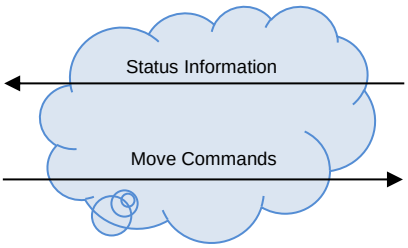
Project Description

The corporation *Wielander Inc.* is an internationally successful manufacturer of load and person elevators. The corporation has been founded in 1907 by George Wielander an engineer for mechanical winching systems. Today Wielander Inc. is a high technology corporation with 900 employees worldwide. The impact of software as future technology has been realized and is reflected in the corporation's product portfolio.

One of new product ideas is the *Elevator Control Center (ECC).* The ECC opens the door for Wielander Inc. to new markets, especially the area of Facility Management. The basis for ECC will be a redesign of the existing control system software. Up to now, all software has been implemented as low-level control software. In future, however, the increasing complexity demands improved reuse concepts and the constantly growing feature set has to be addressed by state of the art software technology.

Elevator Control Center
Software solution to monitor and remote control elevators.



Network
Any network connection suitable for standard Internet protocols.

Elevator
Physical elevators with firmware.

Assignment

Your job is to implement a first version of the remote console, a central part of the new software system. This version will be used to evaluate technical concepts and to define the interface to the low level control system.

Following requirements have to be demonstrated with the prototype:

- Graphical visualization of the elevator. Following information should be displayed: Elevator position, direction, current payload, speed, status of the doors and the set targets of the elevator. For each floor, if there is a call or a stop request.
- It should be possible to operate an elevator either automatically or manually. It should be possible to switch the mode between *automatic* and *manual* at any time.
- In the manual mode the human operator directs the elevator. The operator defines the next target floor. At one point in time only one target floor can be set. The committed direction of the elevator is derived from the target floor.
- An alarm list should show any out of normal states of the elevator (or the system).

- In the automatic mode, the elevator reacts automatically to the calls in the different floors and the stop requests of passengers in the car. The automatic mode is based on an algorithm that directs the elevator to the floors optimally considering calls, requests and the maximum payload.
- The low level system of the elevator will be exposed as RMI interface. Details about this interface are shown in the table below. The accessible values will be updated in cycles of 0.1 seconds.

Implementation and Commissioning

Use the information provided in this document as basis for the project. Further information will be provided by the client upon request (i.e., questions will answered in the course or via the discussion forum). Conduct the project in teams. Apply the techniques and tools introduced in the course in order to develop a high-quality software system. The commissioning will take place in a final presentation and discussion at the end of the course.

Interface Definition (Draft Version)

int	**getCommittedDirection**(int elevatorNumber) Retrieves the committed direction of the specified elevator (up / down / uncommitted)
int	**getElevatorAccel**(int elevatorNumber) Provides the current acceleration of the specified elevator in feet per sec^2
boolean	**getElevatorButton**(int elevatorNumber, int floor) Provides the status of a floor request button on a specified elevator (on/off)
int	**getElevatorDoorStatus**(int elevatorNumber) Provides the current status of the doors of the specified elevator (open/closed)
int	**getElevatorFloor**(int elevatorNumber) Provides the current location of the specified elevator to the nearest floor
int	**getElevatorNum**() Retrieves the number of elevators in the building
int	**getElevatorPosition**(int elevatorNumber) Provides the current location of the specified elevator in feet from the bottom of the building
int	**getElevatorSpeed**(int elevatorNumber) Provides the current speed of the specified elevator in feet per sec
int	**getElevatorWeight**(int elevatorNumber) Retrieves the weight of passengers on the specified elevator
boolean	**getFloorButtonDown**(int floor) Provides the status of the Down button on specified floor (on/off)
boolean	**getFloorButtonUp**(int floor) Provides the status of the Up button on specified floor (on/off)
int	**getFloorHeight**() Retrieves the height of the floors in the building
int	**getFloorNum**() Retrieves the number of floors in the building
int	**getLogging**() Retrieves the current logging level
boolean	**getServicesFloors**(int elevatorNumber, int floor) Retrieves whether or not the specified elevator will service the specified floor (yes/no)
int	**getTarget**(int elevatorNumber) Retrieves the floor target of the specified elevator
void	**setCommittedDirection**(int elevatorNumber, int direction) Sets the committed direction of the specified elevator (up / down / uncommitted)
void	**setServicesFloors**(int elevatorNumber, int floor, boolean service) Sets whether or not the specified elevator will service the specified floor (yes/no)
void	**setTarget**(int elevatorNumber, int target) Sets the floor target of the specified elevator