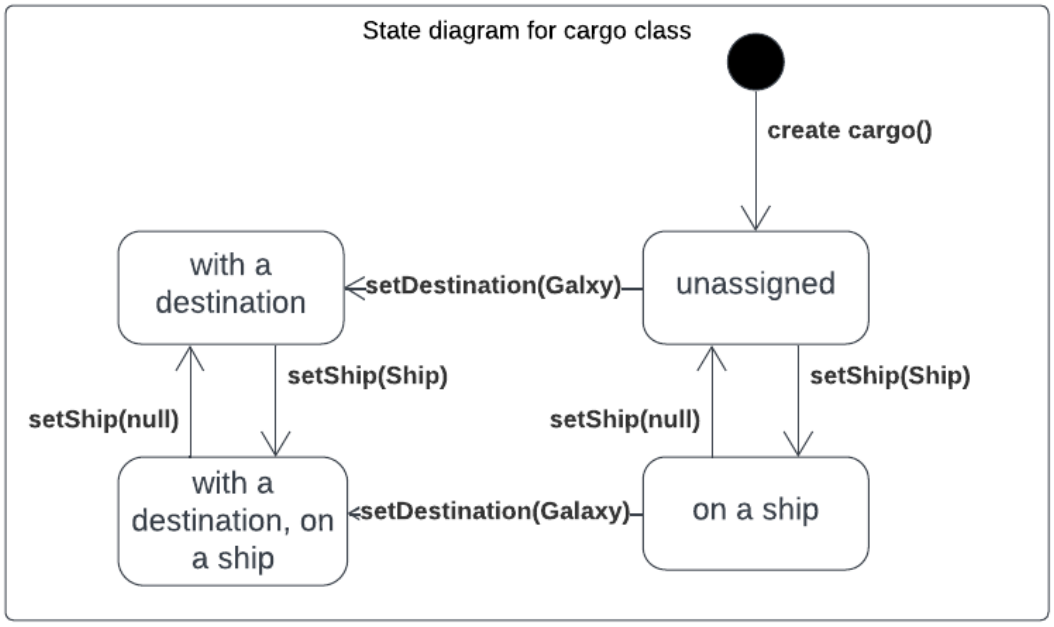
Post-Conditions

Ship has its parameters changed according to user’s input

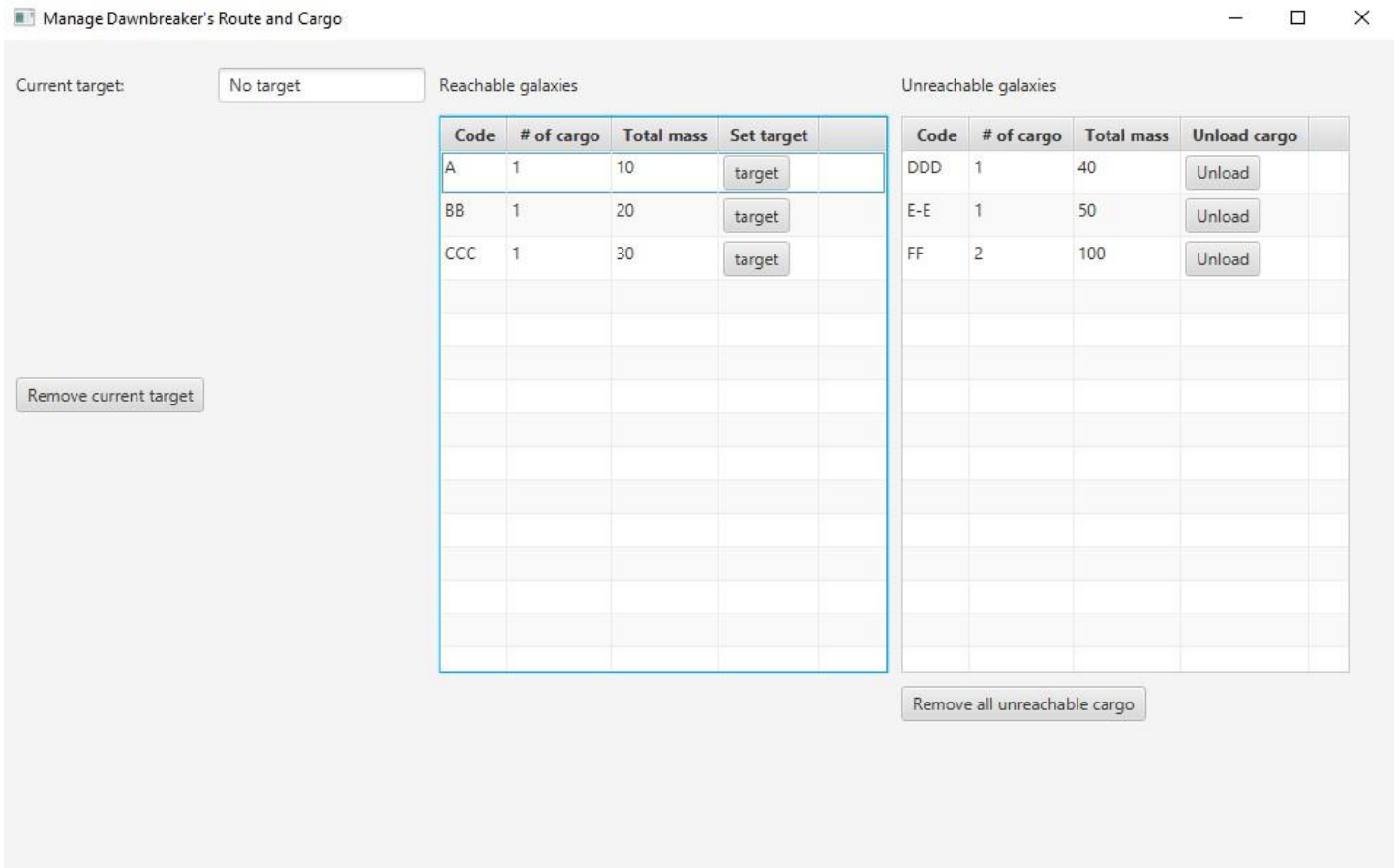
6. The activity diagram for picked use case



7. The state diagram for selected class



8. The GUI design



9. The discussion of design decisions and the effect of dynamic analysis

Persistence has been achieved with Java serialization.

GUI was made with JavaFX

Dynamic inheritance in Galaxy has been flattened. Due to a small number of attributes and to avoid having to change all of the associations in cargo and ship and small number of subclasses flattening was chosen as the optimal solution. It got new methods: changeToPeaceful and changeToDangerous to between classes and galaxyCodeUniquenessCheck.

For multi aspect inheritance of Ship one aspect of Ship has been flattened and the other has been changed into a regular inheritance. Shield-related part has been flattened as it contains less attributes and is not significantly more complicated than the other one. It got new methods: getCurrentCargoMass to get the value of that attribute, unloadCargoGoingTo which removes cargo going to given galaxy and unloadAllUnreachableCargo with self-explanatory name.

Association with an attribute between crew member and ship has been implemented with a contract class. Ship and crew member have one to many associations with contract. Contract got a method setToNulls for removing the association on both sides.

Cargo got a new method validOwnerCheck