9

Assignment “The Frog” (NL)

Project Team Building Challenge / Sasa

Colophon

Date 19 April 2023
Reference PTBC SE(NL)
Version 1.2
Department HBO-ICT
Author Remco van Maanen

Table of Contents

Inhoud

1 The Assignment

1.1 Inleiding

In dit Software Engineering project moet je jouw samenwerk- en Java-vaardigheden inzetten en versterken.

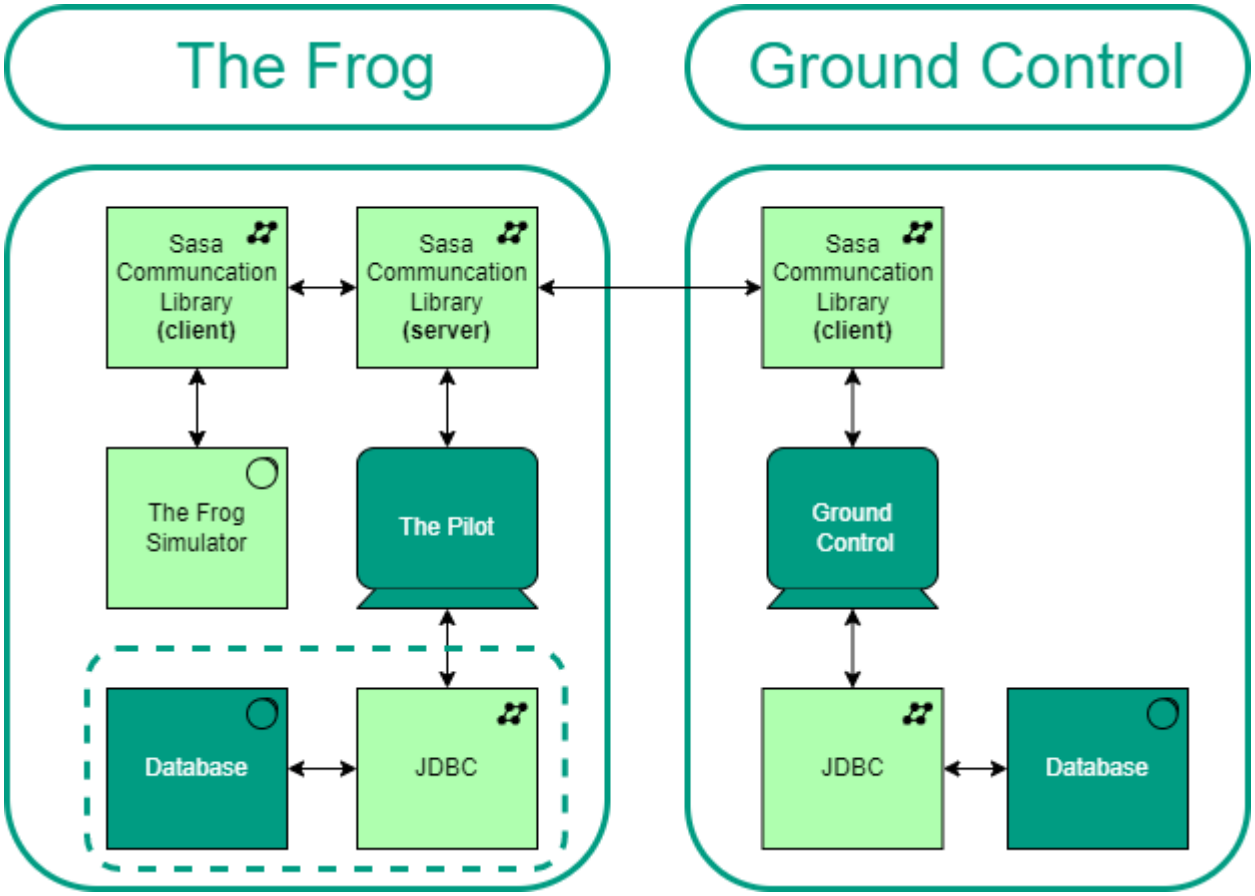
De Saxion Space Academie, SaSA wil binnenkort een missie naar de maan gaan uitvoeren en wil hiervoor een voertuig op de maan zetten. Dit voertuig, met de officiële naam "The Rather Rudimentary Operated Lunar Vehicle", is intussen in ontwikkeling bij de Werktuigbouw studenten samen met studenten van Technische Informatica.

Omdat dit wel een erg lange naam is, heeft dit voertuig de bijnaam The Frog gekregen. The Frog is een voertuig gemaakt om het maanoppervlak te bestuderen, maar deze heeft nog geen software om autonoom beslissingen te nemen. SaSA heeft daarom aan HBO-ICT gevraagd om de intelligente software te ontwikkelen. Deze software, The Pilot software genoemd, moet in staat zijn om het voertuig daadwerkelijk autonoom te laten rijden o.b.v. radardata en commando's vanaf het grondstation op de aarde en ook tijdens het rijden diverse data op te slaan en weer terug naar het grondstation te sturen.

Ook dat grondstation is nog niet af, want ook daar moet de nodige software, genaamd Ground Control, nog voor gemaakt worden waarmee The Frog op afstand kan worden aangestuurd en waarmee gevolgd kan worden waar deze rijdt, welk parcours is afgelegd en wat deze "ziet" (met de radar).

Omdat The Frog nog niet af is, is er alvast een simulatiemodel gemaakt die gebruikt kan worden om The Pilot software te ontwikkelen, die later in het echte voertuig gaat worden gebruikt. De software voor de communicatie tussen The Frog en The Pilot evenals die tussen The Pilot en Ground Control is al wel ontwikkeld door studenten van Technische Informatica en wordt met het simulatiemodel meegeleverd.

1.2 Architectuur



1.3 Sasa communication library

Om **The Pilot** te kunnen verbinden met **The Frog** en met **Ground Control** is er een communicatiebibliotheek gemaakt; Sasa-communication.jar.

Hierin wordt de volgende Interface aangeboden:

```
package nl.saxion.ptbc.sockets;

/**
 * Interface class, implement this interface and after subscription of your class you
 * will receive updates.
 */
public interface SasaSocketInformation {
    public void receivedData(String data);
    public void statusUpdate(String sender, String status);
}
```

De *receivedData* methode wordt automatisch aangeroepen zodra er een bericht is ontvangen. Dit bericht staat in de *data* parameter.

De *statusUpdate*-methode wordt automatisch aangeroepen bij een connect en een disconnect van een client bij de server.

Sasa Communication Server (The Pilot)

Om van **The Pilot** een communicatie-server te maken moet je een klasse aanmaken die de *SasaSocketInformation* interface implementeert. Vervolgens moet in deze klasse de communicatie-server worden aangemaakt.

```
public class SasaServerApplication implements SasaSocketInformation {
    private SasaServer sasaServer;

    /**
     * Method launches the server at the specific port.
     * @param port port to listen on
     */
    public SasaServerApplication(int port) {
        // Setup server
        sasaServer = new SasaServer();
        sasaServer.subscribe(this);
        sasaServer.listen(port);
        System.out.println("Server listening on port " + port);
```

Hiermee gaat de server luisteren naar nieuwe clients die verbinding maken op poort *port*.

Vervolgens moet de daadwerkelijke implementatie van de *SasaSocketInformation* interface worden gedaan, zodat er de berichten kunnen worden opgevangen evenals statusinfo over (dis)connecting clients.

Dus om data te lezen:

```
/**
 * Callback method, that is called when information is send to this server.
 * @param data data that is send from client.
 */
@Override
public void receivedData(String data) {
    System.out.printf("[Received] %s%n", data);
}
```

en om status-informatie te ontvangen:

```
/**
 * Callback method, called in case of a status update
 * @param sender sender of the status update in text
 * @param status status message itself
 */
@Override
public void statusUpdate(String sender, String status) {
    System.out.printf("[State] %s: %s%n", sender, status);
}
```

LET OP dat de berichten die moeten worden verzonden en ontvangen altijd een enkele string zijn, dus als er meer gegevens in één bericht moeten worden verzonden, plak deze dan samen met een uniek scheidingsteken (bijvoorbeeld zoals er gebeurt in een CSV-bestand).

Communicatie kun je altijd het beste van dezelfde kant starten, dus bijvoorbeeld **The Pilot** (de server) stuurt een commando (bericht) naar **The Frog** (de client) om uit te voeren. **The Frog** voert dit commando uit en stuurt het resultaat vervolgens terug naar **The Pilot**. Na ontvangst van dat antwoord herhaalt de cyclus zich; **The Pilot** kan een nieuw commando naar **The Frog** sturen.

LET OP dat als er meer berichten worden geretourneerd in plaats van een enkel bericht, de laatste berichten moeten aangeven dat dit het laatste bericht van die reeks is, zodat de ontvangende kant weet wanneer alle berichten zijn ontvangen (en een nieuw bericht kan worden verzonden).

Sasa Communication Client (Ground Control)

Om als **Ground Control** een verbinding te maken met **The Pilot** moet het volgende gebeuren:

```
// Connect to server
sasaClient = new SasaClient();
sasaClient.subscribe(this);
sasaClient.connect(host, port);
```

Hiermee wordt een verbinding gemaakt met een de communicatie-server van **The Pilot** met IP-adres *host* op poort *port*.

Vervolgens kunnen berichten worden verstuurd,

```
// Send a message to the server
sasaClient.send(messageToSend);
```

of ontvangen (zie het voorbeeld in de vorige paragraaf, via de override van de *receivedData*.

1.4 The Frog aansturing

The Frog heeft een ingebouwde **Sasa** communication-client en kent een aantal commando's, waarmee deze kan worden aangestuurd. Alle berichten van **The Frog** worden altijd voorafgegaan door "FROG " en alle argumenten zijn gescheiden door een spatie. Gebroken getallen worden gescheiden door een komma.

Commando's die naar **The Frog** worden gestuurd moeten altijd voorafgegaan worden door "PILOT ".

Testen van The Frog

Je kan de **Sasa**-communication library vanaf de command line opstarten:

```
>java -jar Sasa-communication.jar
Server listening on port 50000
You can type a message at any time and press enter to send it to the client.
Enter /stop to stop this server
```

De output in de volgende paragrafen is steeds de output die je ziet wanneer je gebruik maakt van de ingebouwde server; alles tussen [en] (zoals [State], [Send] en [Received]) wordt dus niet daadwerkelijk verstuurd resp. ontvangen maar is alleen de output van de ingebouwde server zelf.

Starten van The Frog

Wanneer nu **The Frog** wordt gestart (Windows: TheFrog.exe, Mac: TheFrog of Linux: TheFrogLinux.x86_64), maakt deze automatisch een verbinding met de server op localhost:50000 (dus IP adres 127.0.0.1 en poort 50000):

```
[State] client: Connected
```

Je kunt nu testen met het sturen van commando's naar **The Frog**. Type bijv. "HELP" (zonder quotes) en kijk maar wat er gebeurt.

Wanneer **The Frog** wordt gestart vanaf de command-line kun je ook argumenten meegeven:

- -host: aaa.bbb.ccc.ddd connect met een server op IP-adres aaa.bbb.ccc.ddd (default: localhost)
- -port: xxxxx connect met een server op poort xxxxx (default: 50000)

Na het starten maakt **The Frog** zich bekend bij de server en geeft hij aan waar hij staat:

```
[Received] FROG CONNECTED 2023-1 5000 -45 45 20
[Received] FROG POSITION 257,9739 248,198 17,68 62,43998
```

Zie de Help commando paragraaf voor de betekenis van de verschillende velden.

Help commando

Het commando (bericht) "HELP" geeft in een aantal berichten alle bekende commando's en de te versturen berichten van **The Frog** weer:

```
[Sent] HELP
[Received] FROG HELP COMMANDS:
[Received] FROG HELP HELP this command :)
[Received] FROG HELP PILOT DRIVE motor-power angle duration
[Received] FROG HELP PILOT JUMP x-coordinate z-coordinate angle
[Received] FROG HELP PILOT RADAR ON|OFF|-
[Received] FROG HELP PILOT EXIT
[Received] FROG HELP MESSAGES:
[Received] FROG HELP FROG CONNECTED version max-motor-torque min-steering-angle max-steering-angle max-duration
[Received] FROG HELP FROG DRIVING x-coordinate z-coordinate y-coordinate angle
[Received] FROG HELP FROG POINT x-coordinate z-coordinate y-coordinate
[Received] FROG HELP FROG POSITION x-coordinate z-coordinate y-coordinate angle
[Received] FROG HELP FROG RADAR ON
[Received] FROG HELP FROG RADAR OFF
[Received] FROG HELP FROG RADAR START x-coordinate z-coordinate y-coordinate angle
[Received] FROG HELP FROG RADAR STOP
[Received] FROG HELP FROG STATUS found-Detonator found-TNT explosives-delivered
[Received] FROG HELP FROG HELP these messages :P
```

Drive commando

Dit "PILOT DRIVE" commando kent de volgende parameters:

- motorstand, dus de hoeveelheid power (0 is remmen/ stoppen) en
- draaiing van de wielen in graden (0 is rechtdoor) en
- de tijd hoelang de motor actief moet zijn (0 is onbeperkt)

Na het uitvoeren van ieder "PILOT DRIVE" commando stuurt **The Frog** een bericht terug van de nieuwe positie waar deze tot stilstand is gekomen (eenzelfde "POSITION" bericht als bij het opstarten).

Vervolgens wordt er automatisch een radarbundel verzonden en volgt er een groot aantal berichten met radarpunten (vergelijkbaar met de parkeersensoren in een auto). Zie verder het Radar commando.

In geval er 0 als duur is aangegeven wordt er niet gestopt en blijft **The Frog** gewoon rijden met de gegeven motorstand en draaiing van de wielen. Iedere seconde wordt er dan een "DRIVING" bericht gestuurd met dezelfde parameters als een "POSITION" bericht:

```
[Received] FROG DRIVING 238,0576 275,6794 17,36318 180,5182
[Received] FROG DRIVING 238,0387 273,6254 17,36317 180,519
[Received] FROG DRIVING 238,0102 270,5063 17,36321 180,5253
[Received] FROG DRIVING 237,9742 266,5086 17,3632 180,5109
```

Tijdens het rijden mogen er nieuwe commando's gegeven worden, bijvoorbeeld om een nieuw radarbeeld op te vragen ("PILOT RADAR"). Als dit nieuwe commando echter ook een DRIVE commando is, komt deze in de plaats van het huidige DRIVE commando, dat dus niet wordt afgemaakt.

Om een rijdende **The Frog** te stoppen kan het commando "PILOT DRIVE 0 0 0" gegeven worden.

Jump commando

Voor testdoeleinden is het mogelijk om **The Frog** op een bepaalde positie te plaatsen in de opgegeven richting met "PILOT JUMP"

Hiervoor zijn de volgende gegevens nodig:

- x-coördinaat,
- z-coördinaat en
- rotatiehoek (in graden).

Radar commando

Standaard wordt er een radarbundel verstuurd bij de start van **The Frog** en na ieder "PILOT DRIVE" en "PILOT JUMP" commando. Dit gedrag kan worden aangepast met het "PILOT RADAR" commando:

- PILOT RADAR ON – stuur een radarbundel na ieder DRIVE commando
- PILOT RADAR OFF – stuur geen automatische radarbundel meer
- PILOT RADAR – stuur nu een radarbundel (ongeacht of automatische radarbundel aan staat)

Tijdens het rijden kan er dus een "PILOT RADAR" commando worden gestuurd om een beeld te krijgen op dat moment.

Iedere detectie geeft een aparte melding, dus per radarbeeld worden er en kleine 1000 berichten van detectie gestuurd. Iedere radarbundel wordt voorafgegaan door een "RADAR START" bericht en wanneer de laatste detectie van de bundel binnen is afgesloten met een "RADAR STOP" melding:

```
[Received] FROG RADAR START 261,9367 250,2662 18,28 62,87998
[Received] FROG POINT 9,04 17,86 0,32
[Received] FROG POINT 14,51 15,23 -0,26
…
[Received] FROG POINT 74,85 63,01 -1,6
[Received] FROG POINT 74,17 63,64 -1,62
[Received] FROG RADAR STOP
```

LET OP de coördinaten van de gedetecteerde punten zijn relatief t.o.v. de positie van de radar van **The Frog** op het moment van versturen van de radar-partikels. Deze locatie wordt meegestuurd met het "RADAR START" bericht.

Exit commando

Met "PILOT EXIT" wordt **The Frog** simulator gestopt.

2 The Pilot Requirements

Business requirements

Req.#	Description	F/NF	MoSCoW
RFB-01	The application is written in Java.	NF	MUST
RFB-02	The application communicates with the given SASA communication library to the Frog and Ground Control.	NF	MUST
RRB-03	The application uses a (preferably Java Swing) graphical user interface.	NF	MUST
RFB-04	The application uses a database.	NF	SHOULD

User requirements

Req.#	Description	F/NF	MoSCoW
RFU-01	A driver can drive the Frog in manual mode.	F	MUST
RFU-02	A driver can view the received radar-points in a 2D-graph	F	MUST
RFU-03	A driver can view the position and the direction of the Frog and the driven route on a 2D-map (a given image of the part of the Moon where the Frog is).	F	MUST

System requirements

Req.#	Description	F/NF	MoSCoW
RFS-01	The Frog can send a mission log (a list of coordinates) to Ground Control.	F	MUST
RFS-02	The Frog can send a map of the world (coordinates of obstacles seen so far) to Ground Control on request.	F	MUST
RFS-03	The Frog can drive automatically to a given (preferably received from Ground Control or by pointing that location on the 2D-map) location, driving around obstacles. *Navigation (i.e. path finding with backtracking) is not necessary!*	F	SHOULD
RFS-04	The Frog can drive a pre-programmed mission (a list of drive statements), preferably received from Ground Control.	F	SHOULD
RFS-05	The Frog prevents collisions when driving (in automatic as well as in manual mode).	F	SHOULD

3 Ground Control Requirements

Business requirements

Req.#	Description	F/NF	MoSCoW
RGB-01	The application is written in Java.	NF	MUST
RGB-02	The application communicates with the given SASA communication library to the Pilot.	NF	MUST
RGB-03	The application uses a database.	NF	MUST
RGB-04	The application uses a (preferably Java Swing) graphical user interface.	NF	MUST

User requirements

Req.#	Description	F/NF	MoSCoW
RGU-01	The operator see/ follow the position and the direction of the Frog on a 2D-map (a given image of the part of the Moon where the Frog is), together with the driven route.	F	MUST
RGU-02	The operator can request a map (a list of detected obstacles) from the Frog and see the result on his 2D-map (a given image of the part of the Moon where the Frog is).	F	MUST
RGU-03	The operator can request the Frog to drive to a specific location, preferably by pointing the location on the map.	F	SHOULD
RGU-04	The operator can send a mission log (a sequence of drive commands) to de Frog for it to drive.	F	SHOULD

System requirements

Req.#	Description	F/NF	MoSCoW
RGS-01	A mission can be exported to and imported from a CSV-file	F	MUST
RGS-02	A mission can be saved in the database	F	MUST
RGS-03	A map of collision-points can be exported to and imported from a CSV-file	F	SHOULD
RGS-04	A map of collision-points can be saved in the database	F	SHOULD