

Automated Mission Management of Small Unmanned Aircraft Systems for Critical Events in Urban Air Traffic

Robin Müller^a and Maximilian Bauer^b

Institute of Flight Systems and Automatic Control, Technical University of Darmstadt, Germany

Keywords: Safe Autonomous Systems, Contingency Management.

Abstract: Unmanned aerial systems (UAS) have a great potential to benefit society. This has already been shown in many use-cases. Nevertheless the true potential lies in the upscaling of operations. Therefore a high automation level and ensured safety is needed. A common approach to address safety in aviation is a risk analysis following by the design of procedures to mitigate the risks - so called contingency procedures. This paper presents a functional framework for automated mission management including contingencies for UAS. The framework is based on behavior trees and can be integrated with popular open source flight control software like PX4 and Ardupilot. Missions can be planned in a graphical interface using building blocks or in a Ground Control Station software like QGroundControl. The planning of contingency procedures is decoupled from the mission planning and allows for high modularity. Procedures can easily be added, modified or deleted, which is very important for certification of operation. The functionality of the framework is validated in various simulations, testing a plethora of contingencies and missions. Flight tests are currently conducted. The code needed to use the framework can be found on the website: <https://robin-mueller.github.io/auto-apms-guide/>.

1 INTRODUCTION

UAS already demonstrated to have great potential for a broad spectrum of applications. While most current applications focus on remote controlled or partly automated operations in visual line of sight (VLOS), the requirements of future commercial UAS applications will exceed current automatic capabilities (Federal Ministry of Transport and Digital Infrastructure, 2020, 9). To provide scalable and sustainable UAS-services, self-sufficient beyond visual line of sight (BVLOS) operations are required. Among other automated functions, automatically executable contingency and emergency procedures have to be developed and implemented to enable safe autonomous navigation.


The presented work builds upon existing mission management architectures as well as already defined contingency management requirements and employs open source tools and frameworks commonly used for developing automated systems.


To implement specific behaviors, this work adapts the model-based software design approach, which is

gaining popularity when designing complex systems (Pinquié et al., 2022). As a result, the developer is not required to be extensively experienced in programming, because behaviors are created using building blocks with well defined interfaces instead by re-designing low-level software code.

Within the scope of this work, the behavior tree paradigm is applied in conjunction with recent research on the topic of unmanned aerial operations and state-of-the-art software packages for robotics. Ultimately, this paper designs and implements a flight management framework that enables operators to efficiently plan, deploy and execute highly automated missions including contingency procedures. This is accomplished by

- elaborating a modular system architecture for automated contingency and mission management,
- designing a high-level behavior-based automatic control framework,
- providing a standard interface for managing the life cycle of real-time processes and
- adopting actively maintained open-source software packages widely used in robotics.

^a  <https://orcid.org/0009-0000-6775-389X>

^b  <https://orcid.org/0000-0001-6377-2276>

2 RELATED WORK

(Klößner, 2013, 59) and (Ögren, 2012) introduce mission management for UAV using behavior trees and extensively discussed the advantages while also showing and providing some missing formulation. Standardized formulations, a framework or the consideration of contingency procedures is still missing in literature.

Contingency management for UAS was discussed and a framework for the creation of contingency procedures was introduced by Eduardo et al. (Teomitzi and Schmidt, 2021, 2). This theoretical framework is the basis for the conceptualization of our technical and software framework.

(Usach et al., 2017) introduced many aspects to consider when building architectures for automated contingency management and influenced the design of the framework presented in this paper.

Consequently one can state, that the foundations for automated mission and contingency management were already laid, but a framework that integrates and leverages them to valuable application is still missing. The challenges that come along with it are addressed and solutions for them are presented in this paper.

3 DESIGN

To explain the design we will introduce the relevant aspects of contingency management and then the architecture and its components

3.1 Contingency Management

We adapt the definition of contingency management given by (Teomitzi and Schmidt, 2021). The authors clearly distinguish between contingencies and emergencies: Contingencies are defined as obstacles to the fulfillment of a system's high-level requirements and emergencies are considered as direct threats to the safety of the operation. Based on literature research and system analysis, they assemble an extensive catalog of mission jeopardizing events, mitigation strategies and recovery actions. With such actions, a contingency management system (CMS) is able to proactively safeguard the UAS during the entire operation. (Teomitzi and Schmidt, 2021)

With respect to this definition, figure 1 arranges the approaches to enhance system safety in a single model: The bow-tie model. Originally, this representation has been developed to support the assessment of the risks involved in the operation of an UAS (JARUS, 2017). It focuses on a single hazard, which

may occur due to several threats, which again may result in one or more consequences. The key to minimize the risk of operational safety impairment is to put certain barriers or controls in place. (Teomitzi and Schmidt, 2021)

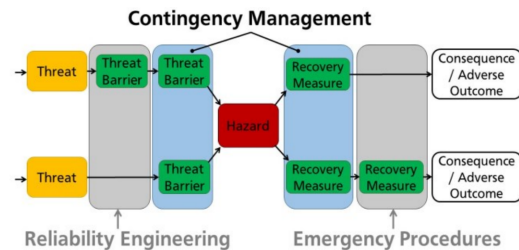


Figure 1: Contingency management within the bow-tie model. (Teomitzi and Schmidt, 2021, 2).

Additionally to contingency and emergency management, reliability engineering has been integrated into the model. Figure 1 shows that threat barriers are applied before the manifestation of a hazard, while recovery measures or controls take place afterwards. Reliability engineering and emergency management only introduce threat barriers respectively recovery measures, however, contingency management is able to do both. (Teomitzi and Schmidt, 2021)

A contingency procedure is a flight procedure designed to mitigate the inherent risk of a contingency state (Usach et al., 2017, 6). In this work, the term contingency recovery procedure (CRP) is advocated instead to emphasize that the developed CMS needs to detect a threat or hazard in order to trigger the procedure. Multiple guidelines for designing contingency procedures with respect to unmanned systems have been established. (Usach et al., 2017) The procedures are developed prior to operation and often need to be reviewed by an authority that regulates the operation. The structure of the procedures is very clear. When a threat or hazard is detected by a subsystem, a sequence of actions is triggered to prevent or mitigate potential negative outcomes. Detection and response can occur at various levels — such as components, software modules, internal, or external — but in most cases, a part of the system must be shut down or reconfigured. After or during this reaction, an automated UAS can either continue its mission, divert from the original trajectory and return after the contingency has been solved, divert immediately to a safe/alternative landing site and abort its mission or request (temporary) support from a human operator. The created framework can be used to implement the automated onboard execution of these contingency plans. Sub-system level reactions like shutting off or re-configuring components and software

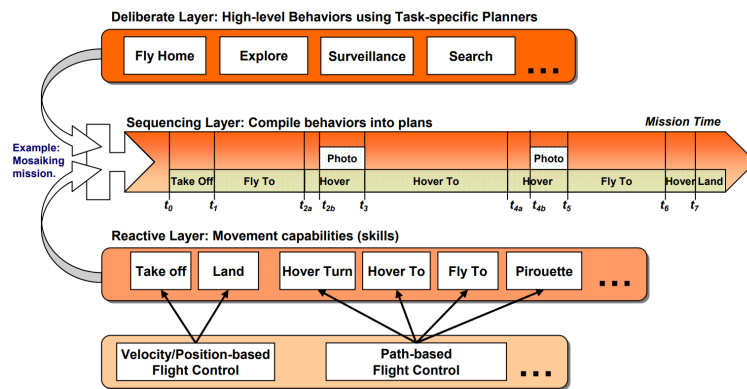


Figure 2: Three tier UAS mission planning architecture. (Adolf and Thielecke, 2007).

modules can flexibly be integrated while for the guidance of the UAS there are already building blocks that can be used. The operator is responsible for the specific implementation of behaviors, such as detection algorithms or re-planning methods.

3.2 Architecture

Since the beginning of the development of robotic systems, there has been three capabilities to consider when performing automated, robotic motion: Sense, plan and act. The relationship between and the implementation of each of those primitives are defined by system architectures which are developed individually for a specific field of application. (Ankit Srivastava, 2019, 1)

This work requires a high-level planning approach for UAS missions that allows to dynamically compile plans for contingency events during run-time. A common flexible and modular architecture that provides these capabilities is shown in figure 2. The three tier architecture (3TA) separates intelligent control into three layers with different abstraction levels.

The *Reactive Layer* comprises a set of elementary skills and controls the UAS with the lowest level of abstraction. In this context, reactive means that these skills are tightly coupled with the environment through sensor readings and actuators. There is no planning step when executing reactive skills, but only a initially defined goal that the skill aims to achieve. Therefore, they are considered as functions reacting to sensor readings and transferring them to actuator outputs according to the underlying implementation and the specified goal. (Adolf and Thielecke, 2007, 4)(Ankit Srivastava, 2019, 3)

The *Deliberate Layer* offers the highest level of abstraction for defining mission tasks. A component of this layer plans prospective movements using a task-specific planner that reasons about goals, re-

sources and timing constraints (Adolf and Thielecke, 2007, 4). As a result, individual skills defined by the reactive layer are compiled into a complex behavior plan to accomplish the given task. These plans alter the existing or create an entirely new mission (Adolf and Thielecke, 2007, 6).

Finally, the centralized *Sequencing Layer* exposes said plans for sequential execution and represents the currently pursued mission. This layer assembles a network of appropriate tasks handled by a sequencer that activates and deactivates respective skills. In general, concurrency between behaviors is not allowed, but certain behaviors may execute additional skills that are not related to movement but for example to the vehicle’s payload (e.g. taking photos with an on-board camera or releasing rope from a winch). (Adolf and Thielecke, 2007, 4)

This architecture presents a solution to mission management as a hybrid control problem and offers a flexible way of modularization. Thanks to the separate behavior-based reactive layer, plans can be generated automatically with algorithms designed for specifically assembling the available behavioral skills. Therefore, this approach is well suited to be utilized in the context of automated contingency management during UAS missions. (Adolf and Thielecke, 2007, 4)

3.3 System Components

A fundamental design principle chosen for the integration of automated contingency management tasks is to introduce modules with well defined responsibilities. This paper advocates the UAS architecture outlined by figure 3.

The figure adapts the architecture of Usach et al. (Usach et al., 2017), which introduces a CMS comprising two modules: The *Safety Monitor* and the *Contingency Manager*. The former evaluates the

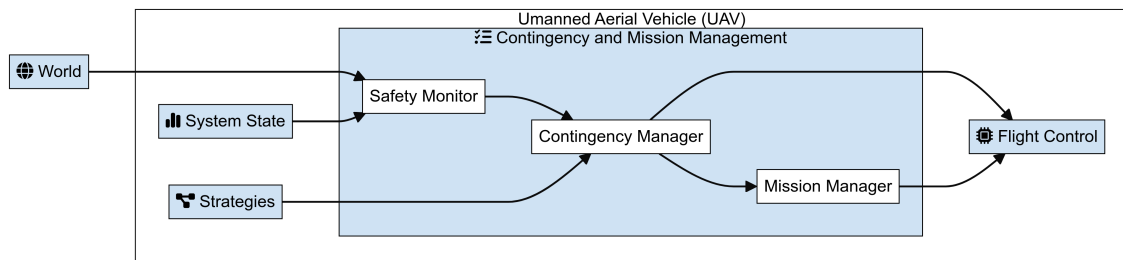


Figure 3: Functional components of an UAS with contingency management.

flight and system data that is available to the UAS and decides, whether an abnormal situation has occurred. In that case, it publishes this information to the contingency manager, which plans and executes a CRP according to a suitable contingency resolution strategy. Therefore, the safety monitor defines the current contingency state. Normally, the *Mission Manager* is responsible for executing the actions that the currently pursued mission dictates, but on the occurrence a critical event, the contingency manager is allowed to take over control. The mission manager will eventually be commanded to stop executing which leaves the contingency manager in charge of behavior execution. Both components use the same functionality to instruct the vehicle's movement. The flight controllers usually run on different, real-time capable hardware and its implementation heavily depends on the used flight stack. Behavior planning and execution solutions should be universally applicable to a wide range of UAS, so flight control is considered to be a standalone module by design.

With respect to the mission planning architecture from figure 2, one may implement the safety monitor and the contingency manager on the deliberate layer. This approach would require a single behavior planning instance to include safety monitoring and contingency handling tasks in the sequencing machine alongside nominal mission actions. Consequently, contingency and mission management concerns would not be well separated. Figure 3 depicts an alternative approach, where contingency monitoring and decision-making functions are implemented on top of the mission manager. Thus, procedures planned by the contingency manager have priority over the nominal mission. Whenever a contingency event is raised, the mission manager works in a slave mode and preempts the execution of the current procedure. (Usach et al., 2017)

In the following, the functional requirements of the components involved in behavior planning and execution are defined. Together, they achieve all contingency management tasks presented in figure 4.

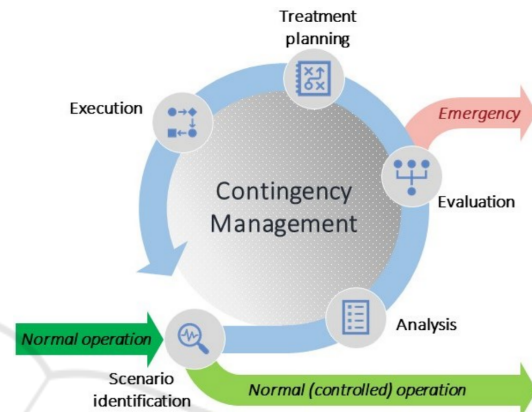


Figure 4: Tactical contingency management process cycle. (Teomitz and Schmidt, 2021).

3.3.1 Safety Monitor

The first process in the hierarchy of contingency management monitors the system state, gathers traffic data and assesses the current situation of the automated vehicle. It is responsible for maintaining operational safety. Therefore, the safety monitor is the first component that becomes aware of an abnormal event that affected the UAS thus it can be considered as the contingency detector. It decides whether a situation must be classified as operation critical or not. Furthermore, it is in charge of scenario prioritization if multiple critical events occur concurrently. (Usach et al., 2017)

The contingency manager observes the output of the safety monitor and acts as a multiplexor that launches the required contingency procedure. Consequently, the safety monitor is superior to the contingency manager. (Usach et al., 2017)

3.3.2 Contingency Manager

The contingency manager is providing the desired reaction after a contingency scenario has been identified. A behavior plan is created according to the current contingency state defined by the safety monitor. The operator has to explicitly connect a finite number of generally feasible behaviors with certain contingency states during design phase. If a critical

event occurs, the contingency manager automatically chooses a designated counteracting behavior from the associated, manually defined catalog and launches its execution. A behavior’s feasibility is dynamically evaluated based on the current system capabilities and characteristic execution requirements. (Usach et al., 2017)

3.3.3 Mission Manager

The currently pursued mission is executed by the mission manager, which is a standalone behavior executor. This component also incorporates the 3TA. The contingency manager may reuse mission manager’s deliberate or reactive behaviors and access its sequencing layer. If it has been decided that the occurred contingency shall be handled by altering the procedure executed by the mission manager, a new mission may be loaded. Hence, the contingency manager is able to override the nominal mission. During normal operation, the mission manager is the only component that requests flight control actions. (Usach et al., 2017)

4 IMPLEMENTATION

The elaborated CMS assumes that fundamental flight tasks are already covered by a basic flight stack and corresponding interfaces are provided and externally accessible. In other words, the system requires an independently acting control layer that manages essential low-level computations and introduces additional processes that increase the vehicle’s behavioral competencies. Moving forward, the widely used open-source autopilot software stack *PX4* is leveraged in this regard (Lorenz Meier et al., 2024). For modeling the involved decision-making processes, the behavior tree paradigm is to be applied. Furthermore, the popular software package *ROS2* (Macenski et al., 2022) is used as a middleware for the real-time system.

4.1 Multilayered Actuation

Effectively, both mission management and contingency handling are processes fully capable of displaying a behavior by controlling the automated system, so they are also referred to as “competences” (Toal et al., 1995, 2). However, the vehicle’s actuators must only be controlled by one instance at a time. In the field of robotics, this control problem is solved by incorporating the so-called subsumption architecture. Here, competences are represented by horizontally arranged layers. Higher-level layers can

subsume the roles of lower levels by suppressing their outputs at any given time. Lower levels therefore implement fundamental functionality, whereas higher levels add specialized competences to the control system. (Brooks, 1986)

Figure 5 applies this architecture to achieve safe automated UAS mission management. The lowest level competence implements functionalities to sequentially execute flight actions according to the mission plan. This level assumes an ideal operational environment and doesn’t account for critical events. Additional competences revolving around contingency management are added in higher levels. Specialized tasks, each of them responsible for handling a specific contingency, are implemented above mission execution. The higher the level of the contingency handler, the higher its priority. Therefore, this architecture queues the contingencies based on their priority which was determined prior operation.

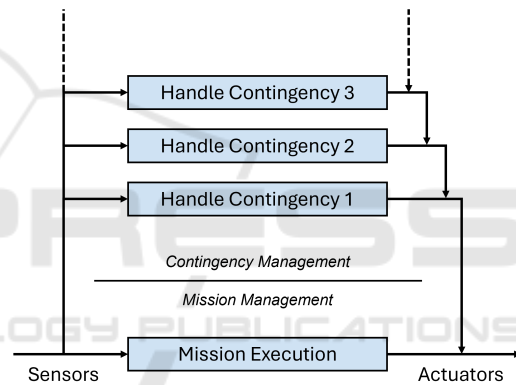


Figure 5: Control architecture for contingency handling. Based on the subsumption architecture by Brooks et al. (Brooks, 1986).

4.2 Behavior Modeling Using Trees

Ögren (Ögren, 2012) and Klöckner (Klöckner, 2013) already put behavior trees in context with UAS mission management. More specifically, Ögren advocates to employ BTs when performing ordered tasks or applying the subsumption control architecture. Consequently, it is sensible to adapt the tree modeling paradigm for behavior development in this research.

Furthermore, the behavior-based approach allows to separate behavior planning from vehicle control concerns, since corresponding tasks are designed to execute asynchronously according to the previously mentioned client-server model. However, there may also exist tasks with other purposes, which are not distributed by servers and execute synchronously. In behavior trees, the action node presents a model to what is referred to as a task here: An *Action* performs some

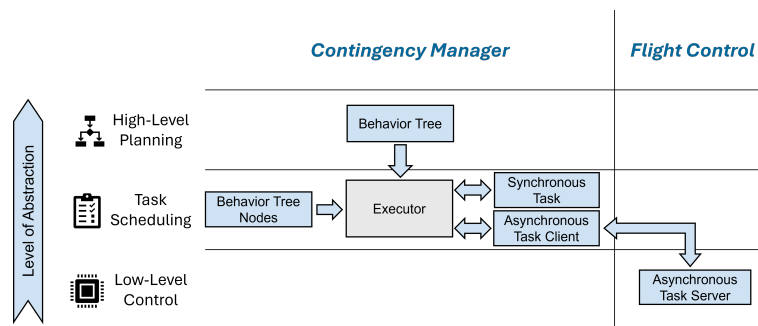


Figure 6: Levels of abstraction during behavior tree execution.

computation and returns *Success* if its objective has been accomplished, *Failure* if a problem occurred or *Running* if completion is under way (Ögren, 2012, 4). Therefore, a task may provide any kind of functionality and is not limited to for example executing flight actions. Klöckner defines a task as “a self-contained goal-directed behavior, which can be executed in order to achieve a given goal” (Klöckner, 2013, 58).

Figure 6 divides different fields that are instrumental to the development of behaviors into three levels of abstraction: High-level planning, task scheduling and low-level control.

At the highest level of abstraction, the behavior designer is concerned with appropriately organizing tasks to achieve specific missions. During high-level planning, the tasks themselves are exposed to the designer by their functional interfaces. A task may be configured by input parameters and pass information to the client during and after execution.

Task clients are created during task scheduling. At this level, the behavior is dictated by the implementation of the behavior tree nodes. Associated literature discusses the following fundamental node types: *Action*, *Condition*, *Sequence* and *Selector*. An *Action* executes a computational task according to a certain goal as described above. A *Condition* determines whether the current state corresponds to a certain statement. A *Sequence* executes all of its children in a certain order, whereas a *Selector* only does so until the first one succeeds. A big variety of behaviors can already be modeled with these four general types, but the capabilities of the tree may be extended by embedding custom logic in additional nodes. However, domain specific programming knowledge is required for this intend, which is why task scheduling provides a lower level of abstraction than high-level planning. (Ögren, 2012)

The lowest level of abstraction in the execution of the behavior tree is represented by low-level control algorithms that are available to the system as services. The algorithms read sensor data that is necessary for

accomplishing the goal of the corresponding task and control the UAS’s actuators accordingly. Hence, the sense and act steps during automatic control are accounted for by the instances within this level. There shall be no planning step at this level of abstraction, effectively meaning that the reactive control architecture applies for algorithms providing the control task (Ankit Srivastava, 2019). Instead, planning is executed at the highest level of abstraction by building a well structured behavior tree.

5 EVALUATION

The capabilities of the created framework are demonstrated in a simulated example mission. Execution of the mission is displayed in Figure 7.

5.1 Scenario Description

The mission comprises two waypoints an intermediate landing and a final landing and was implemented as a behavior tree. It is also possible to create a mission in planning software like e.g. QGroundControl and upload it to the mission manager. During execution of the mission, three different hazards will occur and predefined resolution strategies will be executed. The first threat is a loss of energy (LoE). A LoE occurs when the system is running out of energy faster than expected and successful mission completion is in danger. To detect this hazard, the energy source of the UAS must be monitored and associated with the energy cost of the remaining tasks. The implemented strategy “*Battery critical*” guides the UAS to the nearest safe point, when the hazard is detected. Since the nominal mission is to be resumed later, a temporary mission to the recharge point is executed independently from the mission manager. This mission is additionally designed to return to the position where the contingency has been detected, as soon as the energy level is restored. After the contingency is

resolved, vehicle control is given back to the mission manager.

The second and third hazards are a loss of landing location (LoLL). A LoLL occurs if a designated landing site is not clear for landing. Various resolution strategies are conceivable. If it is impossible for the UAS to land due to malfunctioning infrastructure or a nearby incident, it probably is most effective to skip this landing or determine an alternative landing location. In that case, the landing site is considered permanently blocked. If the landing must be performed at all costs or the landing site is just temporarily blocked due to quick maintenance work or other on-going processes, it's feasible to wait for clearance.

During “*Landing temporarily blocked*”, a so-called safe loiter maneuver is performed. This means that the vehicle commands its actuators so that it holds the current position. This character of the maneuver depends on the type of the UAS. A multicopter is able to stop and hover immediately, while a fixed-wing must circle around a given position.

If “*Landing permanently blocked*” applies, a new location is to be approached for landing. Such a location may be predefined during the strategic phase of operation or it is automatically determined just in time when a LoLL contingency is detected. Afterwards, a corresponding route is calculated and the nominal mission is updated. Following the update, the current mission segment has been expanded to include a number of detour tasks to the now targeted alternative landing site. In this procedure, the contingency manager does not make use of flight control. Instead, it copes with LoLL by taking responsibility of planning and setting up the mission manager, which is left in charge of executing the diversion within a customized nominal mission segment.

Each contingency scenario may arise at any point in time and the original mission is not to be continued until the abnormal situation is resolved. This poses a complex challenge and requires the implemented CMS to dynamically plan and safely execute the desired behavior regardless the drone’s position within the designated flight zone or the progress of the mission. If multiple scenarios apply, it is required to prioritize a procedure and dynamically react to changes of the contingency state. To achieve that behavior, specific priorities are assigned to each of the incorporated contingency states regarding scenario identification and procedure execution. The prioritization of the considered contingencies are summarized in table 1.

Table 1: Prioritization of the contingencies applied during simulation.

Contingency name	Priority level
Battery critical	High
Landing permanently blocked	Medium
Landing temporarily blocked	Low

5.2 Simulation

The first time, the drone detects that the landing site at “Stop1” is temporarily blocked when this waypoint is reached and holds the position. Realistically, the battery level decreases in that period and it is simulated that a critical state is reached eventually. Figure 7 shows that the UAS diverts to the recharge point as intended. On the way back, the landing site is

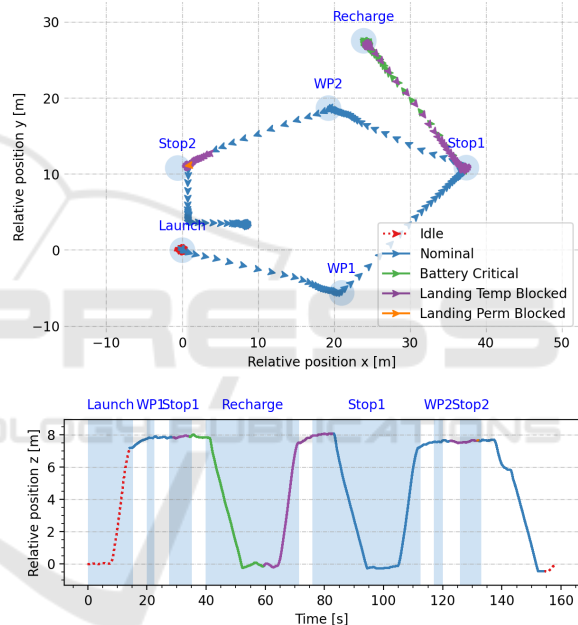


Figure 7: Flight operations involving multiple contingencies.

again recognized to be temporarily blocked and the battery contingency handler guides the vehicle back to the original position where it will continue to wait for clearance. The other situation occurs around waypoint “Stop2”. In the process of approaching the second landing location, its status changes to be temporarily unavailable, so the drone stops one more time and waits for further action. However, this time the status of the landing site changes to be permanently blocked in the mean time which requires the system to land at an alternate location. The mission manager is updated accordingly and the flight continues targeting a new destination. Finally, the vehicle lands at the dynamically calculated coordinates and the operation completes successfully.

6 CONCLUSION

This work contributes to enabling certifiable and therefore economically viable applications for the European drone sector. The software architecture of the de facto autopilot standard, the open-source PX4 flight stack, is extended to incorporate contingency management for UAS. Essentially, two additional abstraction levels which are based on the behavior tree paradigm are introduced to the existing automatic control pipeline: High-level planning and task scheduling. They are build on top of the default functions for controlling the vehicle's movement provided by PX4 and utilize them to offer a convenient interface for designing flight behaviors.

The simulation results validate the functionality of the implemented features and prove the applicability of the created framework. The software of the framework is available online (<https://robinmueller.github.io/auto-apms-guide/>) and can be used in simulation or on a real drone. Flight tests are currently conducted and the results will be published on the web page.

It is envisaged that the contingency manager is supplied with a catalog of various behaviors that are specifically intended for the purpose defined by a particular resolution strategy. Instead of selecting feasible contingency procedures deterministically, further research could address the development of algorithms that dynamically evaluate the success probability of all behaviors from the catalog and optimize the safety maneuver intelligently while still being certifiable by an authority (Colledanchise et al., 2014).

ACKNOWLEDGEMENTS

This work has been funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY center [LOEWE/1/12/519/03/05.001(0016)/72].

REFERENCES

- Adolf, F. and Thielecke, F. (2007). A sequence control system for onboard mission management of an unmanned helicopter. In *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, Reston, Virginia. American Institute of Aeronautics and Astronautics.
- Ankit Srivastava (2019). Sense-plan-act in robotic applications.
- Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23.
- Colledanchise, M., Marzinotto, A., and Ogren, P. (2014). Performance analysis of stochastic behavior trees. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3265–3272. IEEE.
- Federal Ministry of Transport and Digital Infrastructure (2020). Unmanned aircraft systems and innovative aviation strategies: The federal government's action plan. PDF file.
- JARUS (2017). Jarus guidelines on specific operations risk assessment (sora).
- Klößner, A. (2013). Behavior trees for uav mission management. In Horbach, M., editor, *Informatik 2013 - Informatik angepasst an Mensch, Organisation und Umwelt*, GI Edition Proceedings, pages 57–68. Gesellschaft für Informatik, Bonn.
- Lorenz Meier, Daniel Agar, Beat Küng, Julian Oes, Thomas Gubler, Matthias Grob, Paul Riseborough, Roman Bapst, Anton Babushkin, David Sidrane, Mathieu Bresciani, px4dev, Silvan Fuhrer, Mark Charlebois, James Goppert, Nuno Marques, Andreas Daniel Antener, Dennis Mannhart, PX4 Build Bot, kritz, Mark Whitehorn, Kabir Mohammed, Jaeyoung Lim, Simon Wilks, Mark Sauder, Peter van der Perk, Pavel Kirienko, Sander Smeets, Martina Rivizzigno, and Hamish Willee (2024). Px4/px4-autopilot: v1.15.0 beta 1.
- Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics*, 7(66):eabm6074.
- Ögren, P. (2012). Increasing modularity of uav control systems using computer game behavior trees. In *AIAA Guidance, Navigation, and Control Conference*, Reston, Virginia. American Institute of Aeronautics and Astronautics.
- Pinquió, R., Romero, V., and Noel, F. (2022). Survey of model-based design reviews: Practices & challenges? *Proceedings of the Design Society*, 2:1945–1954.
- Teomitzi, H. E. and Schmidt, J. R. (2021). Concept and requirements for an integrated contingency management framework in uas missions. In *2021 IEEE Aerospace Conference (50100)*, pages 1–17, Big Sky, MT, USA. IEEE.
- Toal, D., Flanagan, C., Jones, C., and Strunz, B. (1995). Subsumption architecture for the control of robots. *Proceedings Polymodel-16*.
- Usach, H., Torens, C., Adolf, F., and Vila, J. (2017). Architectural considerations towards automated contingency management for unmanned aircraft. In *AIAA Information Systems-AIAA Infotech @ Aerospace*, Reston, Virginia. American Institute of Aeronautics and Astronautics.