Design – Sluis

Ian Seelen T34

Rafal Grasman T34

Inleiding

Het design van de sluis is gebaseerd op de requirements van de sluis opdracht. Uit de requirements is te achterhalen dat er vijf basis acties zijn die beschikbaar moeten zijn:

- Schutten
- Alarm
- Invaren vrijgeven
- Uitvaren vrijgeven
- Herstellen

Ook is het uit de requirements duidelijk dat er twee type motoren voor de deuren beschikbaar moeten zijn:

- Een motor die normaal functioneert (eenmalig aan/uit)
- Een motor die om de seconde aangezet moet worden indien blijvende werking gewenst is

Verder zijn er twee type sloten voor deuren:

- Een automatisch slot dat normaal functioneer en geen rekening mee moet worden gehouden
- Een handmatig slot waarmee rekening moet worden gehouden, voor he openen moet deze ontgrendeld worden, als deze gesloten moet worden moet na het sluiten van de deur binnen twee seconden de deur op slot gezet worden.

Uiteindelijk zijn er drie verschillende deuren dat het design moet omvatten:

- Normale deur met standaard motor en slot
- Deur met speciale motor en normaal slot
- Deur met normale motor en speciaal slot

Omdat de communicatie verloopt over een TCP verbinding met ascii tekst als commando's, en de states en variabelen zoveel mogelijk strings vermijden om data te onhouden, zijn *mappers* nodig die van enum/value naar string en terug kunnen vertalen. Dit is ook te zien in bijlage 1 (het klasse diagram).

Er zijn acties, variabele en objecten afhankelijk van timing en daarvoor is de Timer klasse in het dessign ontstaan.

Er zijn 4 hoofdcomponenten die samenwerken: SluiceLogic, Networking, Sluice en Door.

De volgende objecten zijn in de requirements geidentificeerd: Door, Valve, TrafficLight, Sluice.

De volgende afleidingen zijn in de requirements geidentificeerd: DoorOneSecondMotor, DoorTwoSecondLock.

Een TrafficLight is een combinatie van 2 Lights: 1x RedLight en 1x GreenLight.

Een Door bestaat uit 3 Valves (voor low, mid en high waterniveau) en 2 TrafficLights (eentje binnen en eentje buiten de sluis).

Een Sluice bestaat uit twee Doors, en kan informatie over de WaterLevel doorgeven.

Omdat alle klassen netwerking commando's moeten kunnen sturen en acknowledgments ontvangen is er een SluiceNetworkingHandler nodig. D.m.v. deze klassen kunnen alle klassen data uitwisselen met de sluis.

Sluicelogic is de hoofd logica van het programma. Hierin wordt één object Sluice aangemaakt. Verder leest de klasse de user input uit en stuur zo de sluis aan. De status van de sluis wordt geprint naar de console.

Valve

De klasse Valve staat representatief voor een klep in de deur. Elke Deur heeft drie kleppen. Deze kleppen zitten boven elkaar in de deur. De klasse bestuurt de kleppen door deze te openen en te sluiten.

TrafficLightStateMapper, ValveStateMapper, DoorStateMapper, WaterLevelMapper, DoorLockStateMapper:

Deze klassen worden gebruikt om de enums van de despbetreffende klassen te mappen naar een string om te communiceren.

Het protocol van de Sluis geeft geen mogelijkheid om te identificeren welke acknowledgment bij welke gestuurde actie hoort dus alles moet sequentieel om in een consistente state te kunnen blijven. Indien het protocol bij acknowledgments de acties specificeerde (in chronologische volgorde) zou het op een consistente en robuuste manier mogelijk zijn om een multithreaded of asynchronous oplossing te maken.

Er zijn 3 hoofdstaten:

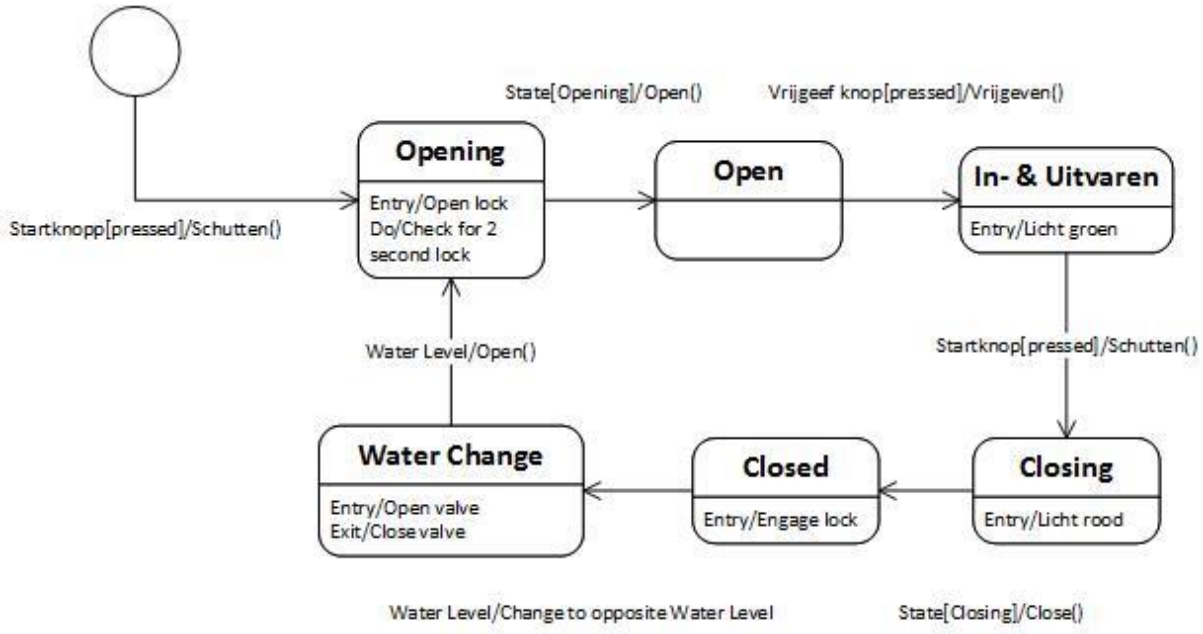
Idle – er gebeurd niks

Schutten – er is een schuttingsoperatie bezig (sluiten van deuren daar waar niveau aan beide kanten hetzelfde is, het waterpeil aan gelijk maken met de tegenovergestelde zijde, de deuren openen waar het water nu gelijk staat, vervolgens naar de idle state gaan)

Alarm – Alles wat er op dit moment gebeurd, stopt. Na herstellen gaat alles weer normaal verder.

De werking van de applicatie is volgend de requirements geimplementeed en het design is een 1:1 vertaling van de requirements. Zie het klassendiagram in bijlage 1.

Bijlage 2 - State Diagrammen



Bijlage 1 – Klassen Diagram