

## Software Project

**SpaceZ** industries is in the business of launching rockets and satellites in space for their customers. Every **SpaceZ** space-craft launch has two parts a launch vehicle or the "Rocket" and the payload which could be a satellite. **SpaceZ** has a Deep Space Network (**DSN**) facility containing a Mission-Control system and communication system from which they launch and communicate with their spacecrafts.

**SpaceZ** wants you to design a software system to run their operations. This software system can be classified as follows:

### 1) DSN Software Component Features:

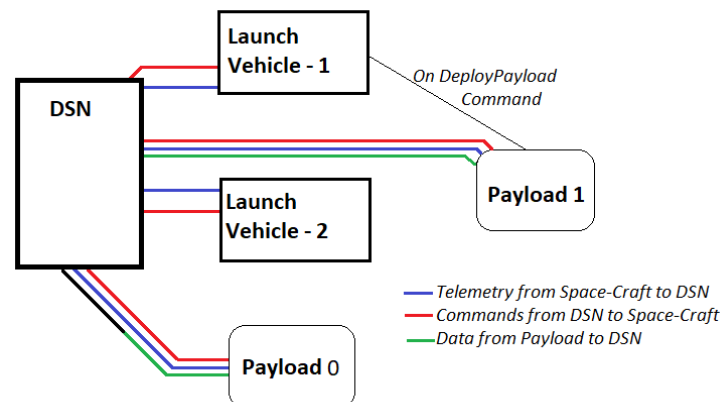
1. Able to show dashboard for
  1. All current active spacecrafts.
  2. All spacecrafts waiting to be launched.
2. Able to select a specific active spacecraft and look at its data.
3. Able to send command to a specific spacecraft.
4. Able to launch a new spacecraft.

### 2) Launch-Vehicle Software Component Features:

1. Able to receive and process commands from DSN.
2. Able to send real-time telemetry of itself back to DSN.

### 3) Payload/Satellite Software Component Features:

1. Able to receive and process commands from DSN.
2. Able to send real-time telemetry of itself back to DSN.
3. Able to send its Data back to DSN.



*Example software components and communication interfaces between them:*

*DSN launched LV-1 and Deployed its Payload, DSN launched LV-2 and not yet deployed its Payload and DSN has Payload 0 with its LV-0 already Deorbited*

## Design Rules:

### Configuration:

- A **Launch-Vehicle** configuration should be defined in a config-file with following information
  1. Name: Name of launch vehicle.
  2. Orbit Info: Radius of orbit in (km).
  3. Payload Config-File: Pointer to the configuration file for payload.
- A **Payload** configuration should be defined in a config-file with following information
  1. Name: Name of payload
  2. Type: Type of payload (Possible Values: Scientific, Communication, Spy)

Data sent by actual "Payload" can be modeled randomly as follows:

**Scientific:** Periodic scientific data of your choice and format

Example,

Solar-Activity in solar-flares per second every 3 seconds    *or*  
Weather-data (%Rain, %Humidity, %Snow) every 1 min.

**Communication:** Periodic communication utilization data of your choice and format

Example,

Bandwidth utilization (Uplink and downlink data rates) every 5 seconds.

**Spy:** Periodic image data of your choice and format

Example:

An image every 10 seconds.

### Launch Sequence:

**DSN** launches a **Launch-Vehicle** as follows:

- User should be able to select a **Launch-Vehicle** configuration file and launch.
- **DSN** software should then start the program/executable for Launch-Vehicle with the selected configuration-file.
- Using the orbit info in the configuration Launch-Vehicle will "fly" till the orbit is reached. This needs to be simulated as follows:
  - Using orbit info launch vehicle will calculate time-to orbit as follows:
$$t = (\text{Orbit Radius in km} / 3600 + 10) \text{ seconds}$$
- After t seconds have elapsed only then **Launch-Vehicle** can then accept "**DeployPayload**" command from **DSN**.
- After t seconds have elapsed **Launch-Vehicle** needs to "update" **DSN** that it has reached its orbit.

## Commands:

**Launch-Vehicle** software should accept following Commands from **DSN**:

- **DeployPayload**: Start the **Payload** software program/executable with the configured **Payload** Config-File.
- **Deorbit**: End the **Launch-Vehicle** software program to simulate that **Launch-Vehicle** has been de-orbited.
- **StartTelemetry**: Start sending (random/realistic up to you) telemetry data every second to **DSN**.
- **StopTelemetry**: Stop sending telemetry data to **DSN**.

**Payload** software should accept following Commands from **DSN**:

- **StartData**: Start sending data configured according to "Type" in **Payload** Config-File to **DSN**.
- **StopData**: Stop sending data to **DSN**
- **Decommission**: End **Payload** software program to simulate that **Payload** has ended its mission.
- **StartTelemetry**: Start sending (random/realistic up to you) telemetry data every second to **DSN**.
- **StopTelemetry**: Stop sending telemetry data to **DSN**.

## Telemetry:

- Telemetry data sent by either Launch-Vehicle or Payload can be random or realistic.
- Each piece of telemetry can include following information
  - Altitude: in km
  - Longitude: in degrees (-90 deg (South) to +90 deg (North))
  - Latitude: in degrees (-180 deg (West) to 180 deg (East))
  - Temperature: in kelvin
  - **Time to Orbit**: in seconds counting down to 0 (This must be the from above t calculation)
- Example:

```
{  
  "altitude" : 400,  
  "longitude" : -45.34  
  "latitude" : -28.34  
  "temperature" : 340,  
  "timeToOrbit" : 45  
}
```

Example guidelines:

Configuration for a Launch-Vehicle and its payload can be as follows:

```
*Launch Vehicle Config*
  Name: Bird-9           // Name of Launch-Vehicle
  Orbit: 600             // Low-earth orbit of 600 km
  PayloadConfig: <path to Payload Config defined below>

*Payload Config*
  Name: GPM              // Name of Satellite (Global Precipitation Measurement)
  Type: Scientific       // Weather data

*Data Sent by Payload every 5 seconds*
  WeatherData: (can be random values)
    Rainfall: 34 mm
    Humidity: 56 %
    Snow: 3in
```

```
*Launch Vehicle Config*
  Name: Bird-Heavy       // Name of Launch-Vehicle
  Orbit: 36000           // Geostationary orbit of 36000 km
  PayloadConfig: <path to Payload Config defined below>

*Payload Config*
  Name: TDRS-11         // Name of Satellite (Tracking Data Relay Satellite)
  Type: Communication    // Bandwidth data

*Data Sent by Payload every 10 seconds*
  CommData: (can be random values)
    Uplink: 40 MBps
    Downlink: 6700 MBps
    ActiveTransponders: 65
```

```
*Launch Vehicle Config*
  Name: Hawk-Heavy       // Name of Launch-Vehicle
  Orbit: 3000            // Polar orbit of 3000 km
  PayloadConfig: <path to Payload Config defined below>

*Payload Config*
  Name: RO-245          // Name of Satellite (Reconnaissance satellite)
  Type: Spy              // Image data

*Data Sent by Payload every 30 seconds*
  ImageData: (can be random images)
    Image: bytes of a jpg/png image
```

## **Submission Criteria:**

- Implementation can be done in any choice of language/platform provided it can be run and tested easily on a Windows Machine.
  - Preferred option is .NET platform with C# language and Visual Studio for solution. WPF for UI side, WCF for inter-process communication.
- Please provide a **complete working solution** and instructions on how to run.
- If you are using any third-party dependencies or libraries, please provide instructions on how to set them up before we can run and test your submission. Please include them in your submission as well.
- Your submission must have ***distinct programs/executables*** for DSN, Launch-Vehicle and Payload and demonstrate the required communication between them.
- Your submission must include **complete source-code** and instructions on how to compile and build it.
- Only the DSN software component has some user-interface requirements, the other two components need not have user-interface.