View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

154
Countries delivered to

Our authors are among the
TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Onboard Mission Management for a VTOL UAV Using Sequence and Supervisory Control

Florian Adolf and Franz Andert
Institute of Flight Systems, Unmanned Aircraft Dept.,
German Aerospace Center (DLR)
Germany

1. Introduction

This chapter addresses the challenges of onboard mission management for small, low flying unmanned aerial vehicles (UAVs) in order to reduce their dependency on reliable remote control. The system presented and tested onboard an UAV provides different levels of autonomy, switchable at runtime either manually by the operator or automatically due to absence of a data link. This way, it is a feasible approach towards autonomous flight guidance within the low-altitude domain (e.g. urban areas) where unpredictable events are likely to require onboard decision-making.
In the following sections the problems of onboard mission management, embedded high level architectures and their implementation issues are discussed. The design of a onboard Mission Management System for a test platform with vertical take-off and landing (VTOL) capabilities is presented, followed by discussions of the implemented system and a research outlook.

2. Autonomy Management Problem

For many UAVs, an operator at a remote control station performs joystick control and plans the mission. The operator often commands the UAV using joystick remote control (e.g. rate or velocity commands) or sets a target location for a position command. With an onboard world model and path planning capabilities, more autonomy is on board the system such that an operator might issue higher-level commands, e.g. directing the vehicle to search a collision free path automatically and fly back to base.
This implies different abstraction levels within the onboard system such that each level of system autonomy is clearly represented. The level of autonomy at which an operator commands the UAV might vary during a mission. For example, while the UAV performs waypoint navigation the operator interrupts the flight in order to manually direct the UAV towards an object of interest that just appeared in a live video feed from the onboard camera.
The design and implementation of different levels of control necessitates provisions for operational safety and certain user requirements. In particular, the operator must remain in

the loop at all levels of autonomy whenever the data link is available. Also, the operational environment is characterized by events that can occur in an unknown order and at sporadic time instances. It must implement input checks for syntactical plausibility and even semantic correctness, wherever possible.
Beside this autonomy management problem, the organization and abstraction of the system into a suitable architecture is a challenge. Thus, in the next section existing architectural concepts are discussed.

3. Related High-Level Control Architectures

More self-reliance and decision-making autonomy poses questions regarding a suitable architecture according to which the management system is designed.
Knowledge-based systems establishing the concept of a cognitive process as decision-making entity were presented in the UAV domain (Hill 1997, Putzer 2003). Concepts exist that are based on the behavior-based paradigm (Weiss 2005), where a set of elementary behaviors (so-called skills, such as movement primitives) is combined in such a way that a new emergent behavior is created. Furthermore, layered architectures (Freed 2005) have been proposed that comprise distinct system modes also known as hybrid control (Egerstedt 1999).
Using knowledge-based systems, classical artificial intelligence spent over five decades trying to model human-like intelligence. Inspired by these systems, several research projects seek to produce a human-like thinking process (also known as cognition) in order to achieve high-level control in decision-making systems (Hill 1997, Putzer 2003). A commonly used cognitive architecture is implemented in SOAR (Laird 1987). Since real-time properties are one crucial design aspect for a UAV decision-making system, a real-time derivate of SOAR, Hero-SOAR exists.
However, there are major implementation issues related to cognitive production systems (Musliner 1995). First, "chunking", a pattern matching technique, might be hard to confine with respect to execution time and memory usage. Second, real-time reflexive actions (a direct connection of a sensor to an actuator) invoke a high-variance of unpredictable system events. Furthermore, problems were experienced when trying to effectively coordinate and mediate reflexive behaviors with the overall deliberative behavior of the system. If the reflexive actions can bypass the normal deliberation mechanisms, it may be difficult or impossible for the deliberation processing to reason about and affect the real-time reaction.
Hence, the architecture for any UAV decision making system should particularly focus on "embedding real-time in artificial intelligence" rather than "embedding artificial intelligence in real-time” (Musliner 1995). Moreover, a principle shortcoming of the cognitive approach is the emphasis on representation at a high, symbolic level. This yields to control strategies that may make conceptual sense to a human designer but the intelligence in such systems belongs to the designer. Additionally, it is questionable whether humans deploy a complex thinking process for every intended behavior rather than think in a more reactive way (Agre 1995).
These disadvantages are addressed by the behavior-based control with the Subsumption Architecture (Toal 1996, Brooks 1990), which does not necessarily seek to produce cognition. It rather uses a hierarchy of fast reactive loops where each loop is capable of executing a distinct behavior. Moreover, higher reactive loops modify the behavior of lower ones. The

concept of arbitration allows to automatically select among behaviors, and the so-called action-oriented perception frames the perceptual input according to the task. Some approaches interconnect elementary behaviors and superposition them, which results in a new, emergent system behavior. The ultimate goal in many behavior-based approaches is to enable robot-learning techniques such that a system can automatically deduce which behaviors must be compiled together in order to achieve a goal. Admittedly, one of the side effects is that they produce complex system topologies if behaviors are interconnected. It then is almost impossible to explain the system behavior. Moreover it is hard to achieve a notion of optimality (Pirjanian 1999).
UAVs are supposed to be semi-autonomous, remotely guided, assistant systems rather than anthropoid, autonomous systems. One of the key requirements of having several levels of system autonomy cannot be achieved with solely deliberate nor reactive architectures. Deliberate architectures relate "autonomy" to human-like intelligence and rational acting, whereas reactive architectures consider it as a system's "ability to act independently in the real world environment" (Makowski 2004).
Thus, it is desireble to combine advantages of knowledge-based and behavior-based architectures. Current robotic development created architectures combining both ideas into one system. Inspired by the Subsumption Architecture and empirical observations, the 3T architecture (Bonasso 1997) separates intelligent control into three interacting layers (or tiers). The first layer comprises a set of so-called reactive skills. These behaviors represent control laws tightly coupled with the environment through sensor readings and actuators. Skills make so-called simple-world assumptions such as, the sensor input is always valid and the desired goal can be achieved.
In order to accomplish a specific task, the sequencer on the second layer assembles an appropriate task network of skills by activating and deactivating respective skills. When more than one skill is active, they form a so-called task network.
The third layer is the deliberative layer, which comprises a planner that reasons about goals, resources and timing constraints with well-known rational techniques.

4. Mission Management System

In the following the overall system design is presented with respect to particular design decisions. The effective architecture of the onboard mission management system is based on ideas discussed in the previous section and yields two main system components: The Sequence Control System and the Supervisory Control System.

4.1 Design Decisions

The major requirements with respect to real-time execution, predictable system behavior and the need for different levels of autonomy at runtime yield the following principal design decisions:

- The embedded system architecture must be separated into interacting layers, to enable the implementation of deliberate and reactive approaches. This leaves room for a behavior-based reactive layer and allows several kinds of artificial intelligence techniques in the deliberative layer(s).

- The layered architecture chosen for this hybrid control problem is the 3T architecture. It offers a flexible way of modularization, centralizes the execution of actions and does not rely on interacting skills.
- The behavior-based paradigm, as a bottom-up strategy for intelligent systems, is worth being considered for the reactive layer, since it enables real-time execution and relatively simple behavior development. This paradigm allows a way to compile elementary problem solutions (e.g. moving to a position) into a library of behaviors.
- Known shortcomings of the behavior-based approach with respect to online learning, behavior interaction and arbitration techniques, are eliminated intentionally.
- When behavior interaction and abstract behaviors are not available in the reactive layer, the discussed disadvantages of the 3T approach can be neglected.

As a result, this design concept for onboard mission management combines the 3T architecture with ideas from the behavior-based paradigm.
In the following, the overall system is described from three points of view. The illustrations in Figure 1 outline how the principles of 3T's high level control decomposition are represented in the system. In order to highlight a system wide context, Figure 2 describes the component organization from an implementation point of view.

4.2 High-Level Control Architecture

The high-level control architecture onboard the UAV is based on the 3T architecture for a hierarchical decomposition of system autonomy (Figure 1). Moreover, the behavior-based paradigm yields distinct behaviors that can be combined sequentially across each layer. The behaviors are either of a basic movement primitive type (e.g. flying a linear trajectory) or of deliberate nature (e.g. searching and tracking an object on ground).

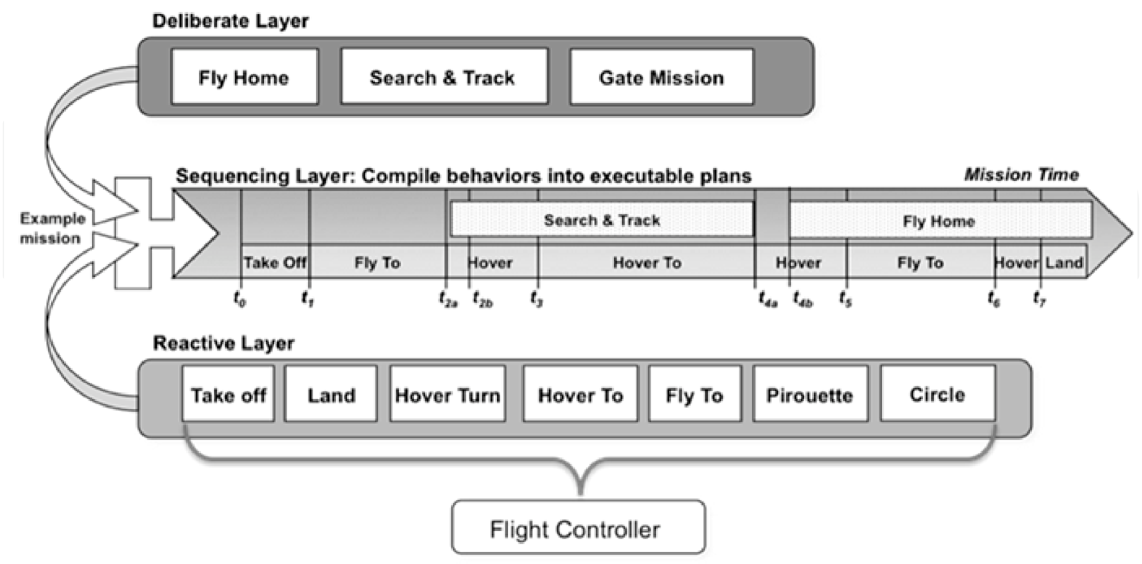


Fig. 1. Onboard high-level control based on the 3T architecture

Two basic prerequisites of the proposed mission management architecture in Figure 1 are implemented by two systems, sequentially executed at each instant of time. The first system implements deliberate behaviors and a set of operational safety features.
The second component generates flight control commands at every instant of time. It contains a library of basic movement behaviors from the reactive layer. Figure 1 illustrates which behaviors are located at the skill layer. These behaviors generate instantaneous trajectory-based control commands that are fed into the flight controller.
The deliberate layer shows examples of complex behaviors. These alter existing missions or create new missions. In this context, mission planning will output the list of sequential behavior commands shown in Figure 1.

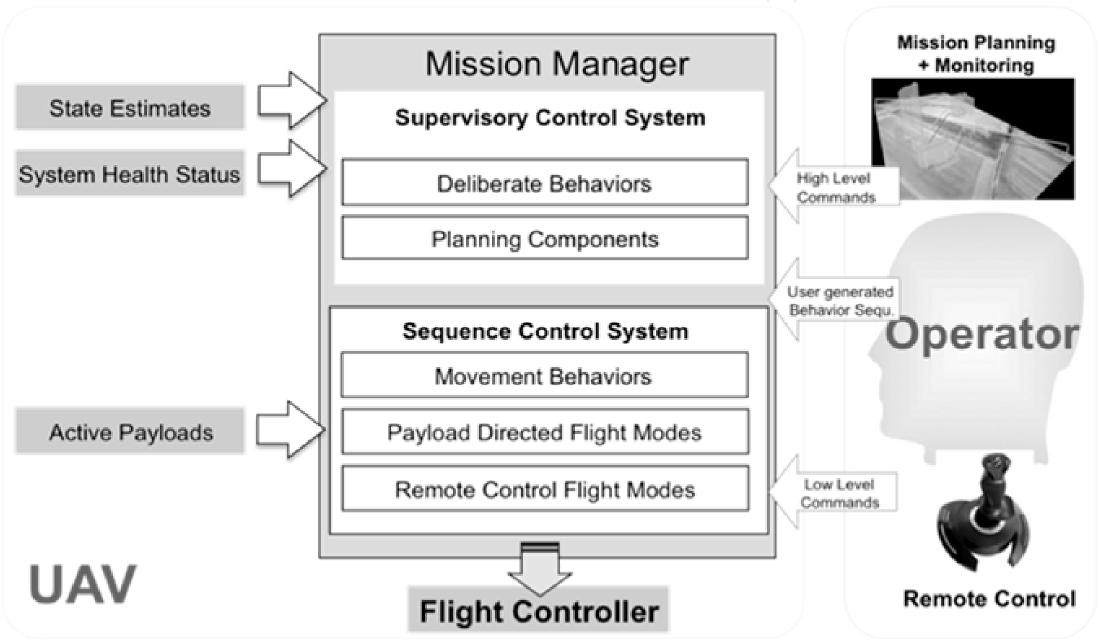


Fig. 2. The Mission Manager allows different levels of autonomy and comprises a supervisor and sequence controller implementing the 3T architecture

The set of basic behaviors from the reactive layer need to be represented as a system. It needs to coordinate and execute the reactive layer's basic movement behaviors that interface with the flight control system. Furthermore, a sequence of such behaviors needs to be executed automatically while handling unforeseen events like a sudden interruption from the remote operator. This is done in the Sequence Control System as it implements the executive component of the sequencing layer of the 3T architecture.
Since neither the deliberative layer nor the skills alone can handle all situations optimally, the Supervisory and Sequence Control System provide additional glue logic to store procedural knowledge that neither belongs clearly to the deliberative layer nor to the skill layer. For example, during flight testing a safety pilot may need to switch between manual or computer control, and thus the system must stop producing actuator commands and set its onboard components into a defined stand-by state.

4.3 The Sequence Control System

The Sequence Control System is exposed to a number of potentially concurrent events. Hence, the specification and implementation of the system is modelled as an event-based system. The majority of event-based systems are modelled using the Unified Mark-up Language (UML), an industry-wide standard notation. It supports the object oriented design pattern and provides dynamic modelling techniques such as state charts, sequence diagrams and activity diagram. State charts have been extensively studied such that abstract testing techniques allow verifying a model (semi-) automatically. Also, there exists good software tool support, which eases the development process significantly. Moreover, there are tools that provide code generators such that the implemented code is directly derived from the state chart-based specification. Otherwise, it is likely that specification and implementation begin to diverge over time.
Thus, the Sequence Control System is modelled as UML 1.2 State Charts. Basically, State Chart diagrams (also known as State Machines in UML 2) are finite automatons with a finite set of states where exactly one state is active at a time. They depict the dynamic behavior based on its response to events, showing how the model reacts to various events depending on its current state. Events can trigger a transition into another state, where so-called guards are the condition that must become true in order to traverse along the transition. The guards on similar transitions leaving a state must be consistent (deterministic) with one another.

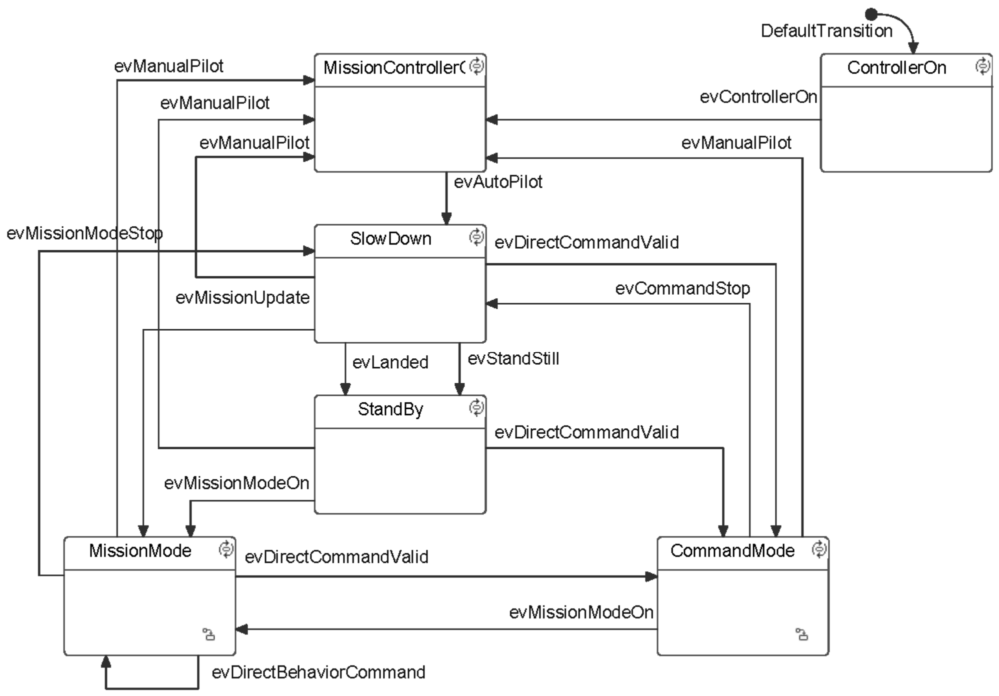


Fig. 3. The UML State Chart Model specifying the top level of the Sequence Control System

The UML model of the Sequence Control System is shown in Figure 3. It has two hierarchical levels where the top level models the procedural flow for a safe operation. The two composite states, ″Mission Mode″ and ″Command Mode″, model mission plan processing and direct command execution respectively. Every state of the top level has a transition to the ″Mission Controller Off″ to handle a manual control event, such that the Sequence Control System stands by in an idle state. If the system is in computer-based flight

mode (auto mode), another idle state ″Stand By″ lets the UAV hover at its current position when the state was entered; including a position on the ground. The state ″Slow Down″ is necessary to assure a smooth changeover into ″Stand By″ regardless of the flight maneuver being executed. In case the operator commands the UAV to stop, a transition from every auto mode state assures that the command is executed. Among events certain priorities exist. For example, an event switching to manual mode is more important than a stop command and requires processing. The order in which the event check is performed accounts for this obligation.
The state mission mode contains the actual library of behaviors. There are no transitions among behaviors assuring that only one can be active at a time. This is required in order to overcome emergent system behavior caused by overlapping and interactions. For each behavior there exists a termination condition, which transits into the command parser ″Parse Command″. Basically this state grabs behavior commands from an existing mission plan (Figure 4). It issues an event for traversing into the appropriate state. When the mission plan is processed, the ″Mission Mode″ composite state is exited. For payload directed flight, the composite state ″Command Mode″ can be entered from every state inside "Mission Mode″.

4.4 The Supervisory Control System

The Supervisory Control System is responsible for taking high level decisions based on internal and external events. It is responsible for managing requests from the UAV operator (e.g. allow to load a mission or executing a complex command), as well as reacting to a loss of the data link. It is executed before the Sequence Control System at every instant of time. This allows the Supervisory Control System to modify a mission when conditions are recognized to imply a necessity of modification. Moreover, as long as there is no mission update incoming from the operator, it can act as a substitute commanding entity for the operator. It can command the Sequence Control System via the same type of commands that a remote operator can send to the Sequence Control System (e.g. start and stop the execution of a mission currently loaded). It is the entity in charge of managing the execution of deliberate behaviors. Therefore, the Supervisor retains planning capabilities, and recognizes associated high-level mission objectives (e.g. "Fly Home").

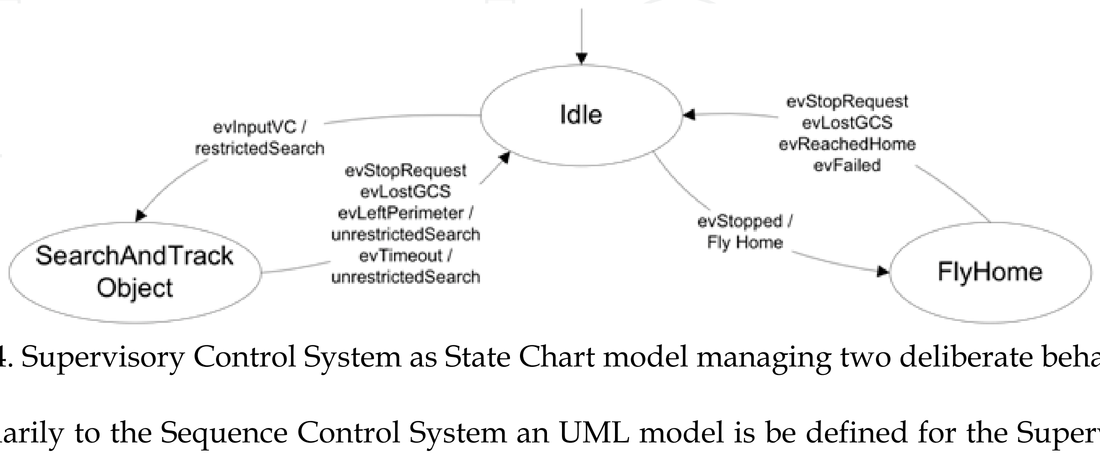
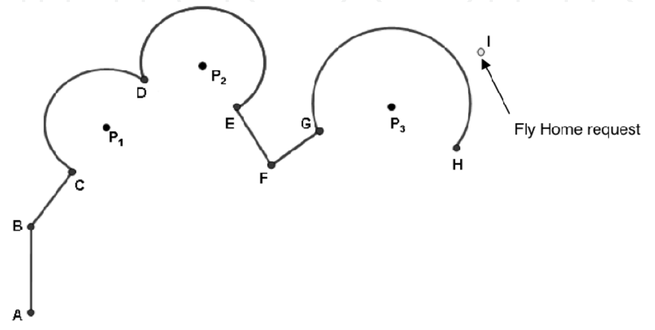

Idle
evInputVC / restrictedSearch
evStopRequest evLostGCS evReachedHome evFailed
evStopRequest evLostGCS evLeftPerimeter / unrestrictedSearch evTimeout / unrestrictedSearch
evStopped / Fly Home
SearchAndTrack Object
FlyHome

Fig. 4. Supervisory Control System as State Chart model managing two deliberate behaviors

Similarly to the Sequence Control System an UML model is be defined for the Supervisory Control System shown in Figure 4. In this example, it implements two high-level behaviors, "fly home" and "search and track object". The first lets the vehicle find a way to fly back to a start position, whereas the second lets the vehicle search and track an objects moving on the ground. The Fly Home behavior provides the vehicle with the capability of returning autonomously to the starting point of a given mission. It implements the replanning process shown in Figure 5 which is based on the properties of each basic movement primitives of the reactive layer. The Search and Track behavior can be used to find and track a moving object on the ground (Figure 6). Once spotted (e.g. using payload directed object detection) it is desirable to track it. Similar to the Fly Home behavior, it seizes information in the a-priori mission plan to implement tactical means, avoiding suboptimal search execution and tracking performance. Further details on the internal ongoings of these behaviors are presented in (Adolf 2009).



Origianl plan	Invert sequence order	Filter tasks	Recompute behavior parameters	Drop behaviors in Proximity
Take Off	Circle XY (P_3, θ_{P3})	Cirlce XY (P_3, θ_{P3})	Cirlce XY (P_3, - θ)	Cirlce XY (P_3, - θ)
Hover To (A, θ_A)	Hover To (G, θ_{G2})	Hover To (G, θ_{G2})	Hover To (G, $\theta_{G,F}$)	Hover To (G, $\theta_{G,F}$)
Pirouette Flight (B)	Hover To (G, θ_G)	Hover To (G, θ_G)	Hover To (F, $\theta_{G,F}$)	Hover To (F, $\theta_{G,F}$)
Waiting For T	Hover Turn (φ)	x Hover Turn (φ)	Hover To (F, $\theta_{F,E}$)	Hover To (F, $\theta_{F,E}$)
Hover To (C, θ_C)	Hover To (F, θ_F)	Hover To (F, θ_F)	Hover To (E, $\theta_{F,E}$)	Hover To (E, $\theta_{F,E}$)
Hover To (C, θ_{C2})	Circle XY (P_2, θ_{P2})	Cirlce XY (P_2, θ_{P2})	Hover To (E, $\theta_{2,E}$)	Hover To (E, $\theta_{2,E}$)
Circle XY (P_1, θ_{P1})	Circle XY (P_1, θ_{P1})	Circle XY (P_1, θ_{P1})	Circle XY (P2, - θ_{P2})	Cirlce XY (P2, - θ_{P2})
Cirlce XY (P_2, θ_{P2})	Hover To (C, θ_{C2})	Hover To (C, θ_{C2})	Hover To (D, $\theta_{1,D}$)	Hover To (D, $\theta_{1,D}$)
Hover To (F, θ_F)	Hover To (C, θ_C)	Hover To (C, θ_C)	Circle XY (P1, - θ_{P1})	Cirlce XY (P1, - θ_{P1})
Hover Turn (φ)	Waiting For T	x Waiting For T	Hover To (C, θ_{C2})	Hover To (C, $\theta_{C,B}$)
Hover To (G, θ_G)	Pirouette Flight (B)	Pirouette Flight (B)	Hover To (C, θ_C)	Hover To (B, $\theta_{C,B}$)
Hover To (G, θ_{G2})	Hover To (A, θ_A)	Hover To (A, θ_A)	Pirouette Flight (B)	Hover To (B, $\theta_{B,A}$)
Circle XY (P_3, θ_{P3})	Take Off	x Take Off	Hover To (A, θ_A)	Hover To (A, θ_A)

Fig. 5. Planning process of the Fly Home deliberate behavior based on the basic behaviors

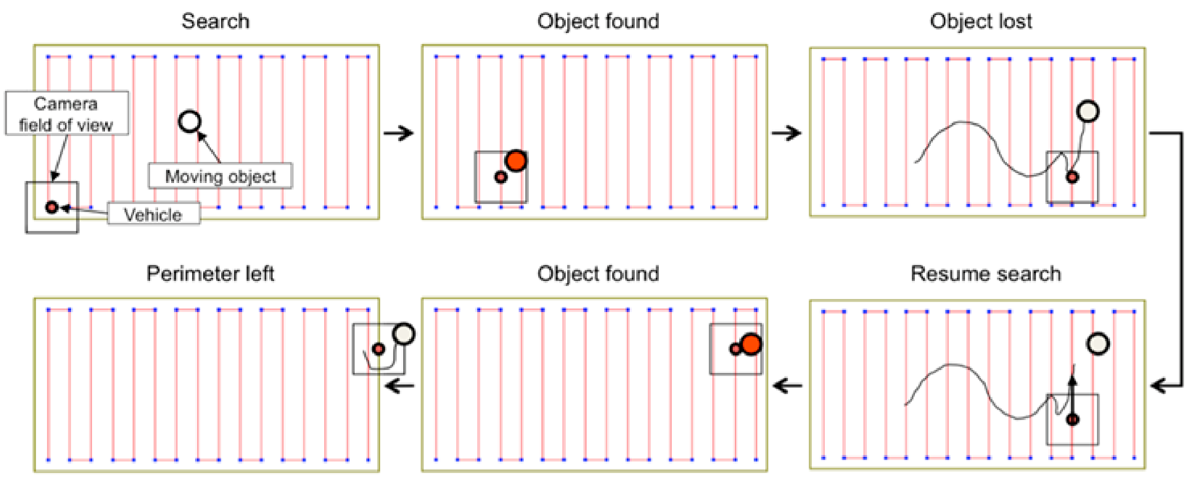


Fig. 6. The Search and Track deliberate behavior shown as it finds and track a ground object with a defined search area

4.5 Abstract System Testing

The overall complexity of the Supervisory and Sequence Control System imposes a thorough test strategy. Abstract tests relate to Model-based testing and as such assure that the model is free of principle design mistakes.

In the modelling stage a set of errors can occur. Some relate to potentially isolated or unreachable states, as well as, missing or erroneous triggers and guards. An even more fundamental problem is the theoretically infinite set of sequences that have to be tested in an abstract manner. That is, regardless of the meaning and function behind events, states and transitions, the tests must traverse through the model even if a certain test would make no sense from a practical point of view.

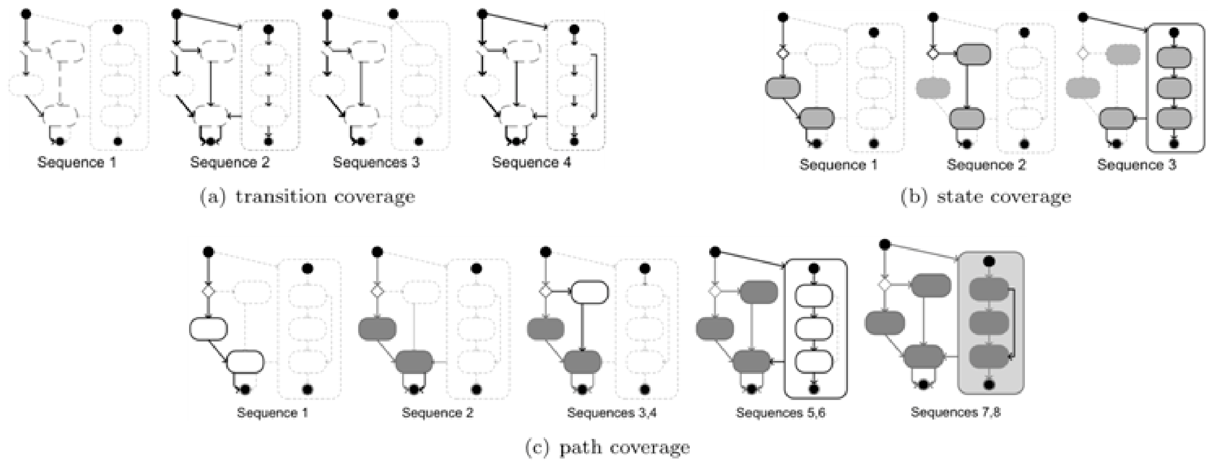


Fig. 7. Abstract testing strategies applied to the State Chart models

UML state charts represent a constructive model such that tests can make use of its internal structure, also known as white-box testing. These abstract tests can be divided into three groups, namely the path coverage, transition coverage and state coverage tests as highlighted in Figure 7. For each of these groups, a set of event chains is generated and executed. Once a full coverage of all possible combinations is reached, the tests are completed. However, there is no defined final state and states can have loops such that infinitely long test chains for path coverage tests occur. Therefore, a relaxation for the path coverage criteria is implemented such that loops must be passed exactly once. This is a strong and feasible criterion (Rumpe 2005). The system with greatest deepness is the Sequence Control System where at maximum six events yield from the start state to the deepest level. This implies 34^6 test sequences for the Sequence Control System.

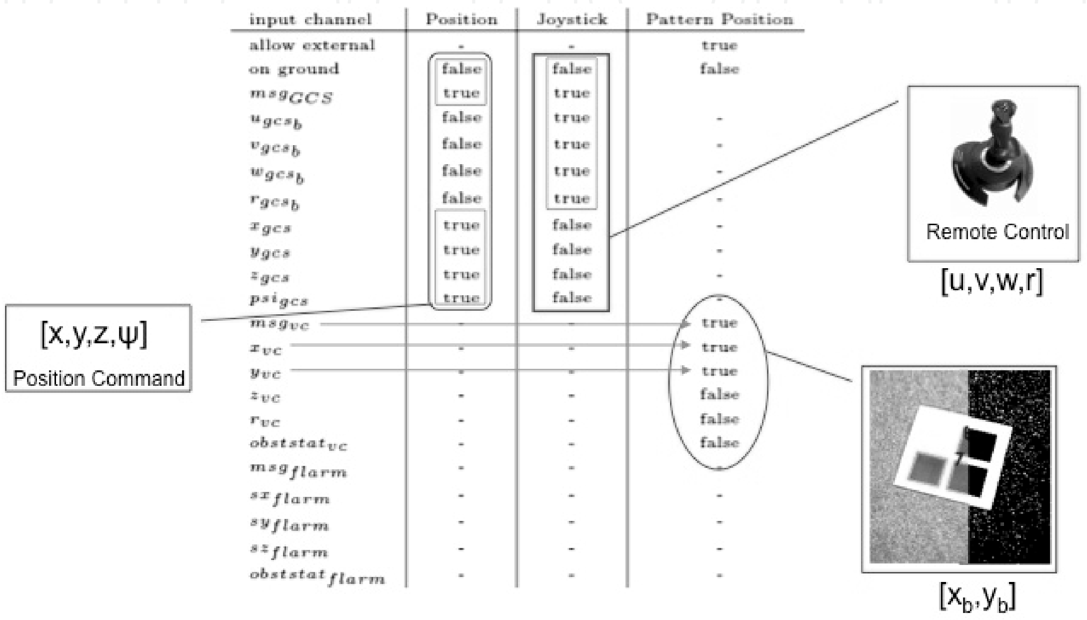
Moreover, the McCabe metric (McCabe 1976) is used to finally estmate the cyclomatic complexity:

$$V(G)=edges+nodes+2 \tag{1}$$

The State Chart with most transitions and states is the Sequence Control System. The system has 43 edges and 23 nodes, thus a McCabe complexity of 22. This is commonly considered as a complex system but still with reasonable risk. A V(G)-value of more than 50 is considered a system not testable. This would be the case, if for example the system wasn't decomposed into a Supervisory and a Sequence Control System. According to this metric, the more deliberate behaviors the Supervisor contains, the more imminent becomes a further decomposition the Supervisory Control System. Since the deliberate behaviors are modelled using individual State Charts this problem is practically circumvented.

4.6 Plausibility Checks at Runtime

It is a relatively complex decision to determine whether payload directed flight should be permitted or not. First, compared to behaviors in ″Mission Mode″, every direct command may not have a termination condition. For example, one instantaneous velocity command from the operator cannot reach a timely bounded target state. For such cases, a quasi infinite behavior is bounded in execution time, such as the time passed since a velocity command was last received. Second, it depends on which data is coming on what input channels. These types of problems are addressed using a static truth table (Figure 8).



input channel	Position	Joystick	Pattern Position
allow external	-	-	true
on ground	false	false	false
msg_{GCS}	true	true	
u_{gcs_b}	false	true	-
v_{gcs_b}	false	true	-
w_{gcs_b}	false	true	-
r_{gcs_b}	false	true	-
x_{gcs}	true	false	-
y_{gcs}	true	false	-
z_{gcs}	true	false	-
psi_{gcs}	true	false	-
msg_{vc}	-	-	true
x_{vc}	-	-	true
y_{vc}	-	-	true
z_{vc}	-	-	false
r_{vc}	-	-	false
$obststat_{vc}$	-	-	false
msg_{flarm}	-	-	
sx_{flarm}	-	-	-
sy_{flarm}	-	-	-
sz_{flarm}	-	-	-
$obststat_{flarm}$	-	-	-

Fig. 8. Truth table assessing priority plausibility of payload or operator low-level commands

It checks valid combinations for payload directed flight and manual interruption of missions by the operator (e.g. joystick or position commands). The table is conflict-free, prioritses all combinations of relevant signals and allows plausibility checks of every incoming signal.
Likewise the command checks for low-level commands, missions defined as sequence of behavior commands need to be considered as one single complex command. Although it is hard to reason about the "sense" of a mission plan, it is possible to implement a plausibility check using a language grammar. This way, a behavior sequence is treated as a programming language that satisfies type 2 of the Chomsky hierarchy of languages. It can be expressed via a context-free language and thus it can be defined in an Extended Backus-Naur Form (EBNF, ISO-14977 2001). Furthermore, using attribute features (e.g. semantic checks of a height parameter), the attributed EBNF shown in Figure 8 has been implemented which fulfills the following general requirements:

- There's a need for a unique identifier for each mission by which the ground control station can recognize that a mission plan has been loaded by the onboard system successfully.
- Optionally, each mission can contain a coordinate transform header in order to transform certain behavior parameters from the ground control station's reference system into the onboard system's reference system.
- Enable delay for a mission start, e.g. by waiting on ground for a given period of time.

- Always command a take-off before any other locomotion behavior is performed, to assure any subsequent behavior command does not move the vehicle while still on ground.
- Only one mission repetition command is allowed and only at the end of the mission.
- Only at the end of a mission, aborting or pausing the execution of a mission plan.
- A single land behavior command without a consecutive take off is safe only when commanded at the end of a mission.

In general, EBNF descriptions specify syntax not semantics. Thus, in this notation each EBNF factor is optionally followed by a semantic action (here: parameters to nonterminals). If an EBNF expression is expected at the place the sequence stands, then a semantic action must either be an expression or be omitted. In the first case, semantic actions are particularly important tools to check floating point values against their "meaning" (=semantics), e.g. expected range for degree values. The implemented attributes show the extended plausibility check capabilities. Checks syntactically hard to perform can be handled. For example, it checks against an allowed maximum velocity of a movement behavior or maximum flight height restrictions. Moreover, behavior parameters are validated against a spatial discrepancy between the end and start position of movement behaviors. A start position of a behavior must always match the expected end position of a previous behavior.

$$\begin{array}{rcl}
\text{mission} & \rightarrow & \mathit{uID}\ \ [\mathit{coordt}\,]\ \ [\mathit{wait}\,]\ \ \{\mathit{bseq}\,\}\ \ [\mathit{redo}\,]\ \ [\mathit{land}\,]\ \ [\mathit{off}\,] \\
\mathit{coordt} & \rightarrow & \texttt{coordinates: local\_cartesian} \\
 & & \texttt{datum: wge} \\
 & & \texttt{\#\_ellipsoid: we} \\
 & & \texttt{origin\_latitude: } \mathit{angle90} \\
 & & \texttt{origin\_longitude: } \mathit{angle180} \\
 & & \texttt{origin\_height: } \mathit{ureal} \\
 & & \texttt{orientation: } \mathit{angle180} \\
\mathit{bseq} & \rightarrow & \ \mathit{takeoff} \\
 & & \mid\ \mathit{land}\ \ [\mathit{wait}\,]\ \ [\mathit{off}\,]\ \ \mathit{takeoff} \\
 & & \mid\ \mathit{hoverto} \\
 & & \mid\ \mathit{hoverturn} \\
 & & \mid\ \mathit{waittime} \\
 & & \mid\ \mathit{flyto}\ \ \mathit{flyto}\ \ \mathit{flyto}\ \ \{\mathit{flyto}\,\} \\
 & & \mid\ \mathit{circle} \\
 & & \mid\ \mathit{piroutte} \\
 & & \mid\ \mathit{gatemission} \\
 & & \mid\ \mathit{task} \\
 & & \mid\ \mathit{off} \\
\mathit{takeoff} & \rightarrow & \texttt{TO}\ \ \langle h\rangle\ \ \{\ \forall h \in \mathbb{R}_- : h_{max} \leq h\ \} \\
\mathit{land} & \rightarrow & \texttt{LD} \\
\mathit{hoverto} & \rightarrow & \texttt{HV}\ \ \mathit{position3D}\ \ \mathit{position3D}\ \ \mathit{angleAll}\ \ \langle v\rangle\ \ \{\ \forall v \in \mathbb{R}_+ : v \leq v_{max}\ \} \\
\mathit{hoverturn} & \rightarrow & \texttt{HT}\ \ \mathit{angleAll}\ \ \langle \dot{v}\rangle\ \ \forall \dot{v} \in \mathbb{R} : 0 < \dot{v} \\
\mathit{wait} & \rightarrow & \texttt{WT}\ \ \langle t\rangle\ \ \{\ \forall t \in \mathbb{R}_+ : 0 < t\ \} \\
\mathit{flyto} & \rightarrow & \texttt{FT}\ \ \mathit{position3D}\ \ \langle t\rangle\ \ \langle x\rangle\ \ \langle y\rangle\ \ \langle z\rangle\ \ \langle v\rangle\ \ \{\ \ \forall t \in \mathbb{R}_+, \forall x,y,z \in \mathbb{R}, \forall v \in \mathbb{R}_+ : v \leq v_{max}\ \} \\
\mathit{circle} & \rightarrow & \texttt{CI}\ \ \mathit{position3D}\ \ \mathit{position2D}\ \ \langle v\rangle\ \ \mathit{angleAll}\ \ \{\ \forall v \in \mathbb{R}_+ : v \leq v_{max}\ \} \\
\mathit{piroutte} & \rightarrow & \texttt{PI}\ \ \mathit{position3D}\ \ \mathit{position3D}\ \ \langle \dot{v}\rangle\ \ \{\ \forall \dot{v} \in \mathbb{R} : 0 < \dot{v}\ \} \\
\mathit{gatemission} & \rightarrow & \texttt{GM}\ \ \mathit{position2D}\ \ \mathit{angleAll} \\
\mathit{task} & \rightarrow & \texttt{payloadcmd-on}\ \ \{\mathit{bseq}\,\}\ \ \texttt{payloadcmd-off} \\
\mathit{redo} & \rightarrow & \texttt{REDO} \\
\mathit{off} & \rightarrow & \mathit{WO} \\
\mathit{position2D} & \rightarrow & \langle x\rangle\ \ \langle y\rangle\ \ \{\ \forall x,y \in \mathbb{R}\ \} \\
\mathit{position3D} & \rightarrow & \mathit{position2D}\ \ \langle z\rangle\ \ \{\ \forall z \in \mathbb{R} : h_{max} \leq h\ \} \\
\mathit{uID} & \rightarrow & \texttt{ID}\ \ \langle i\rangle\ \ \{\ \forall i \in \mathbb{N}_+\ \} \\
\mathit{angleAll} & \rightarrow & \langle \alpha\rangle\ \ \{\ \forall \alpha \in \mathbb{R} : -180^\circ \leq \alpha \leq 360^\circ\ \} \\
\mathit{angle360} & \rightarrow & \langle \alpha\rangle\ \ \{\ \forall \alpha \in \mathbb{R}_+ : 0^\circ \leq \alpha \leq 360^\circ\ \} \\
\mathit{angle180} & \rightarrow & \langle \alpha\rangle\ \ \{\ \forall \alpha \in \mathbb{R} : -180^\circ \leq \alpha \leq 180^\circ\ \} \\
\mathit{angle90} & \rightarrow & \langle \alpha\rangle\ \ \{\ \forall \alpha \in \mathbb{R} : -90^\circ \leq \alpha \leq 90^\circ\ \}
\end{array}$$

Fig. 8. Attributed EBNF for plausibility checks of sequence of behavior commands.

5. Integration and Flight Testing

IntechOpen

The Mission Management system is integrated onboard the Autonomous Rotorcraft Testbed for Intelligent System (ARTIS) helicopters (Figure 9).

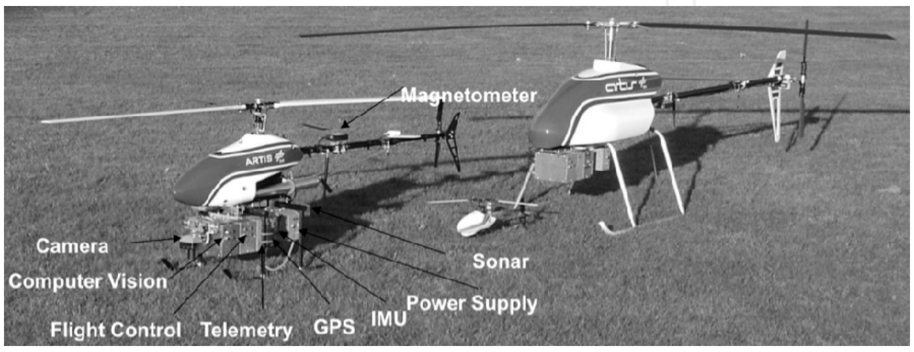


Fig. 9. The ARTIS helicopter UAVs.

The Mission Manager is integrated onboard the flight control computer as a component commanding directly and every cycle to the flight controller. The vehicle state estimates (e.g. position, velocities, acceleration) and further sensor states (e.g. ground distance sensor) are the main input to the system. Furthermore, the ground control station can send instructions on different levels of autonomy as described in the previous section.

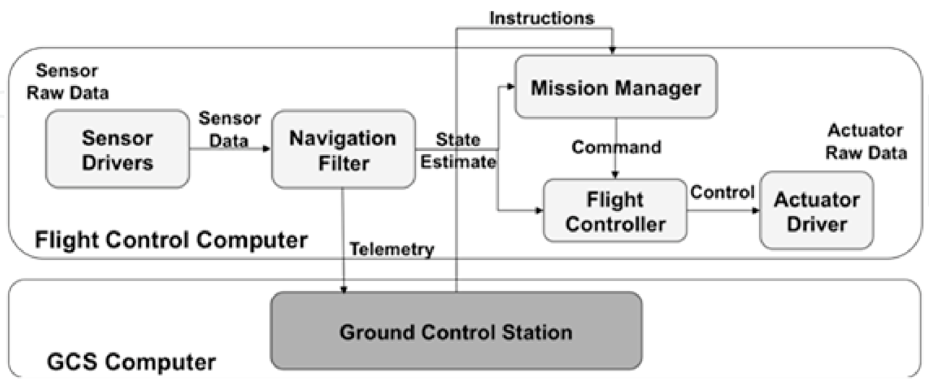
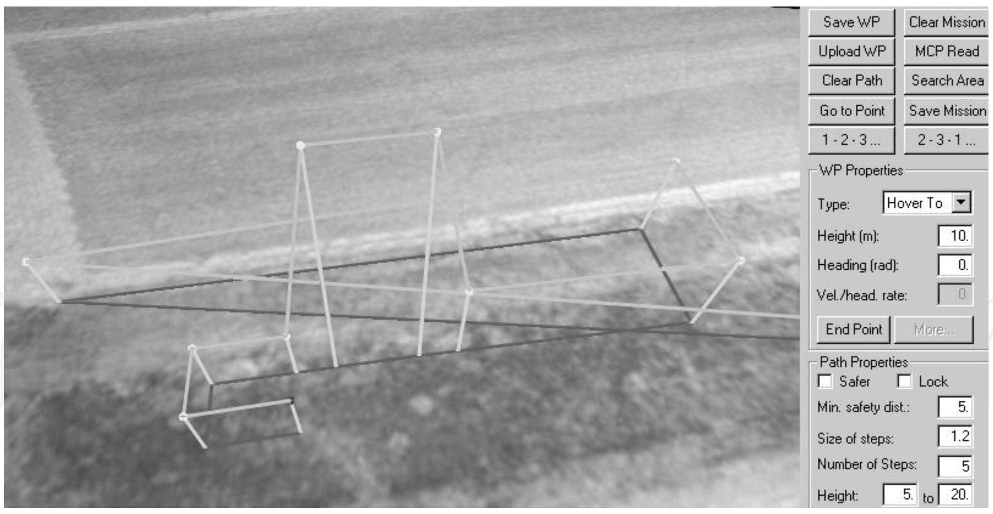


Fig. 10. Integration scheme of the Mission Manager onboard the ARTIS helicopter UAVs.

Before every flight test, the system is tested using the abstract testing method presented in the previous section. Once these tests are passed successfully, the integrated mission management software is tested Hardware-in-the-Loop (HITL) using a set of standard scenarios. Finally, the Mission Manager is flight tested to verify the generation of expected State Chart events (e.g. detection whether the vehicle is on ground or not), state transitions and feasibility of the state abstraction in the design.

Fig. 11. Flight testing onboard the ARTIS UAV (A) using the integrated Supervisory and Sequence Control System (B).

The example scenario in flight test of Figure 11 comprises an automatic take-off, navigation to a set of waypoints and an automatic landing. The behavior sequence is generated once before the test using an automated mission planning system (Figure 12). Once the mission has been accepted by the EBNF and loaded by the Supervisory Control System, it is automatically executed by the ARTIS helicopter UAV.



WP Properties
Type: Hover To
Height (m): 10.
Heading (rad): 0.
Vel./head. rate: 0

Path Properties
☐ Safer ☐ Lock
Min. safety dist.: 5.
Size of steps: 1.2
Number of Steps: 5
Height: 5. to 20.

Fig. 12. Example for an automated planning is used to generate the sequence of behaviors.

The flight test shown in Figure 12 lets the UAV traverse along predefined waypoints, thus focusing on the waypoint navigation capabilities of the Sequence Control System. High-level behaviors of the Supervisory Control System were successfully tested as well. The flight test as shown in Figure 13 addresses the Search and Track behavior. The behavior successfully recognizes convex cells of a search area such that the vehicle does not exceed search cell perimeters while tracking. As a result, the Supervisory Control System autonomously commanded the Sequence Control System using a high-level.

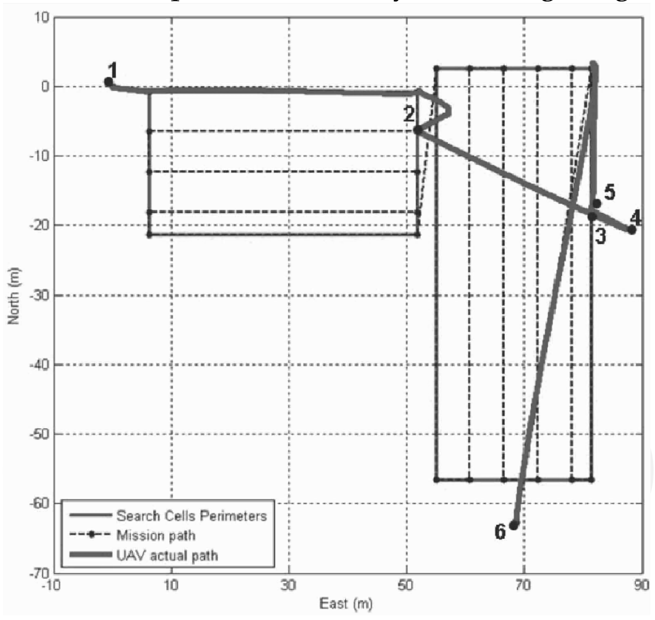


Fig. 13. The Search and Track deliberate behavior shown as it finds and track a ground object with a defined search area

The integrated mission management components system, the Supervisory and Sequence Control System, were first flight tested in September 2006 and since then continuously extended by new features. In particular the implementation of new high-level behaviors is an ongoing activity. While the control architecture remained untouched since its first deployment, the basic movement capabilities are also continuously extended in a "plug-

and-fly" fashion. This way, the architecture and the design consideration of both high-level control components showed to be a feasible solution to the software intensive task of onboard mission management.

6. Summary

This chapter present a 3T architecture combined with a behavior-based approach to integrate different levels of system autonomy onboard of UAVs. The presented approach comprises two State Chart modelled components, the Sequence and a Supervisory Control System, both implementing the sequencing layer of the 3T architecture. The Sequence Control System contains the reactive layer's movement primitives and the Supervisory Control System monitors and manages deliberate behaviors. Both systems are tested in an abstract way and flight-tested successfully.

By design both system components keep the operator in the loop as long as a data link is available. He has the possibility to intercept any running behavior by different kinds of commands, ranging from direct velocity commands (e.g. for remote joystick control), direct position commands, a single behavior (e.g. land) or a complex behavior command sequence.

For each level of system autonomy, the system implements plausibility checks performed at runtime. In particular the attributed EBNF grammar enhances operational safety as it rejects a malformed behavior sequences harming the vehicle or exceeding allowed mission parameters (e.g. the maximum height).

As a result, the ARTIS UAV test platforms are controlled by a robust system that can handle unforeseen events deterministically. New behaviors, both deliberate and reactive, are added to the behavior library such that functional extensibility is facilitated.

Future work will address control architecture extensions as soon as the centralized sequencing layer needs to implement concurrency (e.g. multiple deliberate behaviors need to be active). Moreover, the current system does not allow concurrency between all behaviors and modules in the system. Moreover, a stronger formalism than State Chart models could address limitations in support for temporal design aspects.

7. References

Adolf, F. et.al. (2009). Behavior-based High-Level Control of a VTOL UAV, *Proceedings of AIAA Infotech@Aerospace Conference*, Seattle, WA, April 2009

Agre, P. et.al. (1995). Pengi : An Implementation of a theory of activity, *Computatioanl Intelligence: Collected Readings*, pages 635-644, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1995, ISBN 0-262-62101-0

Bonasso, R. et.al. (1997). Experiences with an Architecture for Intelligent, Reactive Agents, *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, No. 2/3, pages 237-256, April 1997

Brooks, R. (1990). A robust layered control system for a mobile robot, *Readings in uncertain reasoning*, pages 204-213, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, ISBN 1-55860-125-2

Egerstedt, M. et.al. (1999). A Hybrid Control Approach to Action Coordination for Mobile Robots, *Proceedings of IFAC 99 14th World Congress*, Bejing, China, July 1999

Freed, M. et.al. (2005). An Architecture for Intelligent Management of Aerial Observation Missions, *Proceedings of AIAA Infotech@Aerospace Conference*, Arlington, VA, September 2005

Hill, R. et.al. (1997). *Intelligent agents for the synthetic battlefield: A company of rotary wing aircraft, The Ninth Innovative Applications of Artificial Intelligence Conference on Artificial Intelligence (IAAI-97)*, Providence, Rhode Island, July 1997

ISO-14977 (2001). Information Technology ó Syntactic Metalanguage ó Extended BNF, *International Organization for Standardization, ISO/IEC 14977*, 2001

Laird, J. et.al. (1987). SOAR: An Architecture for General Intelligence, *Journal of Artificial Intelligence*, Vol. 33, No. 1, pages 1-64, 1987

Makovski, P. et.al. (2004). Survey on Architecures and Frameworks for Autonomous Robots, November 2004

McCabe, T. (1976). A complexity measure, *IEEE Transactions On Software Engineering*, Vol. Se-2, No.4, December 1976

Musliner, D. et.al. (1995). The Challenges of Real-Time AI, *IEEE Computer*, Vol 28, No.1, January 1995

Pirjanian, P. (1999). The Notion of Optimality in Behavior-Based Robotics, *Journal of Robotics and Autonomous Systems*, 1999

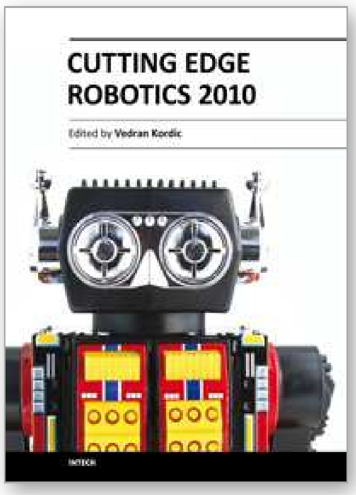
Putzer, H. et.al. (2003). COSA *A generic cognitive system architecture based on a cognitive model of human behavior, Journal of Cognition, Technology and Work*, Vol. 5, 2003

Rumpe, B. (2005). Agile Modellierung mit UML, *Xpert.press*, Springer Verlag, 2005

Taol, D. et.al. (1996). Subsumption Architecture for the Control of Robots, *IMC-13*, Limerick, 1996

Weiss, L.-G. (2005). Intelligent Collaborative Control For UAVs, *Proceedings of AIAA Infotech@Aerospace Conference*, Arlington, VA, September 2005

IntechOpen



Cutting Edge Robotics 2010
Edited by Vedran Kordic

ISBN 978-953-307-062-9
Hard cover, 440 pages
Publisher InTech
Published online 01, September, 2010
Published in print edition September, 2010

Robotics research, especially mobile robotics is a young field. Its roots include many engineering and scientific disciplines from mechanical, electrical and electronics engineering to computer, cognitive and social sciences. Each of this parent fields is exciting in its own way and has its share in different books. This book is a result of inspirations and contributions from many researchers worldwide. It presents a collection of a wide range of research results in robotics scientific community. We hope you will enjoy reading the book as much as we have enjoyed bringing it together for you.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Florian Adolf and Franz Andert (2010). Onboard Mission Management for a VTOL UAV Using Sequence and Supervisory Control, Cutting Edge Robotics 2010, Vedran Kordic (Ed.), ISBN: 978-953-307-062-9, InTech, Available from: http://www.intechopen.com/books/cutting-edge-robotics-2010/onboard-mission-management-for-a-vtol-uav-using-sequence-and-supervisory-control

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen