Material

Let's consider a UAV that performs an item delivery mission. Each UAV undergoes an initialization step before it is ready for take-off. It can fly to a specific location and approach a target destination. At that point, it may switch into the *dropping-item* state, lower its altitude, and drop an item. Once all locations have been visited, the drone can return to launch or fly to another target. Figure 3 shows a state transition diagram indicating the states of this delivery scenario.

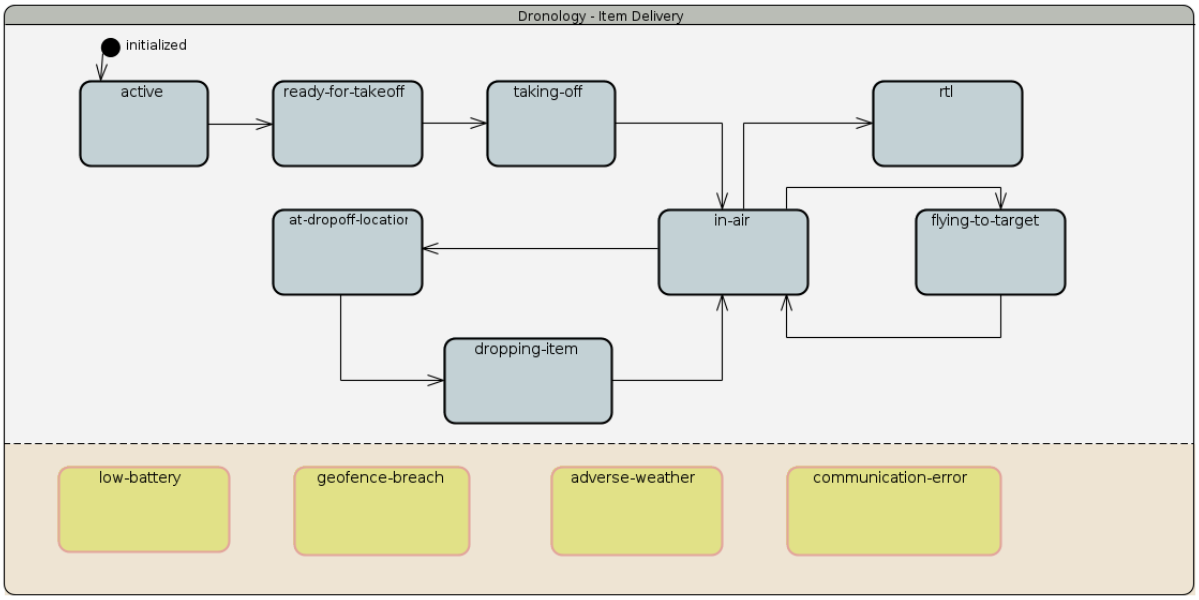

Figure 3: State Transition Diagram for the Dronology Item-Delivery Scenario

To ensure that the fleet of drones operates correctly and does not collide with each other, it is necessary to monitor their behavior. One challenge when implementing this scenario is that different states require different information to be monitored. For instance, during the initialization phase, certain start-up checks are important to monitor, whereas they can be switched off when the drone is in air.

We developed a Domain-Specific Language that can help users to define the monitoring properties that should be monitored, as well as the period of monitoring and whether the monitored information should be sent to a central ground control station.

I: Preplanned Rules

In order to customize monitors in different states, "Preplanned Rules" can be specified in the DSL. Each Rule has a unique name, a specific trigger (the state where it becomes active), and a set of monitors it customizes. A "Monitor" describes a specific property of the system that is monitored, e.g., the Property "GPSLocation". For each Monitor, a period can be specified, as well as the scope, which indicates whether the property is only processed locally or it should be sent to the central server for further analysis. Additionally, for a preplanned rule in the DSL the *application context* can be specified, i.e., a rule can be applied to any UAV in the system (or to a UAV with a specific ID if needed). Examples of Preplanned Rules for the Dronology System are shown in Listing 1.

Listing 1: Example of Preplanned Rules for the Dronolgy System

```
Rule PREPLANNED "Rule␣1" applies forall // a preplanned rule with the name "Rule 1" used for all drones
 Trigger Context["Drone.active"] ENTRY // it is triggered as soon as the state "active" is activated
     Monitor "Drone.GPSLocation":
      CHANGE_PERIOD: every 3 seconds  // once every three seconds,
      CHANGE_SCOPE: scope local      //the GPS location should be locally monitored
     Monitor "Drone.Startupchecks":
      CHANGE_PERIOD: every 4 seconds  // once every four seconds,
      CHANGE_SCOPE: scope central    //the results of the drone start-up checks should be
;                                     //reported centrally to the ground control station

Rule PREPLANNED "In␣air␣checks"  applies forall
 Trigger Context["Drone.in-air"] ENTRY // when a drone enters the state "in-air"...
     Monitor "Drone.GPSLocation":
      CHANGE_FREQUENCY: every 1 seconds
      CHANGE_SCOPE: scope central //... the GPS location should be centrally monitored once every second
;     //...
```

II: Ubiquitous Rules

Besides the "normal course" of a use case, that is reflected by the different states in the state transition diagram, situations can occur that require special treatment and a deviation from the preplanned monitoring rules.

For example, when a UAV shows low battery, two UAVs come in proximity to each other, or when a UAV accidentally enters a no-fly zone, monitoring and data collected need to be adapted accordingly.

For these cases, our DSL allows us to specify *Ubiquitous Rules* which can be triggered by states outside of the state transition diagram. Specifying these rules works the same way as for Preplanned rules. However, they are not part of predefined states and transitions in the state chart but can occur at any time during the mission when a certain ubiquitous or error state is triggered.

An example is shown in Listing 2. The comments explain the meaning of this rule.

Listing 2: Ubiquitous Rule – Battery Warning

```
Rule UBIQUITOUS "Battery_State_Warning" applies forall //The rule holds for all drones
 salience 1 //The rule has priority/salience 1.
            //If multiple ubiquitous rules can be applied, the one with the highest salience is selected
  Trigger Event['Drone.battery-state-warning'] ENTRY //When the event 'battery-state-warning' is
      triggered
      Monitor "Drone.GPSLocation":
       CHANGE_SCOPE: scope local
       CHANGE_PERIOD: every 5 seconds //The GPS Location should be locally monitored every 5 seconds.
      Monitor "Drone.BatteryStatus":
       CHANGE_SCOPE: scope central
       CHANGE_PERIOD: every 2 seconds //The Battery Status should be centrally monitored every 2 seconds.
;
```

III: Default Values

To reduce the configuration effort of the monitors, (i.e., each monitor should be present in every rule), the DSL allows us to set *Default Values* for monitors.

This means that if a monitor is not configured in a certain rule, its default value is used. A monitor default value can (1) either be *off*, i.e., no data is collected (period 0), or (2) *keep*, which means the monitor does not change and keeps the same value (both period and scope) as it had before the transition to the new state occurred.

In the default values shown in Listing 3, "Drone.GPSLocation" has the value "keep", which entails that for the example rules in Listing 1, the property "Drone.GPSLocation" would be continued to be monitored with the same scope and period as defined in a previous state. Note that if no default value is defined, the monitoring property needs to be explicitly defined in all preplanned rules.

Listing 3: Examples of Default Values

```
Default "Drone.Startupchecks" off;
Default "Drone.GPSLocation" keep;
```

IV: Assumptions

To support basic validation of rules, the DSL allows users to define *Assumptions* about the monitored properties that should hold. These assumptions require domain knowledge or knowledge about the system, and define minimum guaranteed values for Monitors. This means that the system can guarantee that values are updated in a specific interval, and that new values are available for the monitors to be sent to the monitoring framework.

For example, Listing 4 specifies that the minimum interval with which we can expect updates is 2 seconds for *"Drone.BatteryStatus"* messages. A rule specifying a Monitor with a period of, for example, 1 second for *"Drone.BatteryStatus"* would be invalid, as data is not provided frequently enough by the system.

Listing 4: Assumption specification for Monitoring properties

```
Assumption "Drone.BatteryStatus" MIN_PERIOD: 2 seconds;
Assumption "Drone.GPSLocation" MIN_PERIOD: 0.5 seconds;
```

Cheat Sheet

Listing 5 shows an example using all the language elements we have introduced.

Listing 5: Example of a DSL file for the Dronolgy System

```
namespace "Drone";

/* Assumptions */
Assumption "Drone.BatteryStatus" MIN_PERIOD: 2 seconds;
Assumption "Drone.GPSLocation" MIN_PERIOD: 0.5 seconds;

/* Default Values */
Default "Drone.Startupchecks" off;
Default "Drone.GPSLocation" keep;

/* Preplanned Rules */
Rule PREPLANNED "Rule␣1" applies forall // a preplanned rule with the name "Rule 1" used for all drones
 Trigger Context["Drone.active"] ENTRY // it is triggered as soon as the state "active" is activated
     Monitor "Drone.GPSLocation":
      CHANGE_PERIOD: every 3 seconds  // once every three seconds,
      CHANGE_SCOPE: scope local      //the GPS location should be locally monitored
     Monitor "Drone.Startupchecks":
      CHANGE_PERIOD: every 4 seconds  // once every four seconds,
      CHANGE_SCOPE: scope central    //the results of the drone start-up checks should be
;                                     //reported centrally to the ground control station

Rule PREPLANNED "In␣air␣checks"  applies forall
 Trigger Context["Drone.in-air"] ENTRY // when a drone enters the state "in-air"...
     Monitor "Drone.GPSLocation":
      CHANGE_PERIOD: every 1 seconds
      CHANGE_SCOPE: scope central //... the GPS location should be centrally monitored once every second
;


/* Ubiquitous Rules */
Rule UBIQUITOUS "Battery_State_Warning" applies forall //The rule holds for all drones
 salience 1     //The rule has priority/salience 1.
           //If multiple ubiquitous rules can be applied, the one with the highest salience is selected
 Trigger Event["Drone.low-battery"] and Event["Done.geofence-breach"] //When the event 'low-battery' and
      'geofence-breach' occur.
  Monitor "Drone.GPSLocation":
   CHANGE_SCOPE: scope local
   CHANGE_PERIOD: every 5 seconds //The GPS Location should be locally monitored once every 5 seconds.
  Monitor "Drone.BatteryStatus":
   CHANGE_SCOPE: scope central
   CHANGE_PERIOD: every 2 seconds //The Battery Status should be centrally monitored once every 2 seconds
       .
;
```

1.1 Create Monitoring Rules (Phase 2 - Task 1)

This task is concerned with specifying monitoring rules for a scenario in our domain-specific language. You can go through the following steps and work with the language editor in parallel.

Monitoring Property	Description	Min. Assumed Period
Drone.GPSLocation	The GPS Location (longitude, latitude, altitude) of a drone	once every 0.5s
Drone.State	The stat of a drone and its current task	
Drone.HomeLocation	The initial takeoff location of the drone	
Drone.Startupchecks	Information about satisfied startup checks (geofence, max height, ...)	
Drone.BatteryStatus	The battery status (percentage, voltage, ...) of the drone	once every 2s
Drone.FlightControllerData	Information from the flight controller (gyroscope, and othe rsensors)	
Drone.Dronecommand	Commands sent from the control statation to the drone	

– Step 1: Assumption
Based on the guaranteed monitoring period by the system, create Assumptions for the minimum period of the two properties Drone.GPSLocation and Drone.BatteryStatus. If nothing is stated in the table, no assumption needs to be specified.

– Step 2: Default Values
Monitoring for all the above mentioned properties should be **turned off** by default. Define the default values in the Domain-Specific Language.

– Step 3: Preplanned Rules
Define pre-planned rules for the following states. All rules should apply for all drones and upon entry into a certain state.

1. Whenever a drone enters the state **active**, its *GPS location should be monitored centrally every 0.5 seconds*, its *Startup Checks checks centrally every 1 second*, and the *Dronestate centrally every 2 seconds.*
2. Whenever a drone is **ready for take off**, the *GPS location should be centrally monitored every 10 seconds*, and incoming *Drone commands locally every second.* The drone's *Startup checks should not be monitored* anymore (0 seconds).

– Step 4: Ubiquitous Rules

Define the following ubiquitous rules:

1. When the **low battery** event is triggered, the *GPS location of the drone should be locally monitored every 3 seconds*, and the *Battery State should be monitored centrally every 2 seconds.* The rule has a salience level of 3.
2. When both a **geofence breach** and a **low battery** event occur, with both active, the *GPS location of the drone should be centrally monitored every 2 seconds*, the *Flight Controller data* should also be *monitored centrally every 2 seconds*, and the *Battery State* should also be *monitored centrally* every 2 seconds. The rule has a salience level of 5.

1.2 Glitch detector (Phase 2 – Task 2)

Now, we will use a glitch detector task, in which we show you an example specification in our DSL that may include errors. Your task is to go through the rules, describe what you see, and identify potential faults. The comments indicate what the specified instance of the DSL is supposed to describe. Faults are not syntax-related (caused by missing symbols or a strange order of keywords), but are semantic issues (resulting in undesirable monitoring behavior or conflicts between specified rules).

– Rule Description

- The preplanned rule "return-to-launch" should be used for all drones in the fleet.
  It is triggered when the state "rtl" (return to launch) is entered.
  The GPS location should be monitored centrally once every 0.5 seconds.
  The Dronestate should be locally monitored every 3 second.

- The ubiquitous rule "battery-and-geofence" should be used for all drones in the fleet.
  Its salience value is 5.
  It is triggered when the events "Drone.low-battery" and "Drone.geofence-breach" are triggered.
  The GPS location should be monitored locally once every 3 seconds.
  The Dronestate should be monitored centrally once every second.
  The battery should be monitored locally once every 10 seconds.