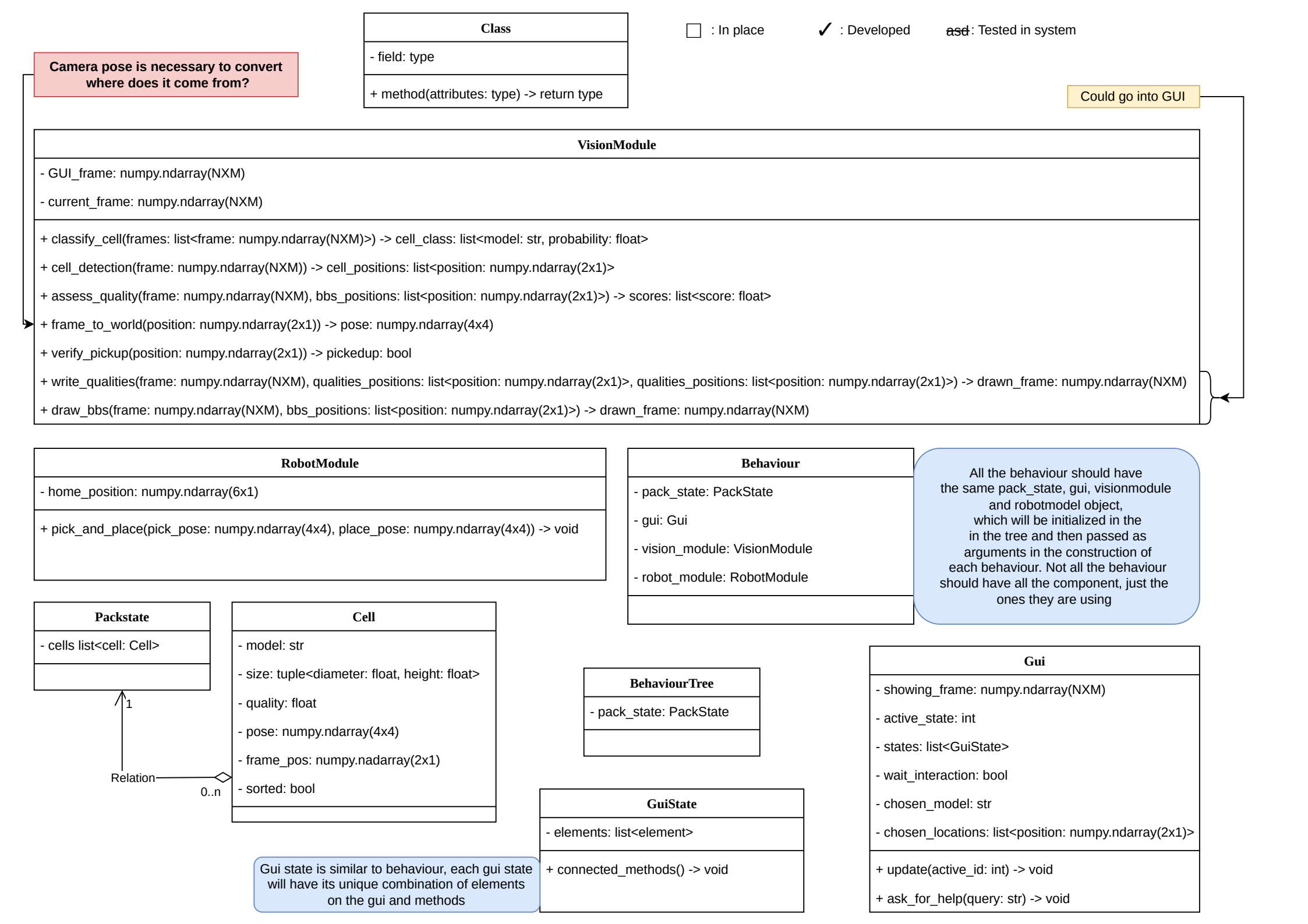
Class
- field: type
+ method(attributes: type) -> return type
: In place
: Developed
asd: Tested in system
Camera pose is necessary to convert where does it come from?
Could go into GUI
VisionModule
- GUI_frame: numpy.ndarray(NXM)
- current_frame: numpy.ndarray(NXM)
+ classify_cell(frames: list<frame: numpy.ndarray(NXM)>) -> cell_class: list<model: str, probability: float>
+ cell_detection(frame: numpy.ndarray(NXM)) -> cell_positions: list<position: numpy.ndarray(2x1)>
+ assess_quality(frame: numpy.ndarray(NXM), bbs_positions: list<position: numpy.ndarray(2x1)>) -> scores: list<score: float>
+ frame_to_world(position: numpy.ndarray(2x1)) -> pose: numpy.ndarray(4x4)
+ verify_pickup(position: numpy.ndarray(2x1)) -> pickedup: bool
+ write_qualities(frame: numpy.ndarray(NXM), qualities_positions: list<position: numpy.ndarray(2x1)>, qualities_positions: list<position: numpy.ndarray(2x1)>) -> drawn_frame: numpy.ndarray(NXM)
+ draw_bbs(frame: numpy.ndarray(NXM), bbs_positions: list<position: numpy.ndarray(2x1)>) -> drawn_frame: numpy.ndarray(NXM)
RobotModule
- home_position: numpy.ndarray(6x1)
+ pick_and_place(pick_pose: numpy.ndarray(4x4), place_pose: numpy.ndarray(4x4)) -> void
Behaviour
- pack_state: PackState
- gui: Gui
- vision_module: VisionModule
- robot_module: RobotModule
All the behaviour should have the same pack_state, gui, visionmodule and robotmodel object, which will be initialized in the in the tree and then passed as arguments in the construction of each behaviour. Not all the behaviour should have all the component, just the ones they are using
Packstate
- cells list<cell: Cell>
1
Relation
0..n
Cell
- model: str
- size: tuple<diameter: float, height: float>
- quality: float
- pose: numpy.ndarray(4x4)
- frame_pos: numpy.nadarray(2x1)
- sorted: bool
BehaviourTree
- pack_state: PackState
Gui
- showing_frame: numpy.ndarray(NXM)
- active_state: int
- states: list<GuiState>
- wait_interaction: bool
- chosen_model: str
- chosen_locations: list<position: numpy.ndarray(2x1)>
+ update(active_id: int) -> void
+ ask_for_help(query: str) -> void
GuiState
- elements: list<element>
+ connected_methods() -> void
Gui state is similar to behaviour, each gui state will have its unique combination of elements on the gui and methods