

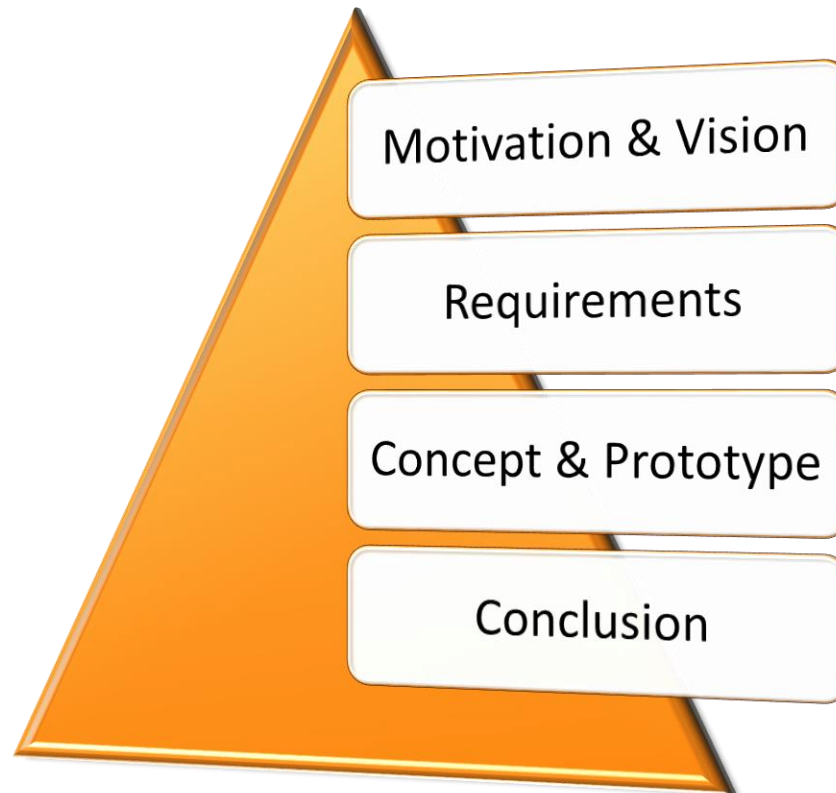


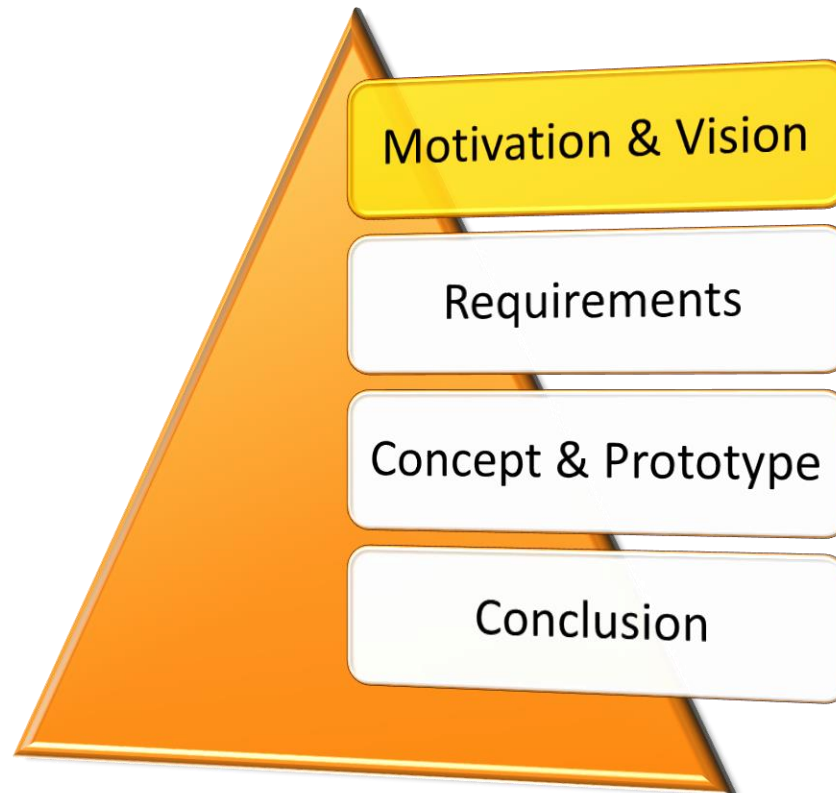
Towards a graphical language for quadrotor missions

Benjamin Schwartz, **Ludwig Nägele**, Andreas Angerer, Bruce A. MacDonald



Universität
Augsburg
University





Current situation

- Hard-coded UAV missions in general-purpose programming languages
- Increasing number of application domains (agriculture, archaeology, ...)
- Graphical end-user solutions with limited functionality



APM Planner

- ✓ Route defined by waypoints
- ✓ Commands specificable for each waypoint
- ✗ No parallel activities
- ✗ No branching
- ✗ No complex movements (e.g. obstacle avoidance)

Distance: 0.8993 km
Prev: 257.36 m
Home: 391.27 m

Waypoints

	Command	Delay	Hit Rad	Yaw Ang	Lat	Long	Alt	Delete	Up	Down
1	WAYPOINT	0	0	0	40.1312555	-105.1109326	100	X		
2	WAYPOINT	0	0	0	40.1314442	-105.1090014	100	X		
3	WAYPOINT	0	0	0	40.1309684	-105.1076925	100	X		
4	WAYPOINT	0	0	0	40.1297133	-105.1081109	100	X		
5	WAYPOINT	0	0	0	40.1294180	-105.1104605	100	X		

Mouse Location
Lat 40.12887660
Long -105.1075208
Alt 1525

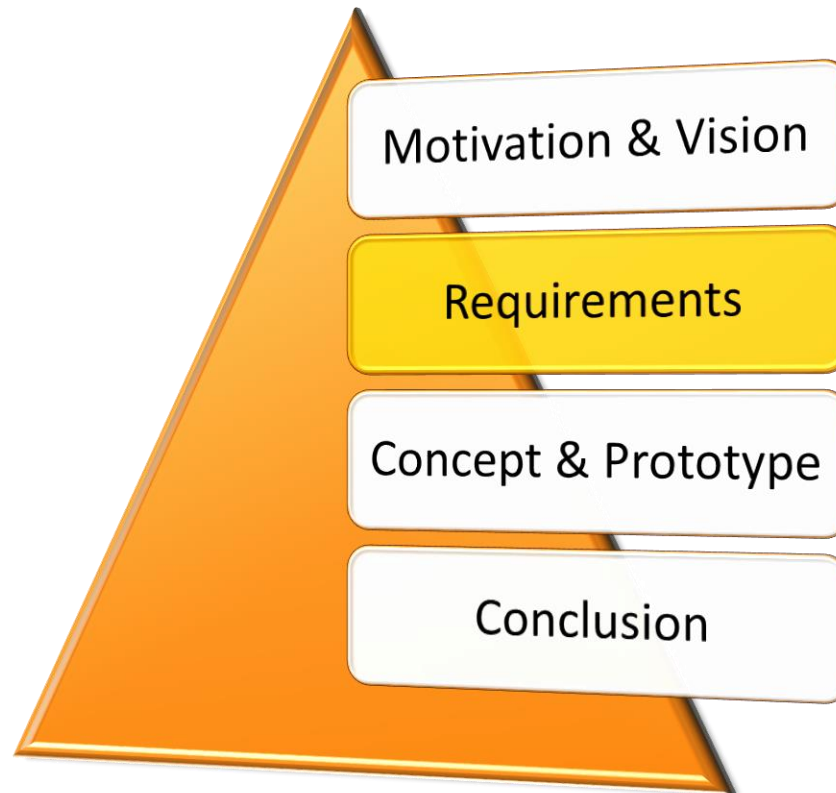
Home Location
Lat 40.13040239
Long -105.1116621
Alt (abs) 20

Current situation

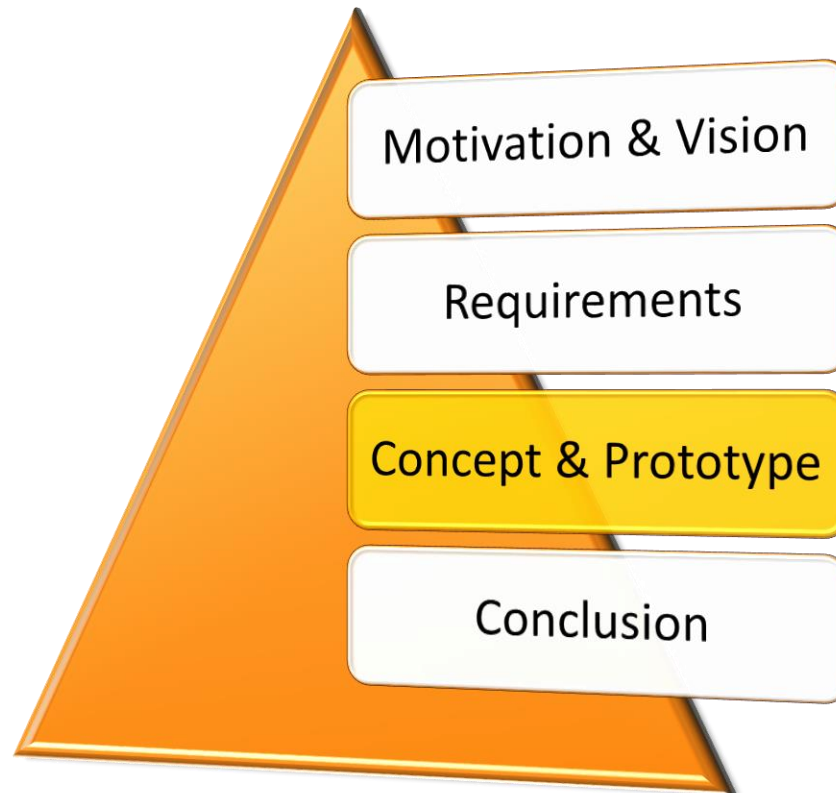
- Hard-coded UAV missions in general-purpose programming languages
- Increasing number of application domains (agriculture, archaeology, ...)
- Graphical end-user solutions with limited functionality

Vision

- **Intuitive** mission definitions by **non-programmers**
- **Powerful** interface for **programmers**
- Hardware-**independent** specification of missions (or even simulator-independent)



- **Graphical language** for intuitive mission definition by non-programmers
- **Clear representation** of the main workflow
- Sufficient **expressiveness** needed for common use cases: Branches, loops, parallel actions
- Concept of **extensibility** of the language for programming experts
- Extensions in turn **reusable** by non-programmers
- Hardware-**independent**: Applicable to different target platforms or simulators (programming languages, frameworks, etc.)



- Specification of a graphical language for quadrotor missions
- Working editor in eclipse for the language (GMF)
- A code generator for MORSE simulator code
Only partial support yet (evaluation purpose)

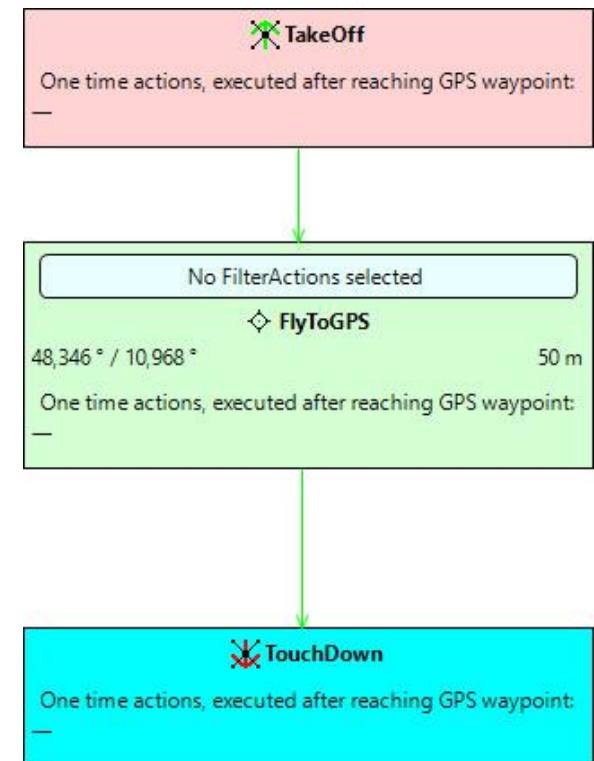
- Separation between:
 - => **Workflow**: Description of waypoints, branching
(stable)
 - => **Actions**: Special behaviour
(modifyable, extensible)
- **Workflow** elements are defined by **meta-model**
- **Action interfaces** are defined by **meta-model**;
Action implementations are part of **model**

Routing elements

- Basic quadrotor functionality
- Describe the main workflow
- States and arrows

Examples:

TakeOff, FlyTo(GPS), HoldPosition,
TouchDown

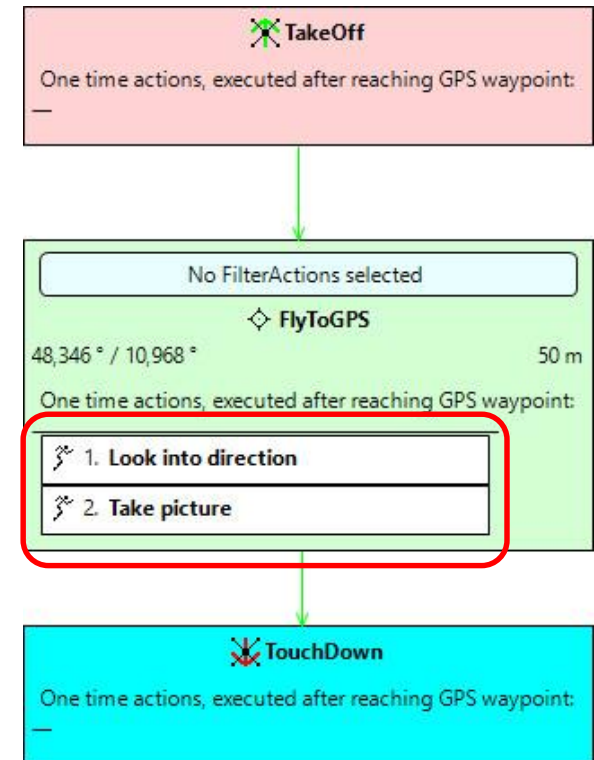


OneTimeActions

- Extended functionality
- Annotated to **routing elements**
- Executed once

Examples:

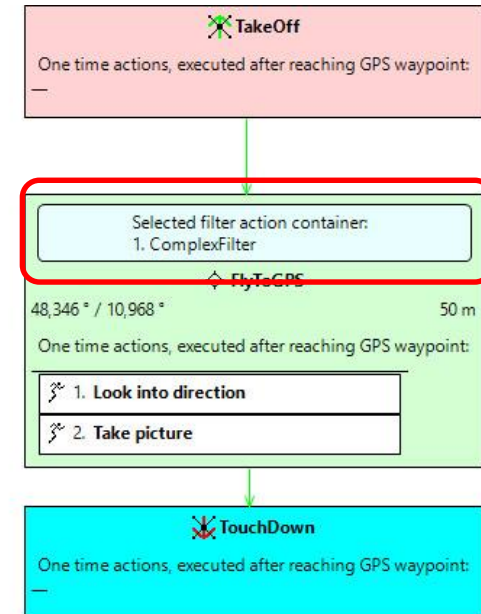
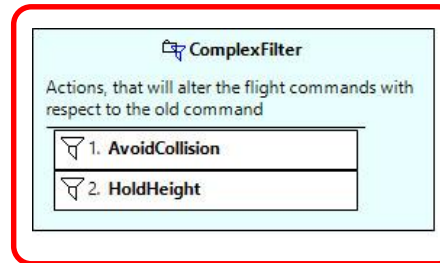
Look into direction, Take picture, ...



Filter elements and FilterActions

- Influence quadrotor flight commands
- Grouping of multiple FilterActions in Filter elements
- Filter elements are annotated to routing elements
- Running synchronously, internal prioritisation

Examples: AvoidCollision, HoldHeight, ...



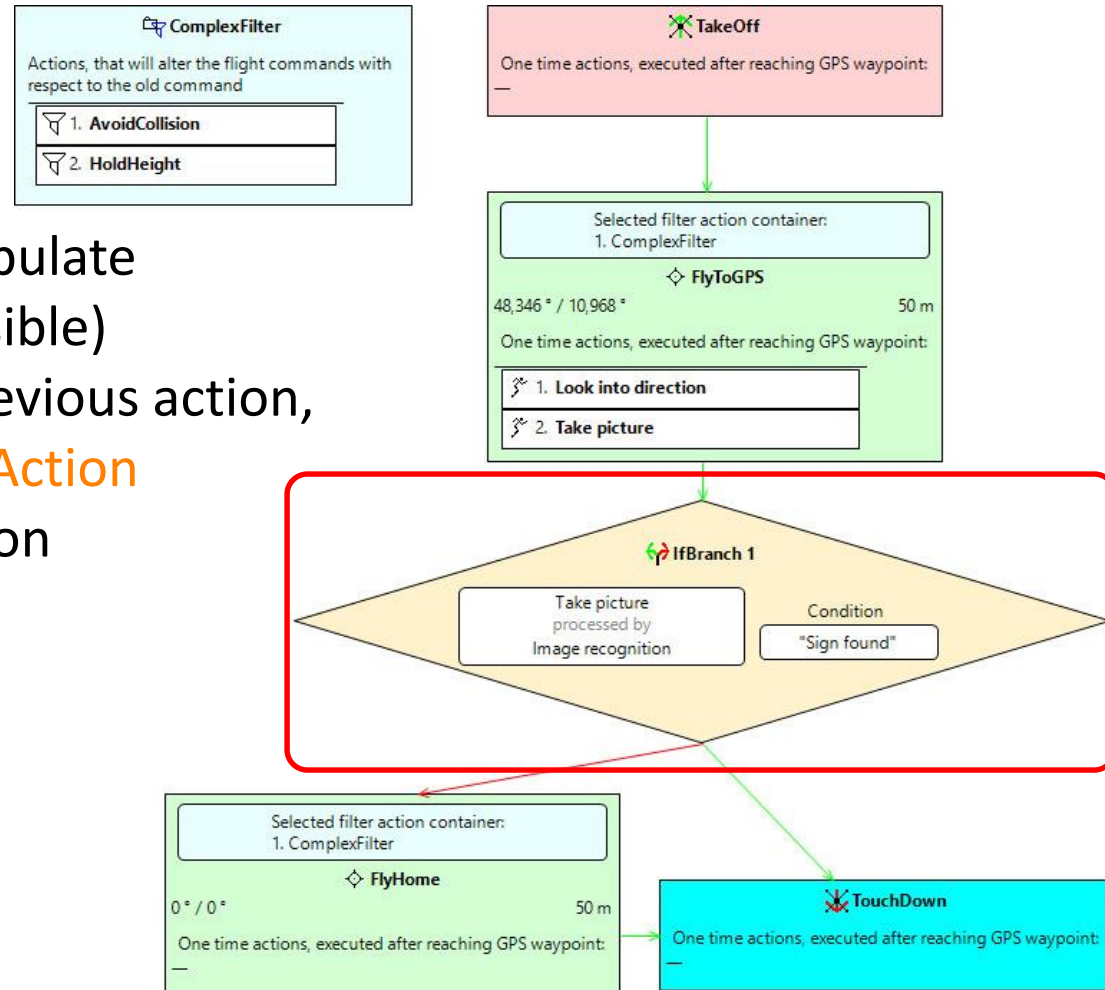
Concept – language elements

IfBranch elements and ProcessingActions

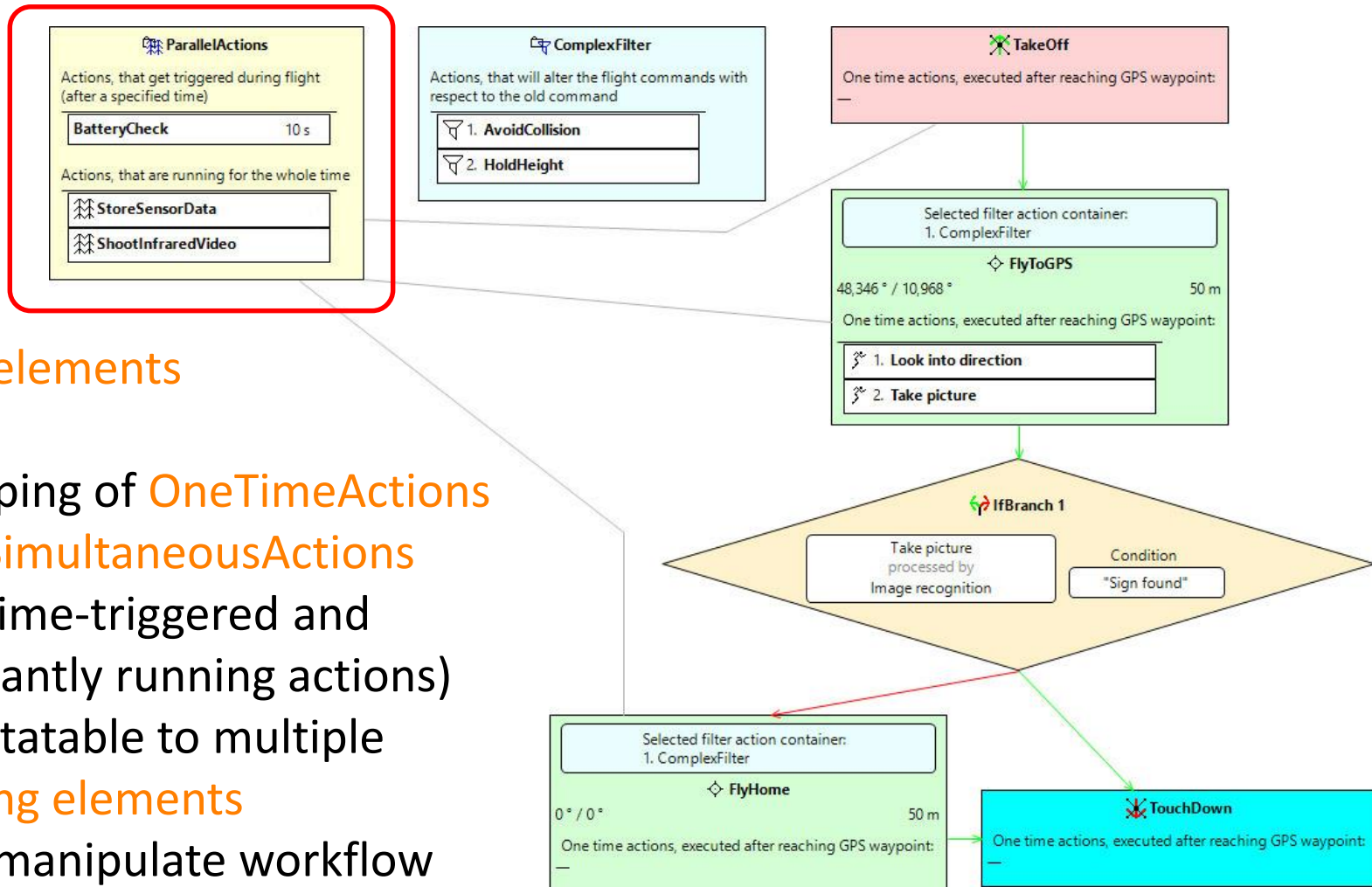
- IfBranch elements manipulate workflow (loops are possible)
- Reference to result of previous action, processed by ProcessingAction
- Comparison with condition

Examples:

Image recognition, ...



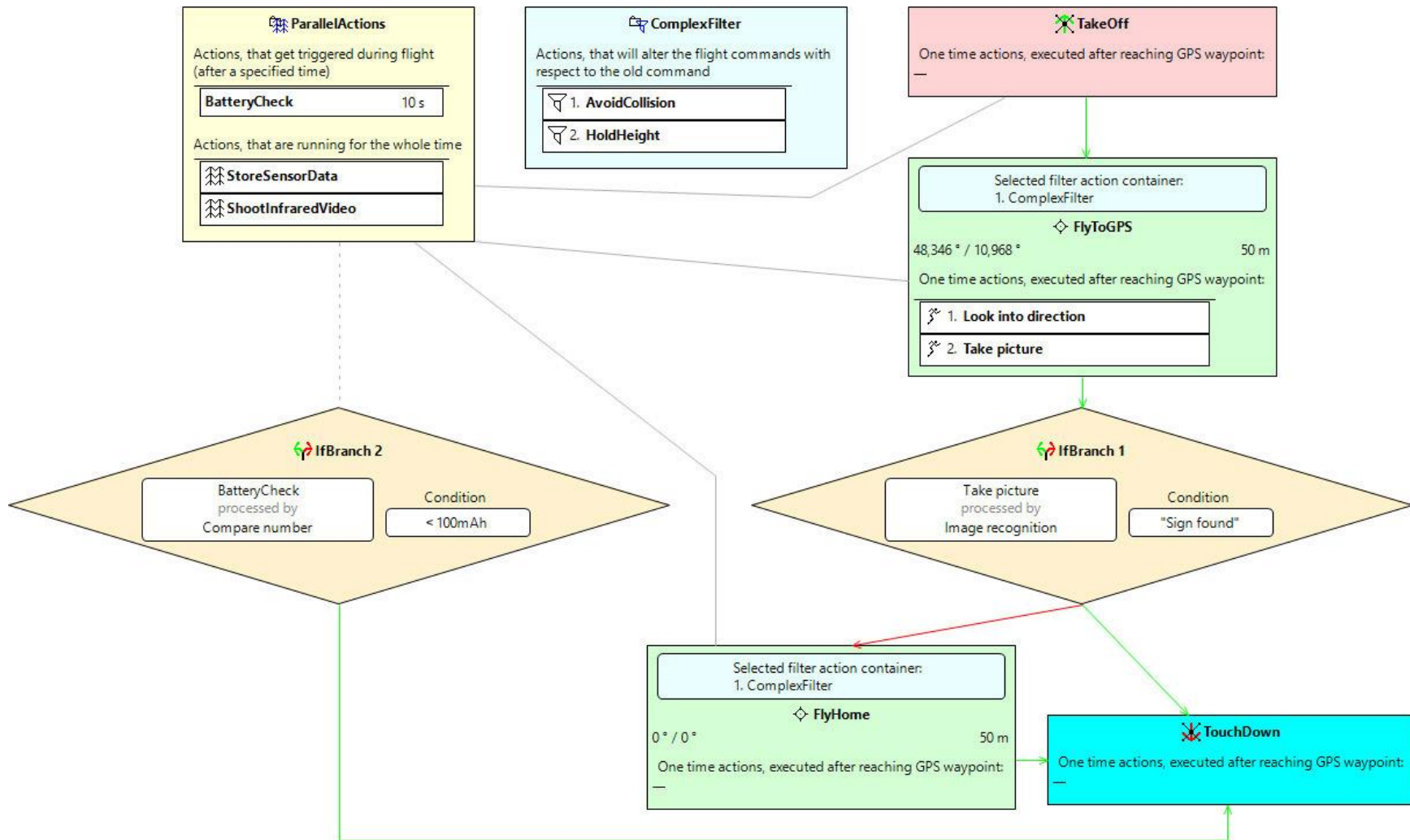
Concept – language elements



Parallel elements

- Grouping of **OneTimeActions** and **SimultaneousActions** (i.e. time-triggered and constantly running actions)
- Annotatable to multiple **routing elements**
- May manipulate workflow

Concept – language elements



Prototype

Toolbox

Dashboard

Project Explorer

Properties view

Package Explorer

- Quad
 - src
 - de.quadrotor
 - de.quadrotor.actions.filter
 - AvoidCollision.java
 - HoldHeight.java
 - de.quadrotor.actions.onetime
 - BatteryCheck.java
 - LookIntoDirection.java
 - TakePicture.java
 - de.quadrotor.actions.processing
 - CompareNumber.java
 - ImageRecognition.java
 - de.quadrotor.actions.simultaneous
 - ShootInfraredVideo.java
 - StoreSensorData.java
 - default.quadcopter
 - JRE System Library [JavaSE-1.8]

default.quadcopter

ComplexFilter
Actions that will filter the flight commands with respect to the old command

- 1. AvoidCollision
- 2. HoldHeight

TakeOff
One time actions executed after reading GPS waypoint

Selected filter action container: 1. ComplexFilter

FlyToGPS
08.055° / 10.998° 20 m
One time actions executed after reading GPS waypoint

- 1. Look into direction
- 2. Take picture

IfBranch 1

- Take picture (triggered by image recognition)
- Cancellation: "Sign found"

Selected filter action container: 1. ComplexFilter

FlyToGPS
0° / 0° 20 m
One time actions executed after reading GPS waypoint

Touchdown
One time actions executed after reading GPS waypoint

Palette

Workflow Elements

- Fly to using GPS coordinates
- Take off
- Touch down
- Fly to
- Generic routing element

Container

- for parallel actions
- for filter rules

Branching

- If branch

Properties

GenericOneTimeAction

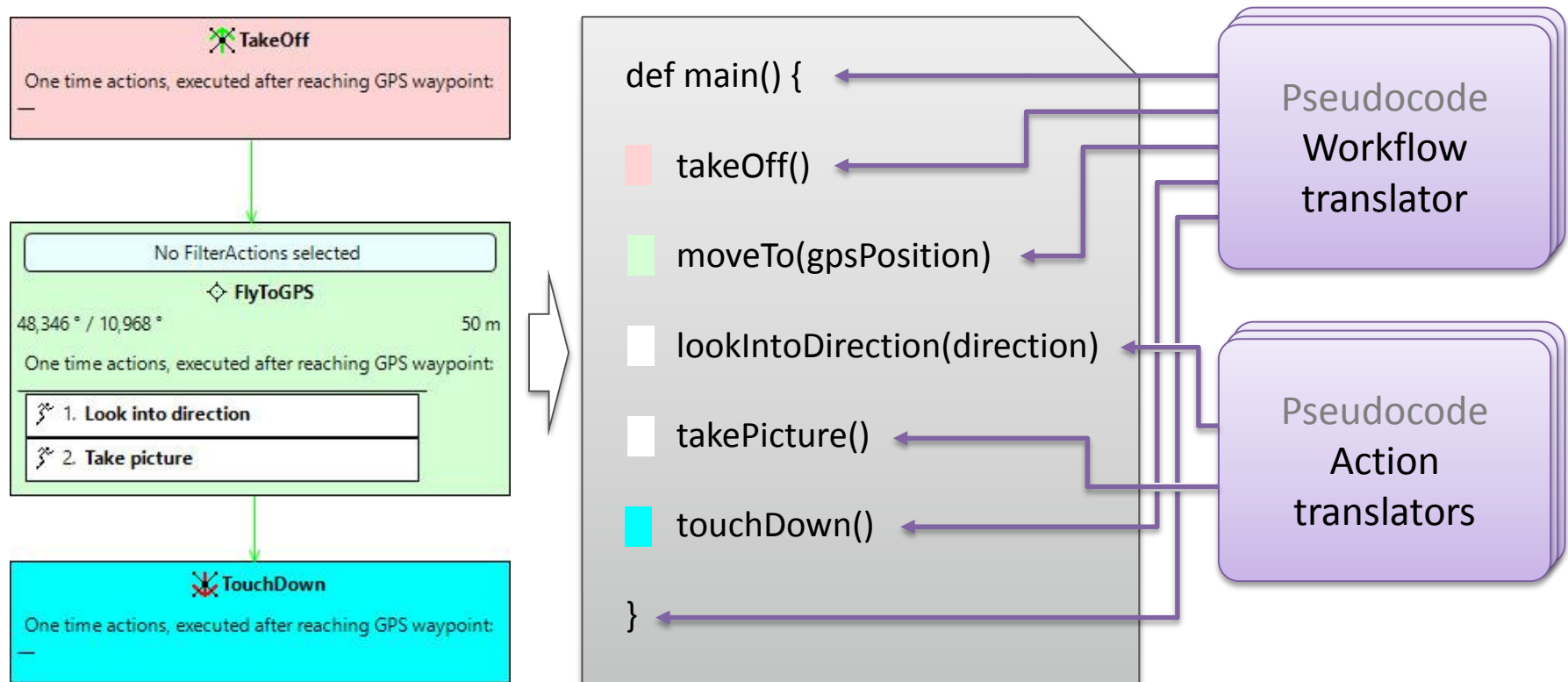
Core	Property	Value
Appearance	Action Type	Take picture
	Name	Take picture
	Picture Quality	100
	Resolution	

Extensibility by new actions

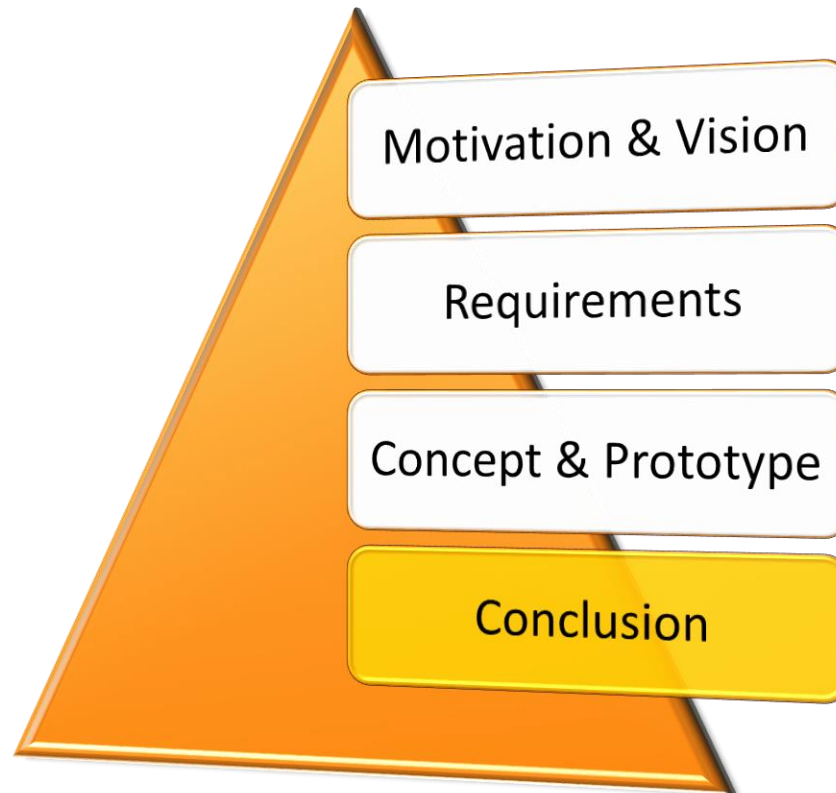
The screenshot displays a software development environment with three main panels:

- Package Explorer:** Shows a project structure for 'Quad'. Under 'src', there are several packages. The package 'de.quadrotor.actions.filter' contains 'TakePicture.java', which is highlighted with a red box. A red arrow points from this box to the 'Take picture' property in the Properties panel.
- Workflow Diagram:** Shows a sequence of actions. A green box labeled 'Selected filter action container: 1. ComplexFilter' contains a 'FlyToGPS' action with coordinates '48,346 ° / 10,968 °' and a distance of '50 m'. Below this, a list of 'One time actions, executed after reaching GPS waypoint:' includes '1. Look into direction' and '2. Take picture'. The 'Take picture' action is highlighted with a red box. Below the green box is a yellow box labeled 'IfBranch 1' with a 'Take picture' action and a 'Condition' box.
- Properties Panel:** Shows the properties of the 'GenericOneTimeAction'. The 'Action Type' property is set to 'Take picture'. The 'Name' property is set to 'Variable check'. The 'Picture Quality' property is set to 'Look into direction'. The 'Resolution' property is set to 'Take picture'. The 'Take picture' property is highlighted with a red box.

Code generation



In our scope: Python-codegenerator for MORSE simulator
(quadrotor simulation)



- Language and editor for graphically specifying quadrotor missions
- Extensible by new (reusable) actions
- Thus, appropriate for skilled programmers and non-programmers
- First code generator to MORSE simulator code showed proof of concept

Future work:

- Distinction between target-platform independent and dependent semantics (limitations on target platforms: no multi-threading etc.)
- Complete evaluation on (different) real hardware

