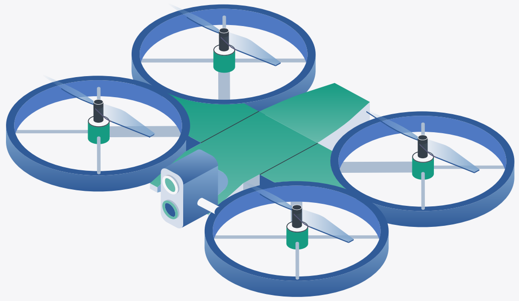


State-Based Testing of Flight Controllers

Catarina Gamboa and Simon Chu

Domain

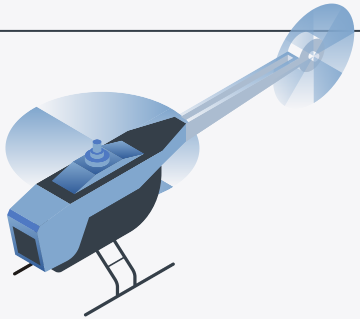
Flight controller has been deployed to more life-critical use cases like search and rescue and medical supply delivery, and shifting towards more autonomy.

Problem

Testing techniques in this space remains limited

- Deploy on Real-world Hardware (costly)
- Simulation-based Testing (costly CI process, not explainable)
- Flight Controller API Unit Test (lack of integration testing in CI)

Existing techniques do not consider the statefulness of the drone and overlook some of the bugs resulted from calling certain API from the wrong state.



Problem

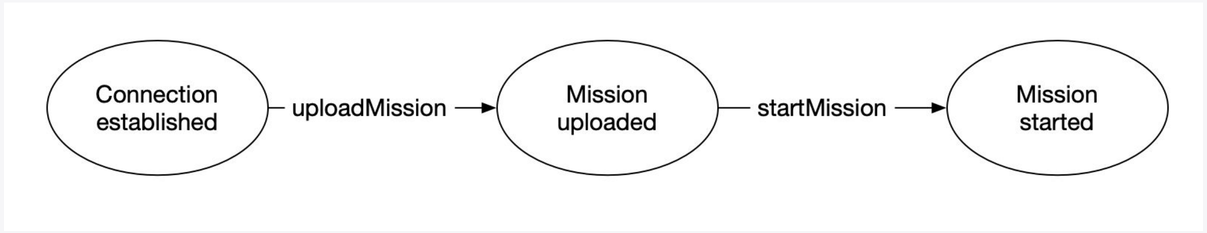
Some API calls are dependent on the drone state

✗
```
Mission mission = drone.getMission();
mission.startMission();
```

✓
```
Mission mission = drone.getMission();
mission.uploadMission(...);
mission.startMission();
```

Mission has to be uploaded before started.

Otherwise the drone may remain on a holding pattern or execute previously cached mission causing crashes or under undefined behavior


$$\begin{tikzcd} \text{Connection established} \arrow[r, "\text{uploadMission}"] & \text{Mission uploaded} \arrow[r, "\text{startMission}"] & \text{Mission started} \end{tikzcd}$$

Approach

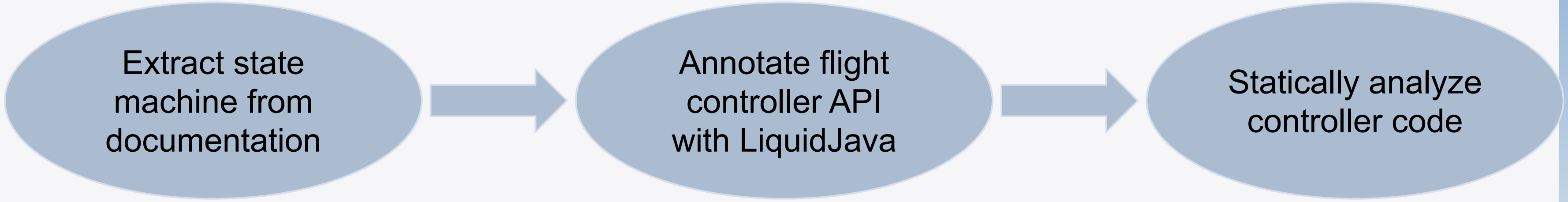
Lightweight

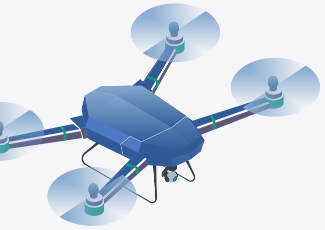
Seamless integration

Explainable

Capture statefulness

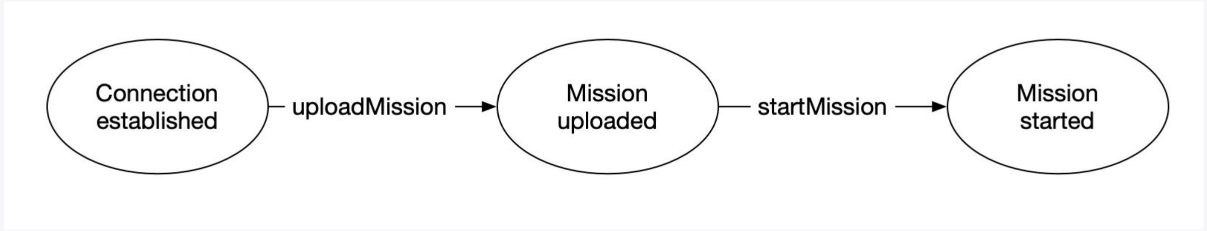
Approach


Extract state machine from documentation
Annotate flight controller API with LiquidJava
Statically analyze controller code

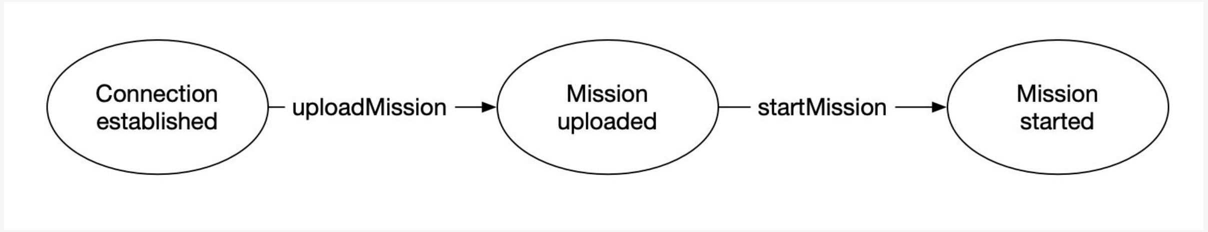


We chose to annotate **MAVSDK** because it is one of the two most popular open-source drone controller frameworks.

Example


Connection established
uploadMission
Mission uploaded
startMission
Mission started

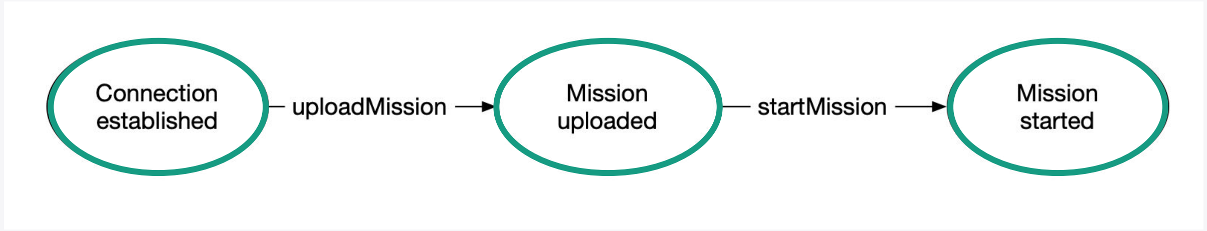
Example



Mission

```
@ExternalRefinementsFor("io.mavsdk.mission.Mission")
```

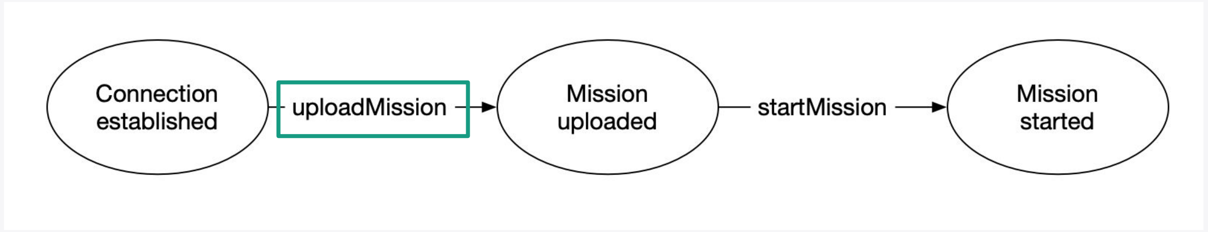
Example



Mission

```
@ExternalRefinementsFor("io.mavsdk.mission.Mission")
@StateSet({...,"connectionEstablished", "missionUploaded", "missionStarted"})
```

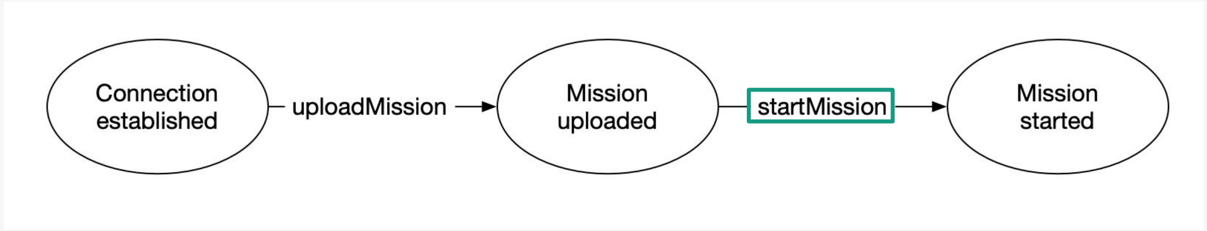
Example



Mission

```
@ExternalRefinementsFor("io.mavsdk.mission.Mission")
@StateSet({...,"connectionEstablished", "missionUploaded", "missionStarted"})
public interface MissionControllerRefinements {

      @StateRefinement(from="connectionEstablished(this)",to="missionUploaded(thi
s)")
      public void uploadMission(MissionPlan missionPlan);




}
```

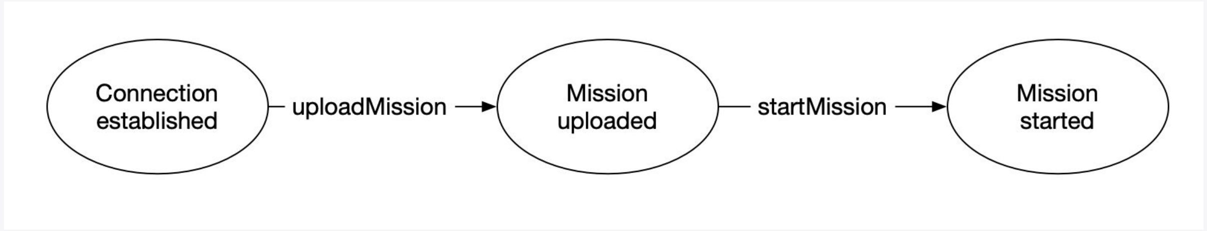
Example



Mission

```
@ExternalRefinementsFor("io.mavsdk.mission.Mission")
@StateSet({...,"connectionEstablished", "missionUploaded", "missionStarted"})
public interface MissionControllerRefinements {

      @StateRefinement(from="connectionEstablished(this)",to="missionUploaded(thi
s)")
      public void uploadMission(MissionPlan missionPlan);

      @StateRefinement(from="missionUploaded(this)", to="missionStarted(this)")
      public void startMission();

      …
}
```

Example



Mission

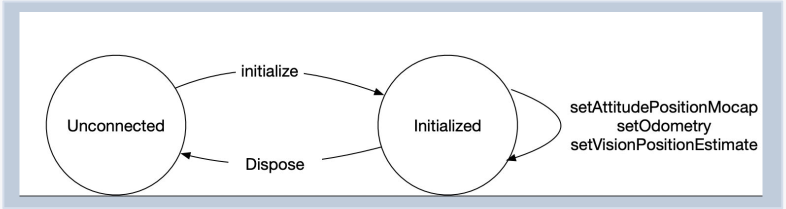
```
@ExternalRefinementsFor("io.mavsdk.mission.Mission")
@StateSet({...,"connectionEstablished", "missionUploaded", "missionStarted"})
public interface MissionControllerRefinements {

      @StateRefinement(from="connectionEstablished(this)",to="missionUploaded(thi
s)")
      public void uploadMission(MissionPlan missionPlan);

      @StateRefinement(from="missionUploaded(this)", to="missionStarted(this)")
      public void startMission();

      …
}
```

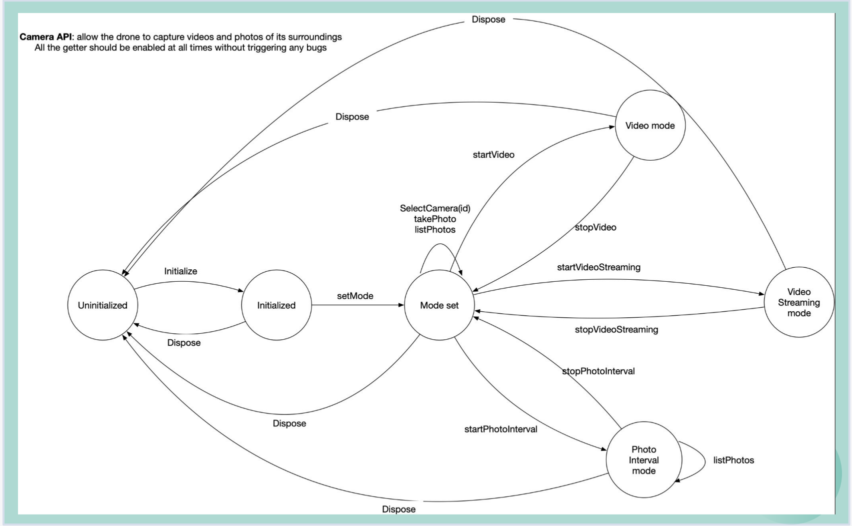
Client

```
 Mission mission =
       drone.getMission();
 //mission.uploadMission(...);
❌mission.startMission();
```

7 classes modelled

	States	Transitions
Mocap	2	6
Geofence	3	5
Mission	4	14
Offboard	4	17
Camera	6	15
FTP	5	19
FollowMe	6	20




Camera API: allow the drone to capture videos and photos of its surroundings
All the getter should be enabled at all times without triggering any bugs

Evaluation

01

Real Examples

For each class we found 1-3 real code examples

02

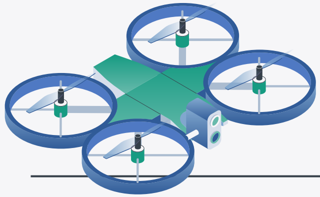
Change the examples

1. Asynchronous -> synchronous
2. Change to Java

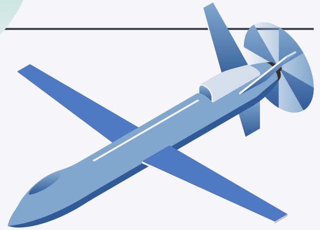
03

Verify with LiquidJava

We have found 4 bugs where code does not follow the declared protocols:
2 in **Camera**; 1 in **Ftp**; 1 **FollowMe**



Insights



LiquidJava

- Handling asynchronous code and aliasing
- State Transitions based on parameters

Flight Controllers

- Modelling the protocols is helpful to find bugs
- Other APIs, for example non-open source and used in industry, should take advantage of these verification techniques

Thank you!