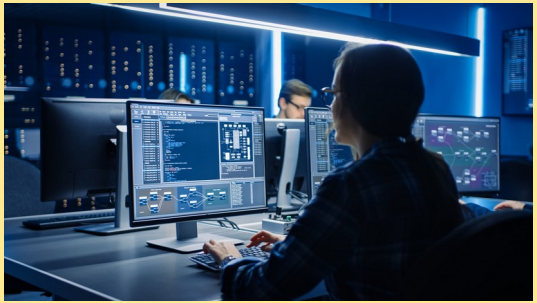
Van simulatie ...

Besturing bij voorkeur van tevoren ontwikkelen en testen op gesimuleerde installatie

- Vervaardiging simulatie kost dagen tot weken, comfortabel achter bureau



- Spaart maanden peperdure tijd en problemen on-site

… naar on-site debugging

Laatste problemen (if any) on-site verhelpen,
De waarden van alle variabelen, incl. inputs en outputs kunnen runtime:

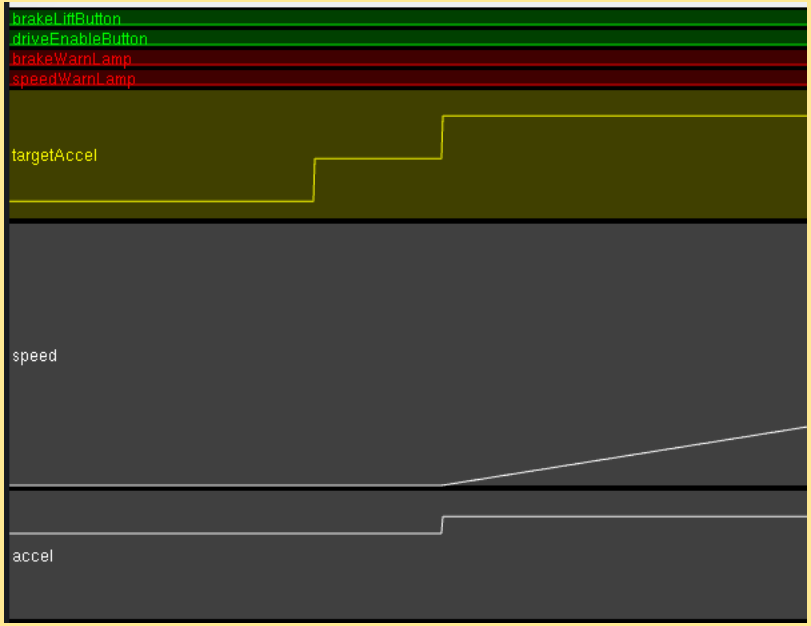
- Worden geïnspecteerd
- Worden gewijzigd (“opgedrukt”)
- Worden weergegeven als functie van de tijd in grafieken

Een effectieve simulatie geeft de essentie van de te besturen installatie weer, gezien vanuit de besturing.

```
class Physics (sp.Module):
    def __init__ (self):
        sp.Module.__init__ (self)

        self.page ('train physics')

        self.group ('control signals', True)
        self.brakeLift = sp.Marker ()
        self.driveEnable = sp.Marker ()

        self.group ('state')
        self.targetAccel = sp.Register ()
        self.speed = sp.Register ()
        self.position = sp.Register ()

        self.group ('limits', True)
        self.maxBrakeDecel = sp.Register (5)
        self.maxDriveAccel= sp.Register (2)
        self.maxDriveDecel = sp.Register (3)
        self.maxSpeed = sp.Register (30)
        self.maxPosition = sp.Register (20_000)

        self.group ('auxiliary')
        self.brakeAccel = sp.Register ()
        self.driveAccel = sp.Register ()
```

Het gaat om simpel en adequaat nabootsen van het real-time gedrag, niet om mooie plaatjes.

```
def sweep (self):
    self.part ('acceleration')
    self.brakeAccel.set (0, self.brakeLift, -self.maxBrakeDecel)
    self.driveAccel.set (self.maxDriveAccel, self.driveEnable, -self.maxDriveDecel)
    self.targetAccel.set (self.brakeAccel + self.driveAccel)

    self.part ('integration')
    self.speed.set (sp.limit (self.speed + self.targetAccel * sp.world.period, 0, self.maxSpeed))
    self.position.set (self.position + self.speed * sp.world.period)
```



Een goede simulatie is zo eenvoudig mogelijk, maar niet eenvoudiger.

Bij voorkeur wordt de simulatie bestuurd door de “echte” besturing,
of door code waaruit de echte besturing automatisch wordt gegenereerd.

De simulatie gedraagt zich dan als “digital twin” van het echte systeem en heeft exact hetzelfde control interface.

Control interface van digital twin

```
class Deckhand:
    def __init__ (self, crew):
        self.crew = crew

    def work (self, sweep):
        with sc.socket (*sw.socketType) as self.clientSocket:
            self.clientSocket.connect (sw.address)
            self.socketWrapper = sw.SocketWrapper (self.clientSocket)

            while True:
                self.input ()
                sweep ()
                self.output ()
                tm.sleep (0.02)

    def input (self):
        sensors = self.socketWrapper.recv ()

        self.courseAngle = sensors ['courseAngle']
        self.vaneAngle = sensors ['vaneAngle']
        self.lattitude = sensors ['lattitude']
        self.longitude = sensors ['longitude']

    def output (self):
        actuators = {
            'rudderAngle': self.rudderAngle,
            'sheetLength': self.sheetLength
        }

        self.socketWrapper.send (actuators)

    def getCurrentLocation (self):
        return self.lattitude, self.longitude
```

Control interface template voor echte systeem

```
class Deckhand:
    def __init__ (self, crew):
        self.crew = crew

    def work (self, sweep):
        # Make connection
        # In a loop, call input, sweep, output and sleep in that order

    def input (self):
        # Receive sensor data from ship

    def output (self):
        # Send actuator data to ship

    def getCurrentLocation (self):
        # Read out GPS data
        # ...
        # Convert to signed lattitude, longitude (see comment in almanac)
        # ...
        return lattitude, longitude
```

Een goede simulatie is in-code gedocumenteerd en blijft bewaard om ook latere wijzigingen eerst off-line te kunnen testen.

```
void correctStackImage (TLocator completeLocator, bool put, bool predicted = false);
/*
Actions:
    If completeLocator is flying
        Fills empty locations below locator with unknown containers from stackimage
        Report error

    If completeLocator is burried
        Remove containers above locator from stackimage
        Report error
*/
```

Voor besturing (Control) en simulatie (Physics) van het bestuurde systeem kan dezelfde taal en dezelfde computer worden gebruikt.

```
class Control (sp.Module):
    def __init__ (self):
        sp.Module.__init__ (self)

        self.page ('train control')

        self.group ('control buttons', True)
        self.brakeLiftButton = sp.Marker ()
        self.driveEnableButton = sp.Marker ()

        self.group ('warning lamps')
        self.brakeWarnLamp = sp.Marker ()
        self.speedWarnLamp = sp.Marker ()
```

```
class Physics (sp.Module):
    def __init__ (self):
        sp.Module.__init__ (self)

        self.page ('train physics')

        self.group ('control signals', True)
        self.brakeLift = sp.Marker ()
        self.driveEnable = sp.Marker ()

        self.group ('state')
        self.targetAccel = sp.Register ()
        self.speed = sp.Register ()
        self.position = sp.Register ()

        self.group ('limits', True)
        self.maxBrakeDecel = sp.Register (5)
        self.maxDriveAccel= sp.Register (2)
        self.maxDriveDecel = sp.Register (3)
        self.maxSpeed = sp.Register (30)
        self.maxPosition = sp.Register (20_000)
```