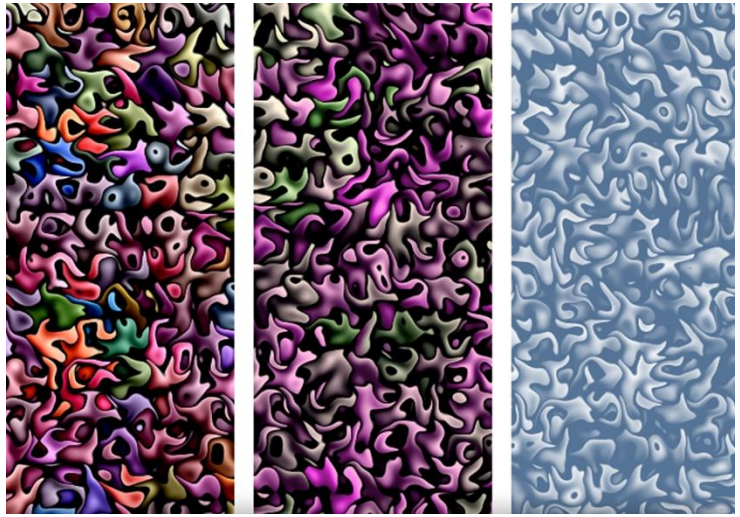
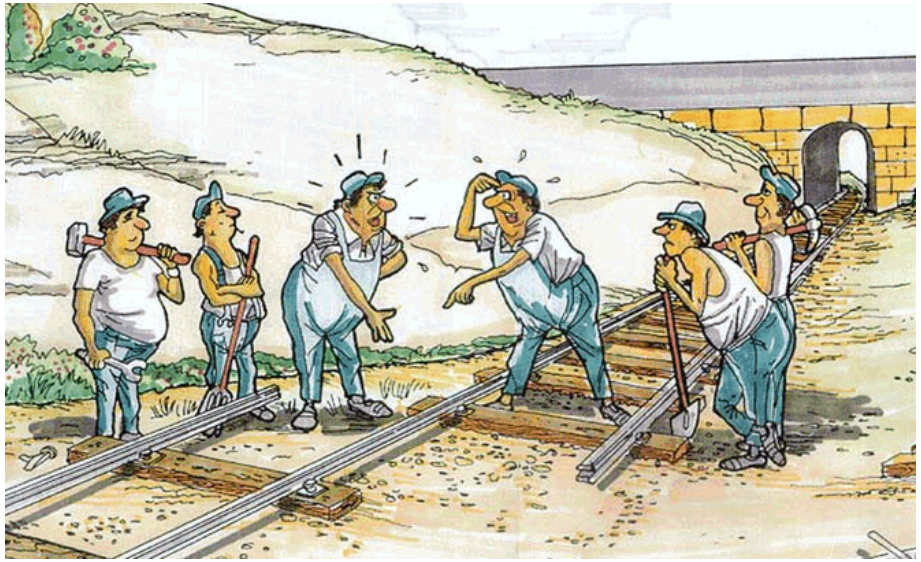
Implementation

Lecture 12

18 June 2026

Slides created by
Prof Amir Tomer
tomera@cs.technion.ac.il



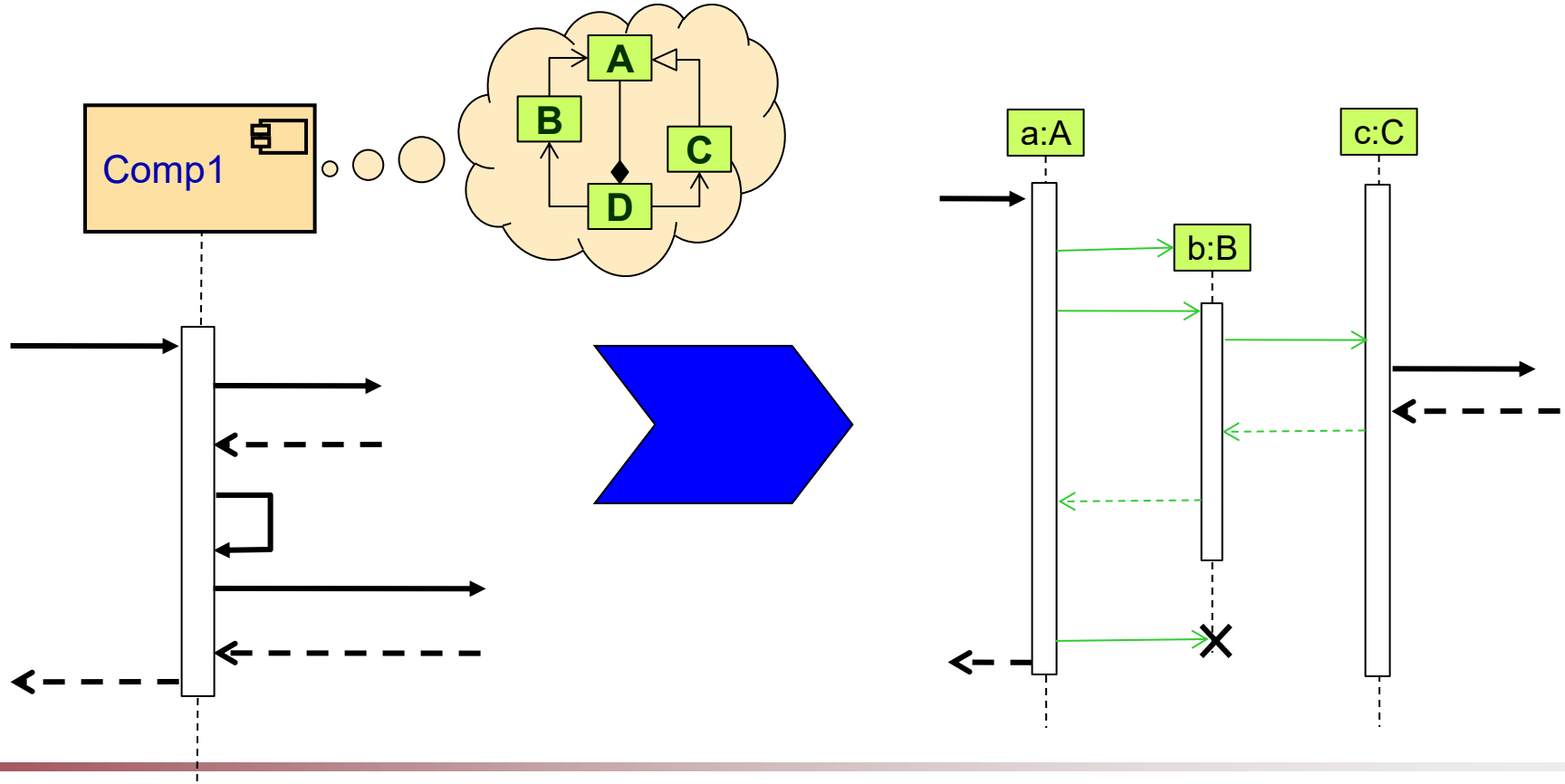


Topics for Today

- Implementing Software Processes

Implementing Software Processes

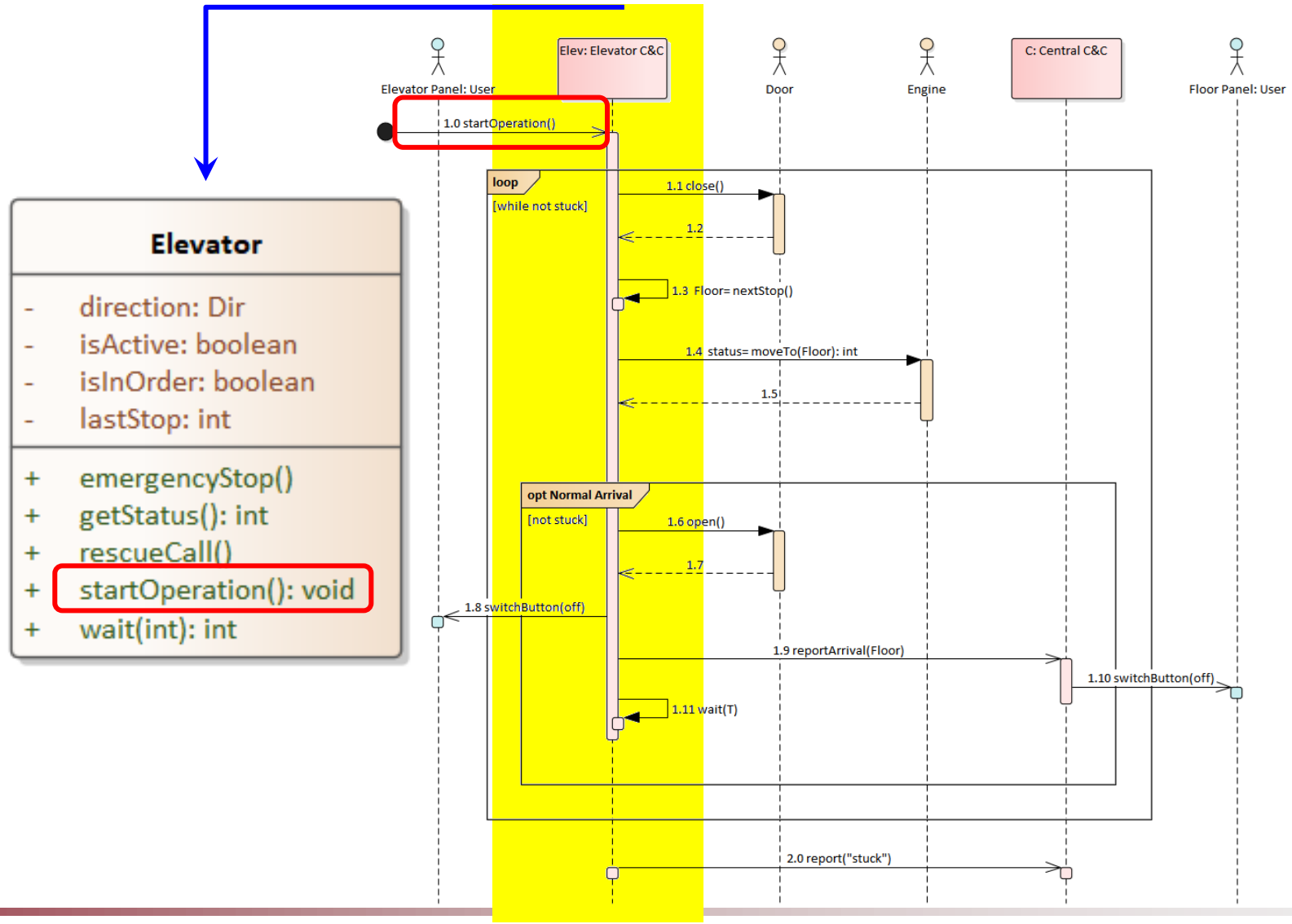
- We implement software processes via objects in code
  - Component functionality → How the component operates in the system's environment
  - Code Objects → Based on class diagrams



Example: Regular Elevator Operations

ref
Travelling between Floors

Reminder



Single Component Class Diagram - Using & Expanding Base Classes

- Build a Class Model for each component
- Each Class Model normally contains the following types of classes:

Sub-set of the base classes

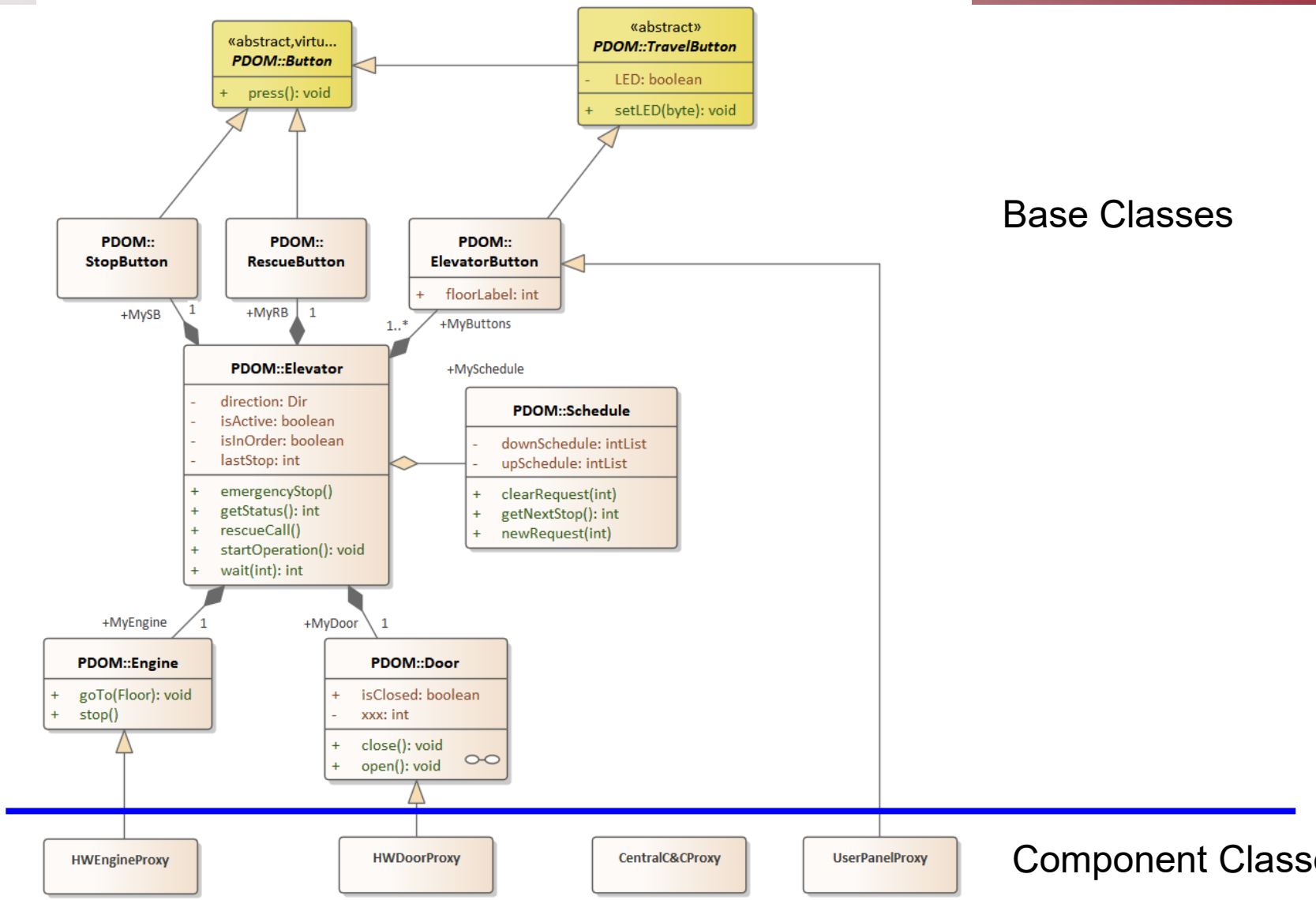
- Classes the component needs to work
- Example: "`ElevatorButton`" class for Elevator C&C

Classes that inherit from the base classes

- Improved or adapted classes the component needs
- Example: `TactileButton` that inherits from `ElevatorButton`

- Example next

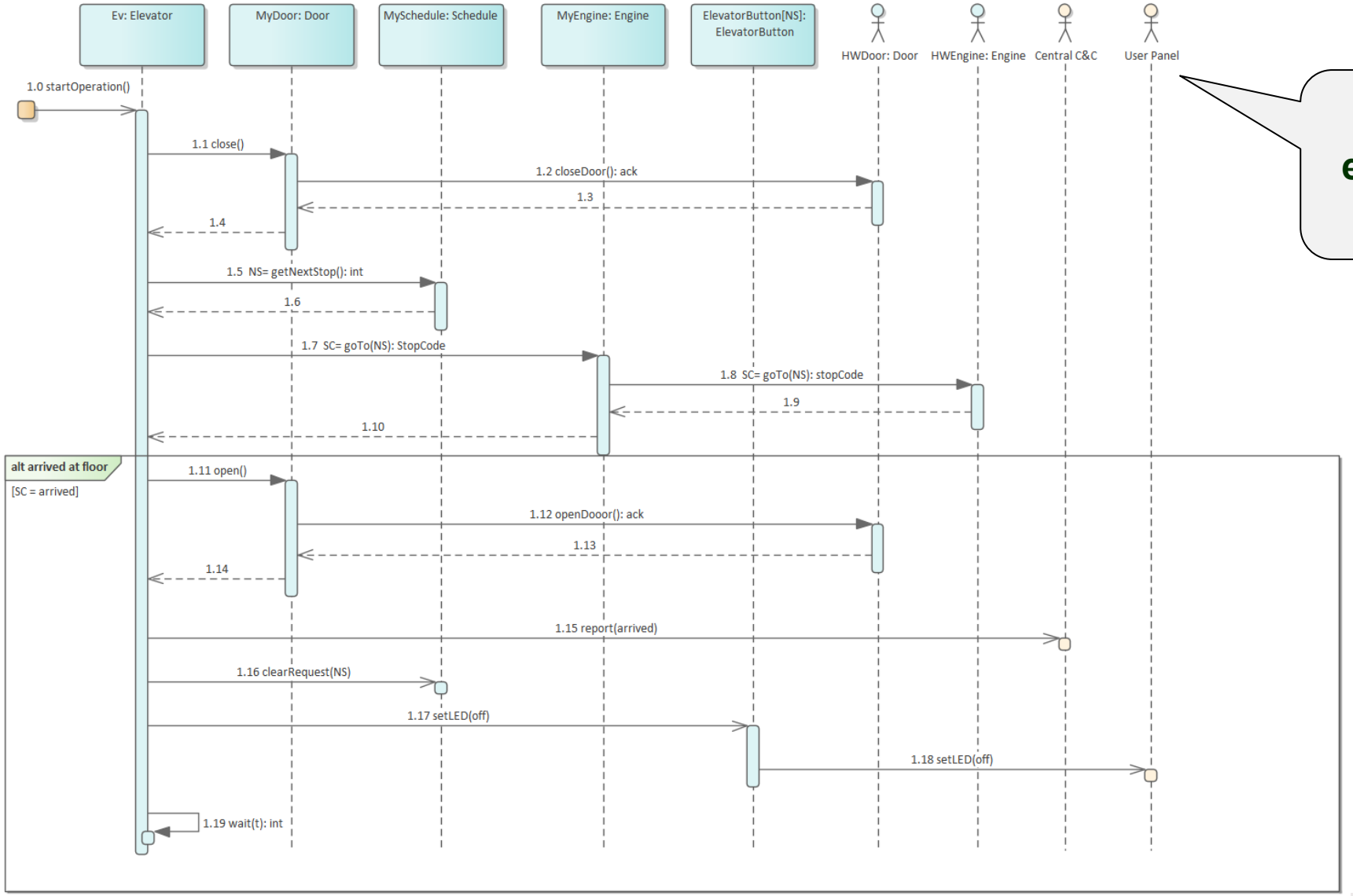
Adding Proxy Classes to Components



Base Classes

Component Classes

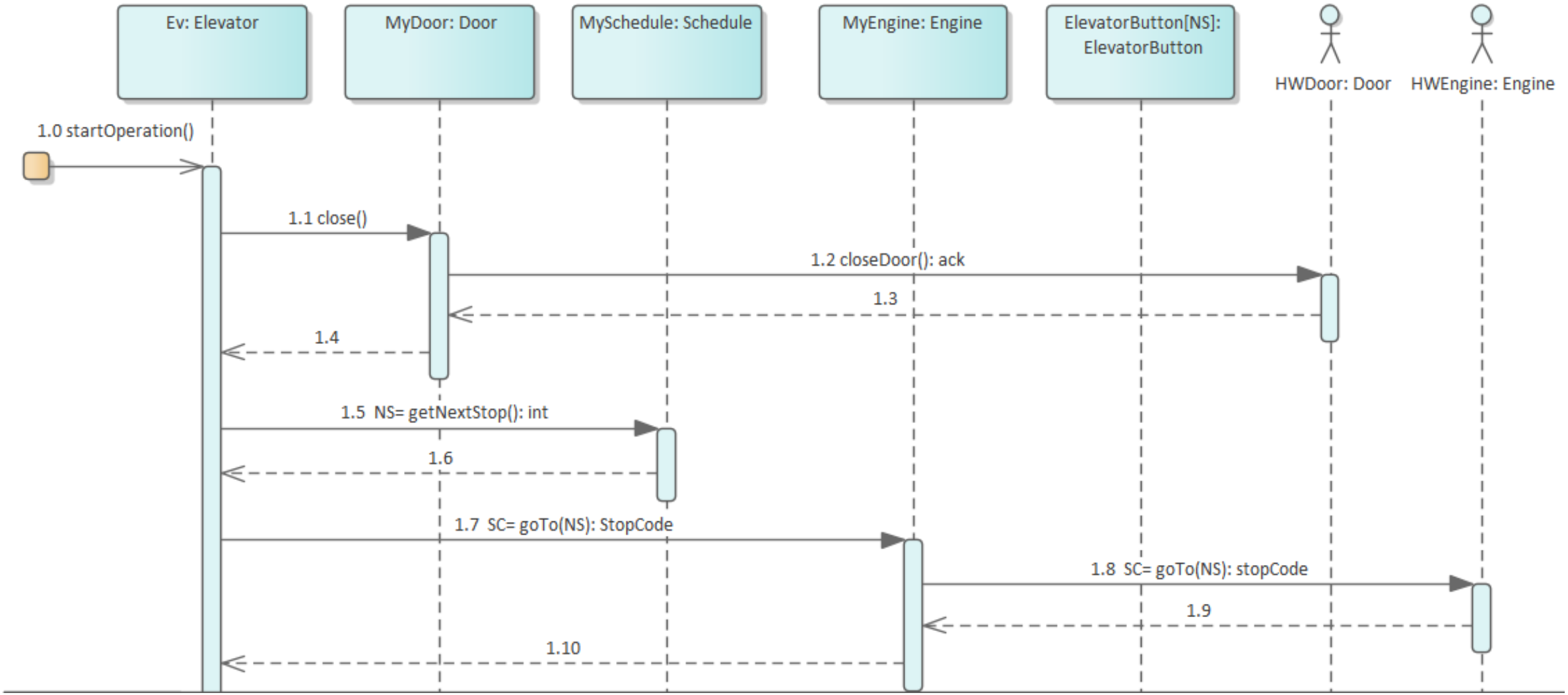
Implementing `startOperation()` method from EV:Elevator object



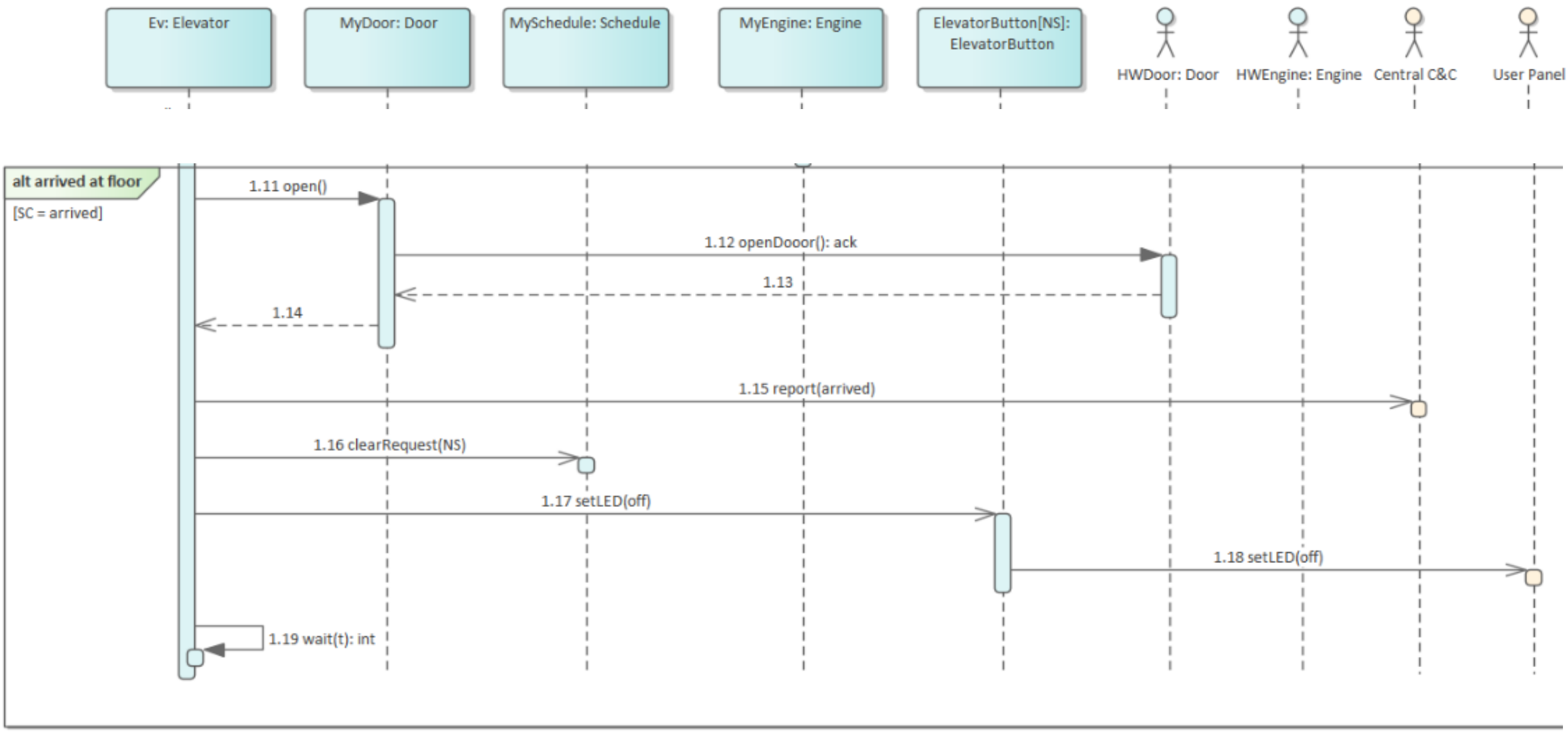
Can replace all external entities with proxies

(Bigger version later)

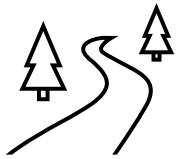
Sequence Diagram – startOperation() (1)



Sequence Diagram – `startOperation()` (2)



Single Component Class Diagram – Dealing with External Environment



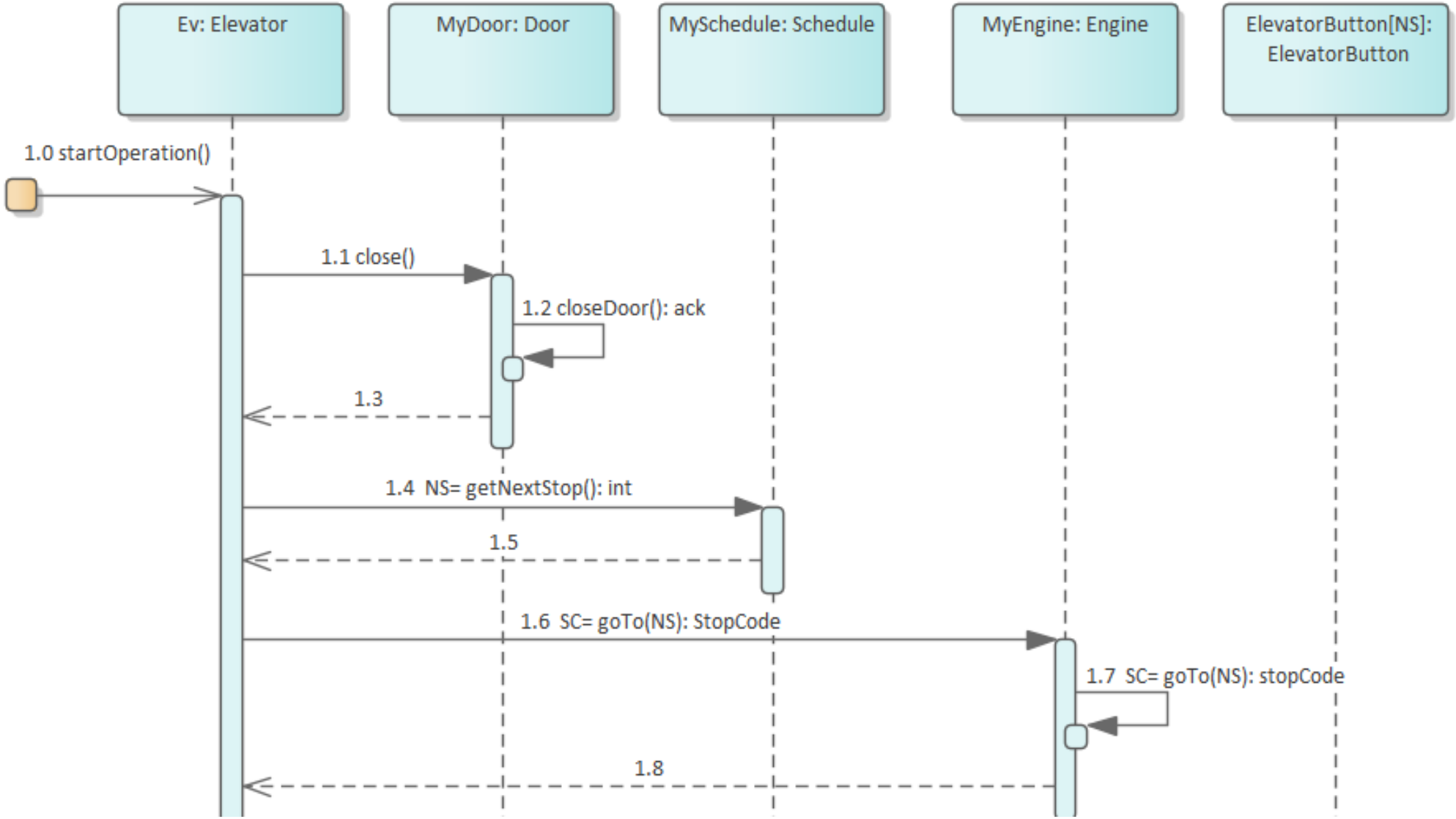
- The component's environment includes
  - Hardware elements with which it communicates
  - Other software elements
- Two ways to deal with the environment

Add "encapsulating" classes that represent external elements	"Short cut" using base class instance methods that direct calls to other components
• Example: `UserPanelProxy` class receives button presses (hardware interrupts) and sends them to objects representing the buttons • Example: `CentralC&CProxy` class represents the Central C&C component for the Elevator C&C component	• Assuming the language supports this

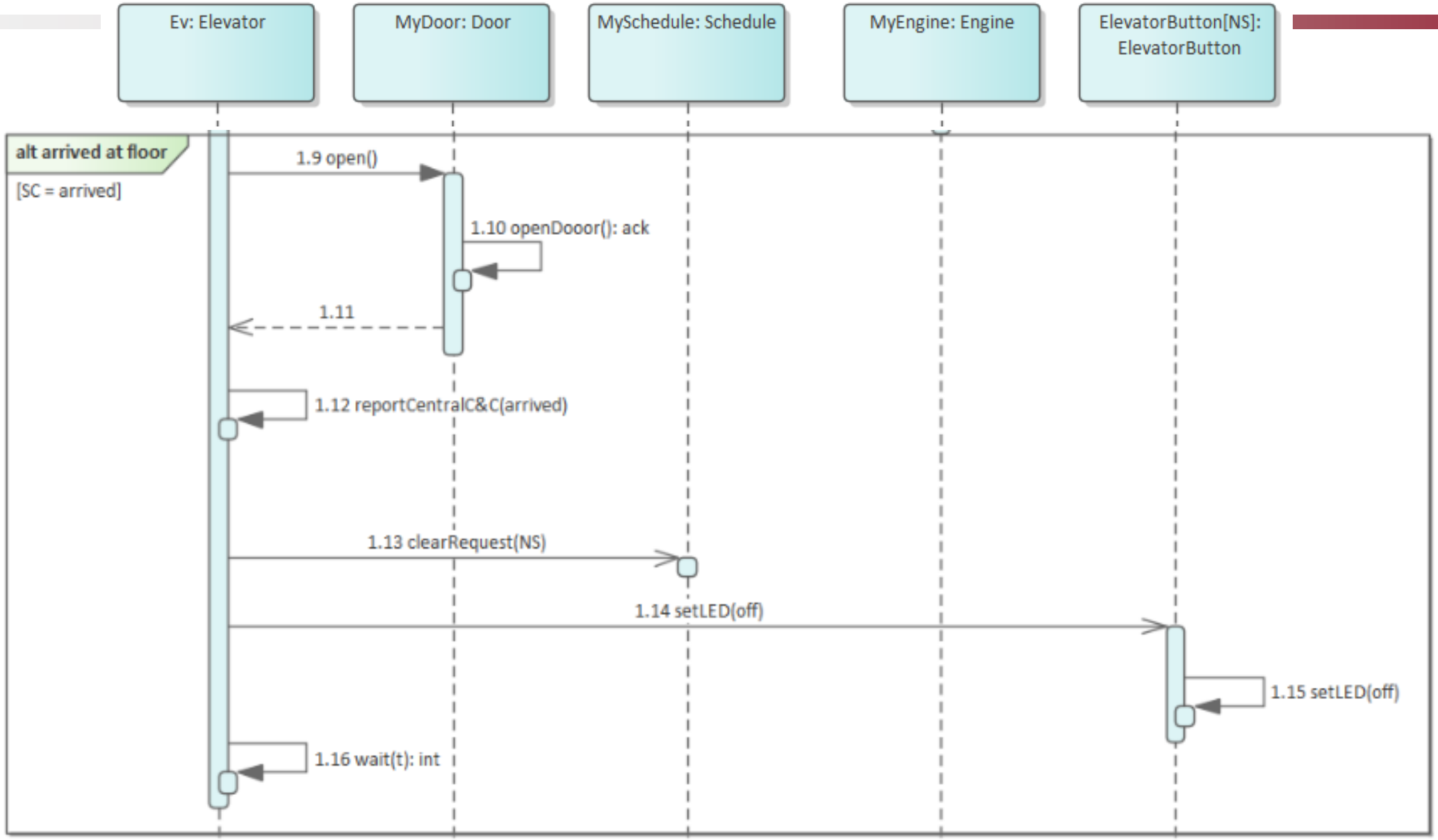
- Example next

18 June 2026

Implementing `startOperation()` method with instance calls (1)



Implementing `startOperation()` method with instance calls (2)



In Class Assignment: Object Level Sequence Diagram

- ePark SUC-1 describes the entry procedure for the park. There is a component-level sequence diagram for the process in the SAD.
- One of the components is Usage Manager
- Create an object-level sequence diagram for the `newRegistration()` method for the component
  - As you build the diagram, add the necessary methods to the objects that are involved in the process
  - Calls to other components will be implemented via instance methods.

Conclusion

- Implementing Software Processes