robotics

Article

Hierarchical Plan Execution for Cooperative UxV Missions

Jan de Gier *, Jeroen Bergmans and Hanno Hildmann

Intelligent Autonomous Systems Group, Netherlands Organisation for Applied Scientific Research (TNO), 2597 AK The Hague, The Netherlands
* Correspondence: jan.degier@tno.nl

Abstract: A generic reasoning approach for autonomous unmanned vehicle (UxV) mission execution is presented. The system distinguishes (a) mission planning and (b) mission execution, treating these as separate but closely interdependent stages. The context of the work is that of tactical military operations, and the focus of the current (simulated) application is on ground-based platforms. The reference behavior for the UxVs is defined by military doctrine. Two operational requirements are met: (1) Mission plan and execution must be constructed such that they can be understood and evaluated (prior to giving the go ahead for the platforms to commence the mission) by a decision maker. (2) Mission plan and execution must account for both observations/information gathered during execution (for example, the spotting of enemy units) and for foreseeable changes in the internal and external situation (e.g., a sub-system failure, or changes in terrain or weather).

Keywords: plan execution; multi-agent systems; coordination; autonomous weapon platforms; UxVs

check for updates

Citation: de Gier, J.; Bergmans, J.; Hildmann, H. Hierarchical Plan Execution for Cooperative UxV Missions. *Robotics* **2023**, *12*, 24. https://doi.org/10.3390/robotics12010024

Academic Editors: Xiaochun Cheng and Daming Shi

Received: 28 December 2022
Revised: 26 January 2023
Accepted: 2 February 2023
Published: 4 February 2023

1. Introduction

There has in recent years been a significant and rapid increase in the availability and the use of unmanned systems in general. This is especially true in the context of military operations. While this is not new (aerial systems have long been used by militaries around the world; the earliest use of drones dates back to at least the Vietnam war [1,2]), the attention this has received from research and the industrial sector has increased significantly. Until recently, this was predominantly in the context of remote controlled or remote piloted systems, that is, systems that are still controlled by a human operator. The benefits of this include the safety of the operator, the ability to have a team of operators to take turns, etc.

In the civilian context, ground-based platforms have been remotely operated for, e.g., Search and Rescue (SAR) missions [3] and infrastructure inspection [4,5], for crime scene inspection/material (bomb) removal (by civil defense units such as police) [6] or to inspect, e.g., nuclear installations in the aftermath of a natural disaster [7].

At The Dutch Organization for Applied Scientific Research (TNO) we have researched remotely operated systems since 1937 [8] (making TNO among the first in Europe to do so). Due to advances in hardware and software and driven by the increasingly pervasive nature of unmanned platforms [9], we are currently witnessing a paradigm shift from using remote controlled systems to using autonomous systems.

In the military domain, the advent of loitering ammunition [10] (sometimes referred to as kamikaze drones [11]) has had a massive impact on operations and is considered to be a (tactical) game changer [12,13] (note that the authors are not fond of this term, as the application and the application domain are not a game to those participating in it). The conflict often mentioned in this context is the war between Armenia and Azerbaijan over the disputed Nagorno-Karabakh region in 2020 [14], where Azerbaijan deployed drones with devastating effect (for details we refer to this post by the Center for Strategic and International Studies (2020): https://www.csis.org/analysis/air-and-missile-war-nagorno-karabakh-lessons-future-strike-and-defense, accessed on 1 February 2023).

Thus, far, the complexity of the operations of these unmanned and autonomously operating weapon platforms has been relatively low, and there is little or no cooperation between multiple platforms. In the conflict in Ukraine that started in February 2022, there has been widespread use of autonomous (mainly aerial) vehicles, again mainly using drones to autonomously (*fire and forget*) search out and then attack pre-specified targets.

Techniques that address methods for cooperative execution by multiple platforms [15,16], in a generic way and for real world applications, is still limited. As is mentioned in [17], planning can be considered the "reasoning side of acting", but in relevant dynamic environments, the actual integration of acting (execution) with planning is still insufficiently researched. A generic approach towards autonomous mission execution and planning for cooperative unmanned (autonomous) systems or vehicles is not yet available.

At the same time, given the political situation(s) around the world and the funding allocated for military spending, the question is not whether this will soon be in use by militaries around the world but only when we will first encounter it in uncontrolled settings.

In this article, we describe an approach developed for the autonomous mission execution and planning of tactical military operations of the Royal Netherlands Army (RNLA) by unmanned ground vehicles (UGVs). It is aligned with the efforts of our colleagues, as described, e.g., in [18]. The application for the work presented in this article is the deployment of intelligent and autonomously operating cyber-physical systems (i.e., mobile, and when applicable, armed platforms that observe and act without human control). It has been undertaken in close collaboration with the Netherlands Ministry of Defense (NLMinDef).

Specifically, we propose a generic automated reasoning methodology and corresponding software which we refer to as Intelligent Coordination of Tactical Unmanned Systems (ICTUS). It encompasses automated planning and execution modules for military missions executed by a team of autonomous vehicles. We address the reasoning over, amongst others, the terrain, enemy courses of action, and the own course of action, that is, the coordination of the tasks for individual autonomous vehicles, in order to achieve user-defined mission goals. Taking these reasoning elements into account during both the planning and the execution of the mission is essential to mission success.

1.1. Scope

The envisioned use is ultimately the tactical execution of a mission for a cooperating team of UGVs in a dynamic and possibly adverse environment, reacting in a reliable and effective way to achieve mission objectives (i.e., in real-time and during combat operations).

However, the scope of our work, as presented here, is a research and development effort and does not yet extend to the actual deployment of the autonomous platforms in real applications, in no small part owing to legal and ethical reasons [19]. In addition, this paper does not address the control challenges that arise from operating autonomous platforms in the real world (as discussed, e.g., in [20]). We assume that such systems are commercial products, purchased with those issues addressed.

The performance evaluation is based on simulations; there has not yet been a validation by deployment in the real world with a human in/on the loop. Furthermore, it should be understood that the work we report on is, arguably, primarily an engineering achievement. We designed and developed a system—albeit thus far only for simulations—that does not yet exist in the literature. This makes a comparative numerical evaluation difficult. Classical systems that solve the same problem rely on recalculating and replanning, whereas we specifically avoid this. One way to evaluate performance is thus to point out that our approach does not require replanning. Comparison on the basis of, e.g., execution time, would require parallel implementations of alternative approaches (as well as some significant tailoring thereof). This is well outside the scope (and budget) of our work.

1.2. Motivation

The following considerations are the most pressing motivations for developing a generic automated reasoning methodology in the context of tactical military operations:

- Planning ahead and coordination actions has been a fundamental part of maneuver warfare [21] for hundreds [22], and indeed, thousands of years [23].
- The realization of planned actions is undertaken by individual units, and as such has to be described, and performed, in a specific manner. This empowers other units to act meaningfully and enables multiple units to perform combined arms operations.
- Given that militaries operate under *rules of engagement* and are subject to legal and ethical guidelines, plans must be such that their impact can be understood by decision takers. Military conflict operates at the fringes of ethics, and no comprehensive set of *if–then rules* exists; we can, however, calculate and offer possible choices to a human commander, who uses his expertise and understanding of the situation to decide.

The above motivation does not specifically address autonomous systems and is equally relevant and applicable for tactical operations undertaken by human personnel. However, given the rapid progress for mobile autonomous platforms, the following are our main motivation for specifically considering unmanned systems:

- Military operations are inherently dangerous, and we believe that human life is more important/valuable than robotic hardware.
- Tactical military operations are inherently complex, and computer systems are faster and more reliable than humans at processing large amounts of real-time information.
- Human personnel will always be comprised of individuals with subjective talents and performances. Cyber-physical systems are significantly more consistent in their performance and durability and are furthermore not susceptible to fatigue and exhaustion.

1.3. Contributions of This Article

This article makes the following contributions:

- We present an automated reasoning approach that facilitates the planning and execution of combined tactical operations. The approach specifically allows the cooperation between multiple, physically separate platforms.
- The presented approach does not require replanning in case that the actually encountered situation differs from the initially assumed state of affairs.
- The approach offers choices which are stated in the form of concise descriptions of the high-level actions (and their consequences) and are comprehensive (complete). Our system is specifically designed with an informed human in the loop.

1.4. Structure and Overview of This Article

In this article, we present our work towards mission preparation, mission execution, and coordination among multiple tactical unmanned systems. As our work is situated in a specific domain, we first provide some background (Section 2) and then propose a framework facilitating the mission planning for such systems (Section 3). We discuss the execution of tasks by an autonomous platform (Section 4) and the planning/execution of missions by multiple such platforms (Section 5) separately, before elaborating on the implementation (Section 6), which also serves as a demonstration and a proof of concept. We close with the usual conclusions and planned and scheduled future work (Section 7).

2. Background

2.1. Application Domain

Robotics [24,25] is increasingly becoming an integral part of everyday life. This is happening as we, as a society, are shifting from automated processes (such as repetitive factory work performed by robots) to autonomous systems (where machines decide on a course of action, within a provided scope, and where these decisions are made in reaction to changes they observe and based on their own perceptions of the world around them). In other words, we are moving from automating processes to enabling intelligent machines to act independently. Some consider this the step towards revolutionizing fundamental aspects of our society, with regard to both people and nature [26].

Robots have become more affordable and are increasingly found in, by now, almost all aspects of society. In the last 20 years, there has been another development in this area, as we moved from single systems to collectives of systems (or swarms [27]) [28]. While this may initially have been a simple aggregation of physically disjoint but otherwise identical systems, it is increasingly a matter of combining different hardware with different, often very specific, hardware: heterogeneous groups [29]. The advantages are obvious: dedicated hardware can be allocated and used where it is needed, as opposed to being included by default. Be it homogeneous or heterogeneous, enabling multiple systems to work together can at the same time reduce system cost while increasing system capabilities and usability. It does come, however, with a series of new challenges that were not an issue when considering single systems. Among these challenges are the coordination of the actions of the systems (over time or space) and that operating in dynamic environments means that the internal representation a system maintains of the environment (including its peers) will have to be able to handle uncertainty.

The challenges, some of which are specifically discussed below in Section 2.3, as operational constraints imposed on the approach presented in this paper, are complex and hard to overcome. For some, no complete solution will exist, so compromises and trade-offs will need to be made. However, in many applications and/or tasks, the benefits of cooperating groups of platforms are considerable and outweigh the challenges.

For militaries around the world, the emergence of robotics and automatic systems can be considered a critical innovation. Such systems have the ability to sense, process the sensed data, and act upon this information; and they can do so with minimal human guidance or intervention [30]. The following application scenarios are examples of possible useful deployment of UxVs. Many more can be envisioned:

- **Information gathering: Intelligence, Surveillance and Reconnaissance (ISR):** Groups of heterogeneous platforms with various (sensing) capabilities can be used to continuously and tirelessly provide surveillance for, e.g., protection for forward bases and troops on the move. This requires an interface with the human commander to task the autonomous team with overall objectives and provide the rules of engagement, and an existing and agreed upon plan between the platforms of how to collectively react to specific events.
- **Infrastructure maintenance:** The extremely dull (and potentially quite dangerous) task of continuously checking infrastructure, such as pipelines or mine fields, can be undertaken by autonomous platforms which are expendable yet precise and reliable. Additionally, here, a commander has to be able to state high-level directives and operational boundaries while the platforms operate according (and possibly adapt dynamically) to a plan agreed to in advance.
- **Protective mother—child systems:** New military platforms—tanks, ships, etc.—are increasingly equipped with support UxV systems that can serve as extended sensors and/or effectors as part of the protection system. The primary tasks and objectives can be set by the human commander, but the execution of protective actions has to happen within such a short time that in order to be meaningful, autonomous cooperative operation is necessary.

2.2. Requirements for Autonomous Operations

As shown by the examples in the previous section, whether it is for civilian or military applications, three high level requirements emerge when considering the deployment of multiple, possibly heterogeneous cyber-physical platforms in realistic scenarios:

R1: **Plan executable missions for autonomous platforms in dynamic environments.** The system needs the ability to collectively plan and execute a mission which may require reactive decisions (decisions to be taken based on the perceived situation in a dynamically changing environment, e.g., atmospheric conditions [20]).

R2: **Return a set of choices (plans) to a human decision maker for the final decision.** We have to offer a high-level description of the various choices to the human operator such that informed and timely operator control is possible.

R3: **Enable the human decision maker to make an informed choice.** This requires the plan to be described in a concise (short), understandable but also comprehensive (complete) way.

2.3. Operational Constraints

Using autonomous platforms has been shown to be non-trivial, especially in (highly) dynamic environments (battle [31], disaster responses [4,32], emergency operations [4,9], etc.) and with heterogeneous vehicles that have multiple capabilities available and a multitude of missions in which they can participate. Specifically with the military domain in mind, operating multiple autonomous platforms is very challenging [33]. From the above listed requirements (Section 2.2), we derive four operational constraints that need to be addressed for a successful deployment of autonomous platforms for (military) missions:

C1: **Mission execution in dynamic environments**: Autonomous platforms operating collaboratively in the real world to achieve some high level goals and objectives must be able to operate under changing conditions. These platforms need to be able to react to new information obtained during execution in the environment (changing weather conditions, an unforeseen change in terrain conditions, etc.). Furthermore, deployed platforms have to be able to react to the behavior and goals of other actors (be they friendly, neutral, or hostile), be they human or cyber-physical, and be able to handle anomalies in their own systems, such as sub-system failures or draining batteries.

C2: **Team coordination**: Multiple unmanned vehicles may be deployed in a cooperating team to achieve the overall objectives, but they will do so through individual tasks. These tasks, or more precisely, the effects of these tasks, need to be coordinated to achieve the best results. Cooperative operations may thus require coordination between vehicles as a vital aspect of the mission execution. Ensuring coordination between the platforms should therefore be a pivotal concept in the plan. Given the domain (military, or at least, adversarial), the coordination between vehicles may not be able to rely on perfect communication between the platforms in the collective, something that, e.g., Cognitive Radio (CR) [34] could offer. The matter of how the communication is realized (e.g., [35]) and to which extend it is trustworthy is ignored in this manuscript, as it is work outside the scope of the project. As stated elsewhere in this paper, the actual deployment of such platforms in uncontrolled settings still faces a number of challenges. By the time this can realistically be attempted, new or emerging technologies (e.g., StarLink/StarShield, https://www.space.com/spacex-starshield-satellite-internet-military-starlink, accessed on 1 February 2023) may have permanently changed the battlefield. For the remainder of this paper, the technical details of how communication is realized are omitted.

C3: **Predictable and accountable behavior**: Military doctrine (see, e.g., https://english.defensie.nl/topics/doctrine/defence-doctrine, accessed on 1 February 2023) [36,37] is used extensively in the armed forces. It ensures a shared understanding and results in a certain level of predictability in mission execution. At least the same level of predictability that exists between humans should be expected from autonomous platforms, since those vehicles will cooperate with other platforms, and eventually, also in collaboration with human soldiers. Simultaneously, the platforms should act in an accountable way and in accordance with their roles and responsibilities.

C4: **Human approval**: Military decisions are subject to final human approval. For this statement to be meaningful (i.e., the decision to be an informed one), the human decision maker must be able to understand the implications and consequences of a plan. This means that the plans must be expressed in a high-level, human-readable manner. Constructing a comprehensive plan in advance (before execution) involves planning for a (possibly) very large time horizon. Creating such a long-term plan is computationally infeasible without providing contextual information that helps to decide on tasks and the behavior of the platforms.

2.4. Automated Planning and Execution

Classically, a planning problem is a problem that requires determining a course of action, i.e., a series of activities that need to be executed, in order to achieve a goal that is specified as part of the problem. A solution to a planning problem is a complete plan that achieves that desired goal, where a plan is a representation of the activities and consists of a set of tasks, together with temporal, causal, and possibly other constraints on those tasks. Typically, the objective is to determine a plan that is both executable and optimal (expressed in some problem-specific measure of performance). Automated planning is a field of research that aims to automate the planning process by designing algorithms that construct plans, which are typically autonomously executed by unmanned vehicles.

We use the term task to denote activities performed by autonomous systems that have an effect in the world. In other words, a task can be understood as summarizing a specific behavior (with a specific set of consequences in mind). Examples of tasks include repositioning, perceiving something, and interacting with objects in the world. Note that in the planning literature, tasks are often referred to as actions, and the execution of a plan (typically in a dynamic and unpredictable environment) as acting.

2.4.1. Planning

Automated planning [38] requires the modeling of the real world, the goals that need to be achieved, and the behavior corresponding to each of the tasks that can be performed by the unmanned systems. The task models predict the outcomes of the task behavior in the real world, which allows a planning algorithm to find a sequence of tasks, a plan, that will lead to the desired goal.

In automated planning, a distinction is made between domain-specific planning and domain-independent planning. The difference is that in domain-specific planning, algorithms use domain-specific knowledge and information to solve planning problems efficiently by exploiting the modeled knowledge of that domain.

Examples of domain-specific planning include algorithms for playing board games such as chess [39] and Go [40] (for an overview, see [41]). These algorithms use game-specific information such as rules and strategies. Another example is the automated planning of airline flights, which use information on distances and flight times between airports to generate efficient flight schedules [**?**].

Domain-independent planning, on the other hand, uses general-purpose representations and algorithms, which can therefore be applied to a range of domains. Thus, although domain-specific planning can be more effective for solving specific problems, the algorithms in domain-independent planning can be applied more generally.

The most prominent form of automated planning involves so-called state space search, where the planning algorithm searches through a tree of states. A state is a model of the system together with the environment for which a plan is constructed. The root of the tree is the initial state, edges represent the execution of a task in a state to arrive at another state. The search algorithm traverses the tree, exploring different branches and expanding the search space until a sequence of tasks that leads to the desired goal is found.

A second approach in automated planning is referred to as plan space planning. This form of planning involves the decomposition of a complex planning problem into smaller subproblems [15], after which each of the subproblems is solved. This approach allows for reasoning about each of the subproblems relatively independently. The subplans, the solutions of the subproblems, are combined into a complete plan that solves the original planning problem. In this case, the search space encompasses all possible plans. A well-known form of plan space planning is hierarchical task network (HTN) planning. In Section 2.4.2, a short description of the HTN method is given, as it was our inspiration for the hierarchical task decomposition concept [15], as used in ICTUS.

2.4.2. Hierarchical Task Network Planning

In HTN planning, the decomposition of a planning problem into subproblems is hierarchical in nature (hence the name). The top level in this hierarchy represents the goal of the problem (here: our mission), and the lower levels detail the goal into subgoals, each of which must be solved to achieve the goal of the planning problem. A detailed overview of HTN planning can be found in [43].

A typical application for HTN planning is video games (see, e.g., [44]), where the planning is used to generate realistic behavior for non-player characters in these games. We use an approach to HTN planning that is primarily based on the work as described by [31]. In this paper, the coordination of units in a video game, to be able to execute combat maneuvers, is made explicit in the HTN and the resulting plan. An important aspect of the approach is that there is no single state that is available to, and adjusted by, the tasks. Instead, information between tasks is shared by connecting so-called task inputs and task outputs. This approach enables the decomposition of tasks into lower level tasks in any desirable order, instead of in (foreseen) chronological order.

HTN planning can be domain-independent or -specific, depending on how it is implemented. It can use domain-specific information to drive the decomposition. This can be augmented with general-purpose (domain independent) information representation and search algorithms. In our HTN, we use algorithms specialized for military reasoning.

2.5. Discrete Event Simulation

In our approach towards plan execution, we use a separate component, the *supervisor*, which is responsible for keeping track of plan progress and the dispatching of tasks to the individual unmanned systems (assets) that execute tasks in the real world. This is done by employing a Discrete Event Simulator (DES) process, a simulation process described briefly below. We refer to [45] for an extensive overview of techniques and applications.

A DES is a computer program that simulates the behavior of a system that consists of multiple entities. The behavior of the entities is modeled as processes that generate discrete events. An event marks a changes in the state of one of the processes or in the relationship between processes, and occurs at a particular point in time. These discrete events determine the behavior of the individual entities, and the entities together determine the behavior of the system as a whole. Note that in between events there are no changes in the system, and the simulation can jump to the next time instance at which an event occurs. DESs are often used to model and analyze complex systems in fields such as engineering, manufacturing, and logistics; see, for example, [46].

As mentioned in the above, we use a DES for the execution of a plan. Every task in the plan hierarchy is a process with discrete start, interrupt, and finish events. In addition, incoming data, such as observations and status updates from the assets, are also events in the DES. All planned coordination between tasks is expressed as event sets that are logical conditions to trigger other events, forming a coherent representation of the plan that can be executed by processing events step by step. More details are given in Section 4.

3. ICTUS: Intelligent Coordination of Tactical Unmanned Systems

3.1. The Framework

The developed reasoning and execution framework, which we refer to as ICTUS, and the phases required for the deployment of its system in a mission, are illustrated in Figure 1. Various models are provided to the framework to specify different aspects of the mission. Such models can describe a complex set of rules and behavioral paradigms to control what tasks can be performed by the autonomous systems in the current environment.

Starting from mission design, the following three phases increasingly refine mission requirements and goals towards the actual realization of the mission in the environment:

1. **Mission design**: Design doctrine based mission templates, choose matching analysis methods, and specify necessary asset task capabilities for the preparation phase.

2. **Mission preparation**: Generate a specific plan based on mission parameters and available assets; and choose matching assessment methods for the execution phase.
3. **Mission execution**: In real-time, assess the situation regarding the environment and assets, determine actionable runtime information, and *dispatch* planned tasks.

In each phase, both concrete information and reasoning methods are chosen and passed to the next phase and more immediate time scale and abstraction level. Automation is used in all phases, but autonomous behavior of the system increases toward the execution phase. As indicated in the figure, the ICTUS framework scope is focused on the cooperative and autonomous levels of mission execution.

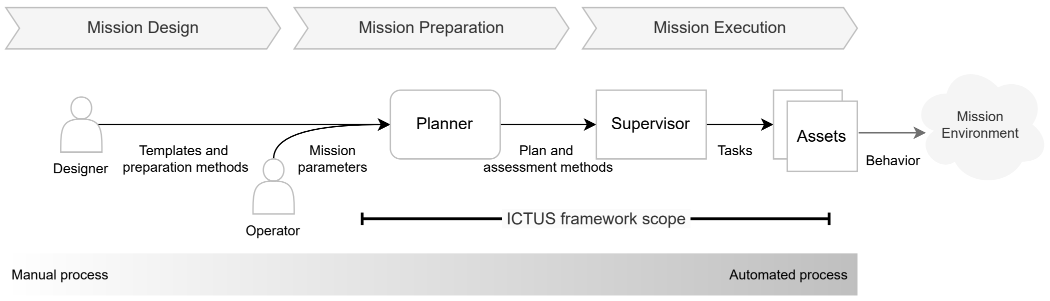

Figure 1. Phases in mission development: from the manual design phase to the execution of the mission in theater. In each phase, both the concrete information and the reasoning methods (used to assess new incoming data) are passed. They are increasingly tailored to the specifics of the mission objectives. The primary scope of the ICTUS framework is indicated.

Given the descriptions of the environment in which the team of assets will have to operate, together with a set of orders defining the goals that need to be achieved in that environment, planning can commence. A planner constructs plans, which often have to be approved by a human operator. Operator analysis of a plan may trigger a refinement of the orders given and thus resulting in a revision of the plan.

Once plans have been approved, they are submitted to a supervisor for execution. During execution, the supervisor interacts with the autonomous vehicles (the assets) to perform tasks in the environment. The execution phase in which the assets are deployed may no longer include the human operator, dependent on the situation.

What about Replanning?

Generally speaking, the more interesting or relevant a situation, the more likely it is to be subject to change. Especially in the military domain, this poses challenges [33], and one has to continuously assume that the world does not always behave and react as one has expected. When events which interrupt a planned sequence of actions occur during the execution of a plan, contingency plans [16,47] should already be in place to enable the executing asset to react in accordance with doctrine and the rules of engagement.

As preparing such contingency plans in advance can require significant additional efforts, many approaches rely instead on a replanning process [48]. Once such a process is triggered, the execution of the current plan is suspended or reset, and a new plan is generated on the basis of the changed information now available to the planner. However, depending on the complexity of the situation and the mission, generating a new plan, especially in a multi-asset operation, can be time-consuming. Given the adversarial nature of the domain, reactions to events in the environment are ideally instantaneous or at least as immediate as possible, which in tactical situations may need to be very fast.

In the military, one relies on doctrine and standardized behaviors and responses. The field has been subject to intensive studies (e.g., [21–23], and as a result, many (if not all) of the most disruptive events are known and can therefore be anticipated. In ICTUS, the generated plan contains contingency sub-plans for all anticipated disruptive events. Since these reactive behaviors (including the "fallback" or "abort" actions) can be part of the general plan, the individual assets can themselves react appropriately to any other asset in their team by falling back on a contingency.

The term replanning, as used in ICTUS, exclusively refers to the human operator stepping in and aborting the mission so as to trigger a new planning process on the basis of new/revised mission goals or using new mission parameters/conditions.

3.2. System Modules

Figure 2 offers an overview of the primary functional modules and their interactions. The individual building blocks, which form the ICTUS system, are explained individually below: the PLANNER (Section 3.2.1) constructs the plan before the mission; the unique SUPERVISOR (Section 3.2.2) and its assessor, together with the TASK MANAGERS (Section 3.2.3), one for each autonomous asset, execute the plan.

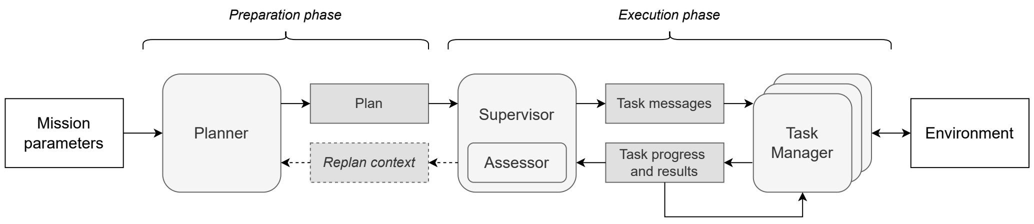

Figure 2. An overview of the primary functional modules and their interactions. Two separate phases are distinguished: the preparation or planning phase (in which the PLANNER module is active) and the execution phase (involving the SUPERVISOR and the TASK MANAGER module). The former happens offline to generate the plan and have the human operator approve this plan, and the latter has all assets execute their tasks separately while being coordinated by the supervising module.

An example of ICTUS modules being deployed on physical hardware is given in Figure 3. In this example, the PLANNER is available on a base station, together with a human–machine interface (HMI) that allows for interactions with an operator. The SUPERVISOR is deployed on one of the autonomous assets, and all the autonomous assets have a TASK MANAGER. The interactions between, e.g., the PLANNER and SUPERVISOR, the TASK MANAGERS themselves, and HMI (and hence the operator) and the SUPERVISOR, are supported by a (typically wireless) communication infrastructure.

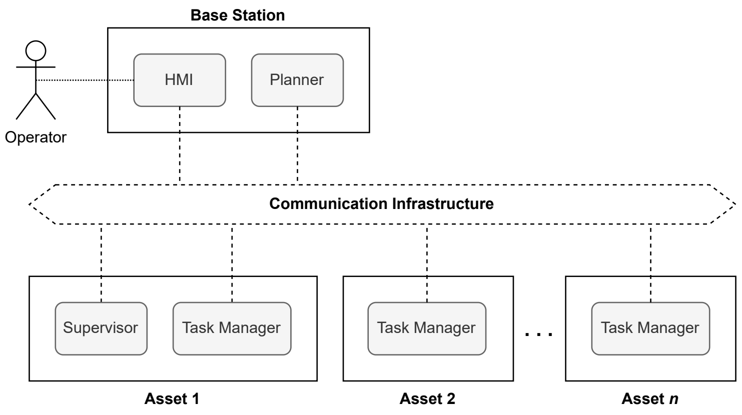

Figure 3. An example of several ICTUS modules running on hardware: the PLANNER runs on a base station and the SUPERVISOR on one of the *n* autonomous assets, all of which also run a TASK MANAGER.

3.2.1. The PLANNER Module

The main functionality of this module is to interpret the available information, e.g., the assets and their capabilities that are available for the mission, and the operator-provided goals and constraints for the mission. This information is converted into a mission plan object in a planning process. To construct the plan, it is required to assess anticipated uncertainties and construct contingencies for these uncertainties to ensure that the mission goals are achieved. Examples of such uncertainties involve not-yet-known information about terrain conditions, possible locations that provide cover and concealment, weather conditions, the health status of (sub-)systems, and the intents and courses of action of expected opposing forces. The PLANNER module is discussed in detail in Section 5.1.

3.2.2. The SUPERVISOR Module

The SUPERVISOR and the TASK MANAGERS together are responsible for the execution of the mission plan. The SUPERVISOR module uses a DES as a plan execution model. It dispatches task information to the individual assets and receives task progress and result information from these assets. In this context, the assets are referred to as *workers*, as they perform tasks in the environment to achieve the mission goals. Additionally, the SUPERVISOR keeps track of the mission's progress and performs assessment of both the perceived information and the mission plan through an internal ASSESSOR sub-module. The SUPERVISOR module is discussed in detail in Section 5.2.

3.2.3. The TASK MANAGER Module

The responsibility of the TASK MANAGER is to effectively instruct the components of the asset (either real or simulated hardware). Each TASK MANAGER interprets the received task information and perceives and interacts with the environment and entities in that environment during its task execution, while providing feedback on the execution and results of the task. The protocol of task-related messages between the SUPERVISOR and TASK MANAGERS is one of the primary interfaces in the ICTUS framework and crucial in the application of ICTUS. The TASK MANAGER module is discussed in detail in Section 5.3.

4. Task Execution

4.1. Task Execution Requirements

To coordinate the planned behavior of a team of autonomous assets, the task execution model that is used by the SUPERVISOR has a number of requirements:

- Tasks have to be grouped into a meaningful hierarchy of tasks consisting of (higher-level) compounded tasks—which have subtasks—and *basic* tasks as the leaves of the hierarchy. Using this hierarchy, the overall intended behavior can be constructed through (reusable) task definitions. The same definitions can be used to keep track of mission progress and to provide sufficient context for when the outcome of the task execution differs from what was expected.
- Tasks can be executed concurrently. While an individual asset executes tasks in series, the supervisor must manage several tasks and subtasks in parallel.
- Plan execution must allow for *runtime resolving* of task parameters. For example, after the observation of a contact, it should be possible to assign a FOLLOW CONTACT task using the information from that observation. This specific information will only be available during execution, that is, at runtime; therefore, the plan must provide for placeholders of that information. In ICTUS, this information is called *runtime data*.
- To allow for the composition of complex team behaviors, a (compound) task must allow for repetition. In other words, a task should repeat while (or, alternatively, until) predefined conditions are met. Examples include the continuous scanning of an area or the repeated engagement of targets during an attack.
- A branching mechanism is required to support decisions made during execution. While conditional execution of tasks is an important approach to supporting this, we introduced a decision task to explicitly account for this functionally in the plan object.
- There should be a detailed description of the basic tasks that the assets can perform autonomously. These tasks can be performed without coordination by the supervisor and are possible while interacting with other assets. The basic task descriptions determine

the interface to the task managers of the assets. For this reason, the implementation of this interface will be application and use-case-specific.

In Figure 4, an overview over the task hierarchy and the different types of tasks is given. The highest level task is the plan or course of action.

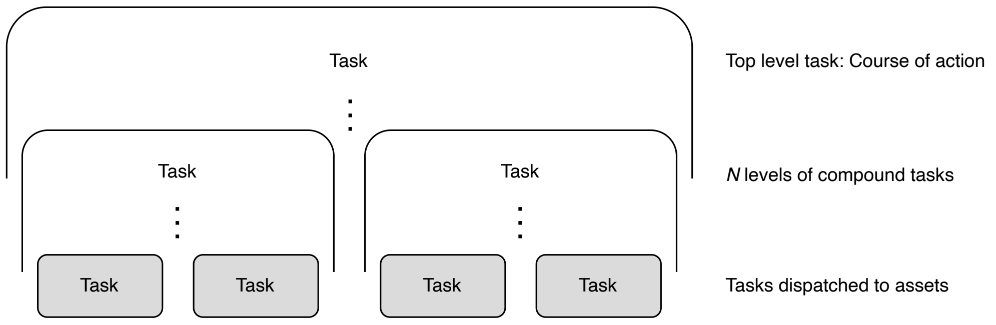
Task
Top level task: Course of action
Task
Task
N levels of compound tasks
Task
Task
Task
Task
Tasks dispatched to assets

Figure 4. An illustration of the task hierarchy with the three types of tasks: top-level (the plan), compound tasks, and basic tasks.

Note that, in the context of this article, a plan encompasses the design and specification of cooperative behavior and coordination that is crucial during the execution phase. Related to the implemented use case involving a defense operation (see Section 6), their definitions aim to follow military doctrine as closely as possible.

The complexity of the compound and basic tasks range from relatively straightforward to very complex. Choosing the appropriate conceptual level of these tasks depends on the ability of assets to individually or jointly perform certain tasks. Another aspect to take into account is whether the tasks should be part of the mission plan explicitly and how they may be represented to human operators before and during execution. For the use case and demonstration, the level of abstraction is primarily based on the extent to which an asset is able to individually execute the task.

4.2. The Task Execution Model

Each task within the hierarchy of tasks (the plan tree) is either a compound task or a basic task. A compound task contains one or more subtasks. An basic task (the leaves of the plan tree) can be dispatched to and autonomously performed by an asset.

This hierarchy provides a clear and explainable distinction between team-level coordinating mission tasks and individual asset tasks. The transition from mission-level information to individual task data naturally follows from this structure and is tractable at both design time and execution time.

Execution of tasks is event driven: the planned task hierarchy is not scheduled in time like a train table, but each task has `start`, `finish`, and possibly `interrupt` events. These events can be used to trigger the execution of other tasks. Each planned task is provided with several condition sets for when to start, interrupt, repeat, or finish the task. The execution process for a task is explained in Section 4.3.

The supervisor oversees and manages this process by dispatching asset tasks and receives responses (task status messages, feedback messages, observations, interventions, etc.). In this way, information from the environment is received by the supervisor module. The received data are processed by the assessor module to produce actionable information. This information is used during plan execution in the form of so-called external events.

External events are additions to the start, interrupt, and finish events of the tasks. These events originate from external data sources (messages, asset feedback, observations, etc.) and are emitted by the assessor when these data are relevant for the execution. By the mechanism of generation of external events, together with internal task events, the plan execution can react properly to the mission, the environment, and intermediate results in a pre-described, and hence planned, manner.

4.3. The Task Execution Process

4.3.1. Execution Conditions

The execution of a task is determined by four sets of conditions. For each, a simple example is given to illustrate how the condition would appear in our implementation:

1. **Start conditions:** When all start conditions are satisfied, the task is started. Take a simple example plan involving two tasks, task A and task B, where task B starts after task A has finished: `taskB.starts_after(taskA.finish)`, where `taskA.finish` is the finish event of task A.
2. **Interrupt conditions:** When all of these conditions are satisfied, the task and its subtasks are interrupted as soon as possible. As an example, consider an external event `enemy_observed` being triggered after an appropriately assessed observation: `taskA.interrupt_when(enemy_observed)`.
3. **Repeat conditions:** Whenever all conditions are satisfied, the task is restarted. If applicable, all its subtasks are recreated and processed. Consider the following example: `taskA.repeat_after(taskB.finish)`.
4. **Finish conditions:** When all are satisfied, the task is finished and will produce a finished event. An example: `taskC.finish_after(any(taskA.finish, taskB.finish))`.

In the first three examples, a single event was used as the condition, but as the last example shows, a set of conditions can be turned into a complex condition by using logical operators to combine events and/or other condition sets. As shown in Figure 5, conditions can be combined using either the `ANY` (triggered as soon as one of the events is triggered) or the `ALL` (triggered when all input events are met) operator. Figure 6 shows a schematic of the execution process together with all four condition sets.

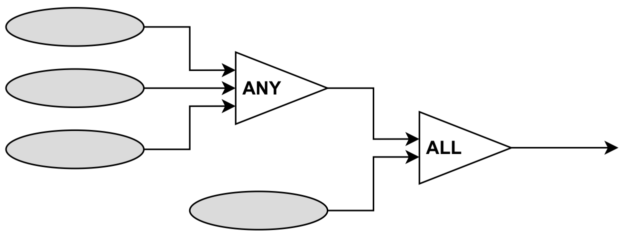

Figure 5. Task execution is determined by a number of conditions (cf. Section 4.3.1); conditions can be logically composed using the `ANY` and the `ALL` operators.

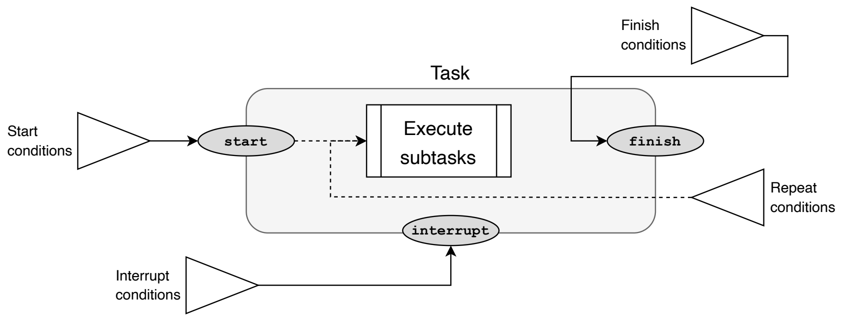

Figure 6. The task execution process showing all 4 execution conditions introduced in Section 4.3.1.

4.3.2. The Execution Process

The outline of the execution process for a task is as follows:

1. Initially, the task awaits for its start conditions to be satisfied.
2. The start of a task could be disabled (by an event), which will result in the interruption of the task, which means that the task will not execute and hence never start.
3. When started, all subtasks are concurrently processed (all await their start conditions), and the task will await either its repeat or its finish conditions. When either of these conditions is triggered, all active (started) subtasks will be interrupted, and in the case

of the repeat conditions, the task is restarted. Alternatively, when the finish condition is satisfied, the task finishes.

4. In the case of a basic task—which is a leaf in the task tree—there are no subtasks. When a basic task starts, a TASKREQUEST is sent to the appropriate asset, and the process awaits its response. When the request is accepted, the task is marked as started and the task awaits a response (that is, a task result message) from the asset. Upon receiving this response, the task finishes. When the task is rejected by the asset, the task is disabled. If the task has to be interrupted, a cancel request is sent to the asset in order to stop its execution.

The pseudocode for the execution process of a (compound) task is given in Algorithm 1, and the task conditions and events are schematically illustrated in Figure 6.

Algorithm 1 Task execution procedure.

```
1:  procedure PROCESS(task)
2:      try:
3:          wait for start conditions
4:          repeat
5:              Perform task                    ▷ Send TASKREQUEST or queue subtasks.
6:              emit task start
7:              wait for repeat or finish conditions
8:          until finish conditions
9:          emit task finish
10:     except interrupt conditions                        ▷ emit task interrupted
11: end procedure
```

5. Mission Planning and Execution

5.1. Mission Planning: The PLANNER *Module*

The hierarchically organized plan described in Section 4.3.1 is constructed autonomously by the PLANNER module (cf. Section 3.2.1). The focal point of the planning approach presented in this article is the iterative decomposition of compound tasks into lower level tasks, an approach that is adopted from HTN planning (briefly discussed in Section 2.4.2). However, the concept of HTN planning (and hence the many implementations that are discussed in the literature) does not address all requirements for execution (as listed in Section 4.1). Taking these requirements into account results in a more extensive planning input and a more elaborate planning process compared to classical HTN planning. In contrast to HTN planning, which constructs a plan consisting of basic tasks only, a hierarchy-preserving approach was taken, based on [31]. All tasks that are decomposed remain available in the mission plan. This allows for assessing the mission progress in the execution phase and will help in providing the task contexts when decisions within the plan need to be made during execution.

Having compound tasks available as context during plan execution helps in determining the consequences of task execution by an asset that is different from default behavior. This is illustrated by an example involving a MOVE task of an asset that takes significantly more time than anticipated. If this MOVE is part of a surveillance operation, taking more time might not be problematic. On the other hand, in a scenario in which the MOVE is part of a coordinated SURPRISE ATTACK task involving multiple assets, this asset being late might render the SURPRISE ATTACK infeasible and even jeopardize the mission. This means that a new approach to an attack needs to be selected.

5.1.1. Input to the Planning Process

Before detailing the planning process itself, we describe the input to this process. We distinguish three elements in the input:

1. **Task templates.** The hierarchically structured templates for tasks are provided in the design phase by the designer (see Figure 1). These task templates and their

(hierarchical) relations ensure that a mission with a large time horizon, consisting of many tasks, can be planned comprehensively before mission execution. The hierarchy of task templates yields a mission-specific decomposition into subtasks.

2. **World model.** Information about the world and algorithms to reason over elements in the world are also provided by the designer (as can be seen in Figure 1). Examples of reasoning algorithms include methods for analyzing the environment, e.g., weather conditions and terrain information, for analysis of the enemy's possible courses of action, and for the selection of assets to execute certain subtasks, based on situation specific circumstances and/or asset capabilities.
3. **Mission parameters.** This situation specific information is provided by the operator, as shown in Figures 1 and 2. The mission parameter specification contains information about, e.g., the assets that are available for this mission, the type of mission for which a plan needs to generated, an area of operation, and information on expected threats and other contacts which might be present during the operation.

5.1.2. Task Relations

The task templates, and hence the tasks in the plan, are interrelated through three types of relations. The most apparent relation between tasks is that of hierarchy, which has been motivated and discussed earlier. The other two relations between tasks are:

1. **Causal dependencies:** Tasks relate to each other and to detected changes in the environment via events and conditions. These provide a natural mechanism to describe causal relations between tasks. Typically, the parent task in the tree of tasks is responsible for specifying what the conditions are for subtasks to be started, interrupted, or finished. Simple examples include parallel starting of multiple tasks or a set of tasks that need to be executed sequentially.
2. **Information propagation:** Reasoning components typically require (mission specific) input parameters that allow them to generate output. The input needs to be provided during the process of decomposition. Therefore, we allow tasks to explicitly share information via so-called task outputs which will be linked to task inputs, similarly to what was done in [31]. Tasks can only be decomposed into its subtasks if all inputs of the task are linked with outputs of other tasks. Such tasks are known as grounded tasks. The parent task of a task that requires input is responsible for providing that information, either directly or by creating a link with the output of another task.
   An example of passing on information involves an area of operation that needs to be split into regions, which is done using a reasoning method associated with a compound task. This region information in turn can serve as input to a task that identifies a strategy for surveying that region. This information can be already available in the planning phase, but it is also possible to share runtime data, a placeholder, which will be resolved during execution. An example of a runtime data source as task input is an observed hostile contact which requires follow-up in the plan, without knowing the location and other details about that contact in the preparation phase.

5.1.3. The Mission Planning Process

The complete plan hierarchy is constructed by recursively decomposing tasks into subtasks. During the process of decomposition, the following steps are taken:

1. Reasoning methods are triggered to generate/calculate required information.
2. The subtasks are instantiated, based on their respective task templates.
3. Input for the subtasks is provided, either directly or by linking task output from a task to a task input of another task. This allows compound subtasks to become grounded, which means that these tasks can be decomposed themselves. Ultimately, the propagation of input information results in basic task parameters that are required for the execution of these tasks by (one of) the assets.
4. Causal relations between the subtasks are explicated by linking task events and external events to task conditions.

Note that in the current implementation, the reasoning that is required in the assessment module is directly available in that module, which results in a somewhat ad hoc design. The scope of a compound task could also provide a slot for reasoning methods that are required during execution and made available to the assessment module. This will be implemented in follow-up projects.

After decomposition of the task templates into a course of action, the compound and basic asset tasks in that plan contain all required information, and are linked via conditions and events. Pseudocode of the iterative decomposition of tasks, resulting in a plan, is provided in Algorithm 2, and the steps required for the decomposition of an individual task into subtasks are provided in Algorithm 3.

Algorithm 2 Plan construction.

```
1:  procedure CREATE PLAN(rootTask)
2:      tasksToDecompose ← {rootTask}
3:      plan ← {rootTask}
4:      repeat
5:          for groundedTask in grounded tasksToDecompose do
6:              tasksToDecompose.remove(groundedTask)
7:              newTasks ← DECOMPOSE(groundedTask)          ▷ See Algorithm 3.
8:              tasksToDecompose ← tasksToDecompose ∪ newTasks
9:              plan ← plan ∪ newTasks
10:         end for
11:     until decomposibleTasks = {}
12:     return plan
13: end procedure
```

Algorithm 3 Task decomposition.

```
1: procedure DECOMPOSE(task)
2:     Execute reasoning methods                                  ▷ Using task inputs.
3:     Instantiate subtasks                              ▷ Using template information.
4:     Set task input of subtasks   ▷ Using reasoning results and task input/task output.
5:     Set subtask conditions                  ▷ Using task events and external events.
6:     return subTasks
7: end procedure
```

As shown in Figure 2, the execution of the mission plan is performed by the SUPERVISOR module, together with a team of assets. We describe the responsibilities of the SUPERVISOR and its ASSESSOR and that of the TASK MANAGERS that are available in each asset in the next section.

5.2. Mission Execution: The SUPERVISOR *Module*

The SUPERVISOR component (briefly introduced as the SUPERVISOR module in Section 3.2.2) fulfills several core functions within the ICTUS framework:

- **Execution of the plan:** managing tasks, conditions, and events in the planned hierarchy. The SUPERVISOR implements the task execution process using a discrete event simulation. Starting from the top level task in the plan, all subtasks are processed according to their condition sets, as described in Section 4.3.
- **Messaging:** The SUPERVISOR is the central node that requests tasks from the worker assets and receives results and feedback. Messaging is done by a publish–subscribe messaging system on which all team members are subscribed. The interpretation and content of task-related messages are outlined below; note that the precise technical implementation is not essential for ICTUS. In addition to this, all messaging with the worker assets is done and received information is passed to the (mission-specific)

ASSESSOR module. The assessment leads to triggering of events to move forward in the plan execution.

- **Assessment:** Working together with the ASSESSOR and a data store for runtime information (the blackboard), the SUPERVISOR manages and acts upon relevant information about mission progress and results. As the complete planned hierarchy is known during the entire execution, mission progress can be closely logged and monitored by the SUPERVISOR. In future systems, this can be exploited to give operators insight in mission progress both during and after execution.

5.2.1. Messaging

To coordinate the interaction between worker assets and the SUPERVISOR, the ICTUS framework defines a set of message types. The most important messages are shown in Figure 7.

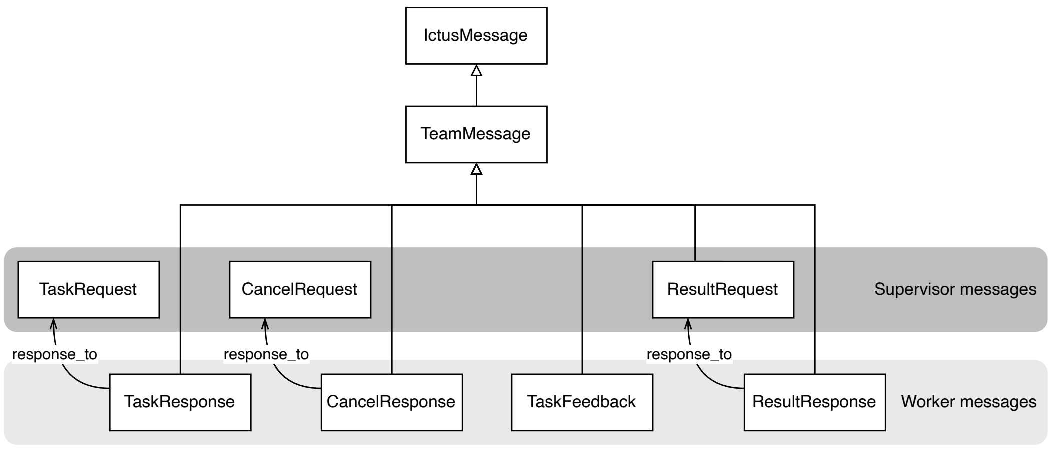

Figure 7. An overview over the most relevant messages that are exchanged between the SUPERVISOR and workers' TASK MANAGER in the ICTUS system.

When the SUPERVISOR wants to dispatch a task to a worker, it sends a TASKREQUEST for a specific task type with appropriate task parameters. In response, the TASK MANAGER sends a TASKRESPONSE either accepting or rejecting the task. When accepted, the task is executed, and the TASK MANAGER sends FEEDBACK messages with information on task progress. When the task finishes (successfully or not), a RESULTRESPONSE with result info is sent to the SUPERVISOR. The actual contents of request, feedback, and result messages are task-specific and will depend on the type of asset and its capabilities. The described interaction between SUPERVISOR and TASK MANAGER is summarized in Figure 8.

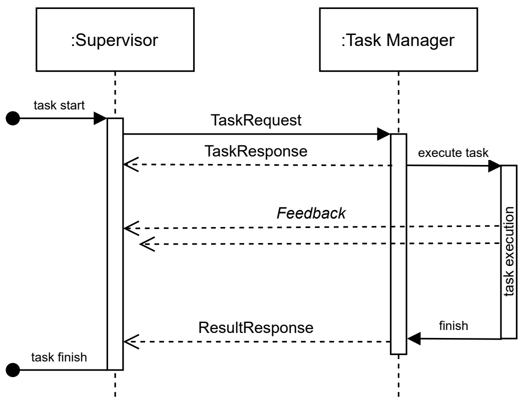

Figure 8. The interaction between the SUPERVISOR and a TASK MANAGER for the execution of a task.

5.2.2. The Assessor Module

The Assessor module is responsible for the execution-time data handling and the assessment of the plan and the situation. The concept of the Assessor is best illustrated with a relatively simple example, which involves the following behavior:

Several drones search an area for an object. When the object is spotted (which relates to an `observation` event in the plan), the spotter drone (an asset deployed to perform reconnaissance) stops searching and starts to hover above the object. The other drones keep searching. This seems and should be straightforward to plan beforehand, but at plan time, when putting a HOVER task into the plan, it is not yet known which of the drones will be the one spotting the object. That information will not be available until during the execution of the SEARCH tasks.

To achieve the correct behavior, we introduced runtime data as a mechanism to resolve this. In the plan, the HOVER task is assigned to a placeholder drone which we refer to as spotter (a unique data identifier in the plan that refers to a data value that will be determined during execution). This is combined with an Assessor method that—during execution—receives feedback data from the SEARCH tasks. When the observation feedback matches with the object that was searched for, it can easily determine the actual drone that was identified with spotter in the plan and put that information on the blackboard. The Assessor subsequently triggers the observation event that is present in the plan. The Supervisor is then able to combine the placeholder with the information on the blackboard and dispatch the HOVER task to the correct drone. This mechanism can be extended to more involved reasoning methods to assess and determine runtime information and plan-specific events. Note that the assessment methods become an integral and essential part of the plan object. This is the reason that in ICTUS the Supervisor/Assessor module has been separated into two parts. The Supervisor is doing the generic execution and communication processing, and the Assessor, which is plan specific, contains filtering and decision logic to process outside data into plan-relevant, actionable information.

5.3. Mission Execution: The Task Manager *Module*

As briefly introduced in Section 3.2.3, the Task Manager is a module that is deployed on each autonomous asset. As discussed, it interacts with the Supervisor by receiving information on the execution of tasks and by providing the Supervisor with task results, feedback on task execution, and sensor information. The responsibility of the Task Manager is to have the asset properly execute the tasks it receives, either individually or in cooperation with other assets. In other words, the Task Manager ensures that the correct behavior is being performed, by sending commands to the different modules and payloads of the asset. For example, if a MOVE task is received, this will result in triggering path planning and subsequently commands a navigation controller. In addition, the Task Manager provides information to the Supervisor (and hence Assessor) to allow for team-level situation awareness and understanding, and keeping track of task and mission progress, making sure that the mission goals will be achieved.

When a Task Manager receives task information from the Supervisor, it first checks whether or not the asset is able to perform the received task (in the current circumstances) and will start executing that task if this is indeed the case. Every asset has a set of tasks it can execute, and assets can exhibit different behaviors for the execution of the same task. For example, a task might require the presence of a certain payload (e.g., a sensor or weapon) on the asset. In addition, a MOVE task of a fixed wing drone will result in different behavior from that of a tracked vehicle. Tasks that are typically not available for all autonomous assets are LAUNCH and DOCK tasks, MOVE tasks, and WAIT tasks.

If a Task Manager is not executing a task, this will typically result in the same behavior as an explicit WAIT task. But if the task cannot be executed, or if at some moment the task execution fails, the Task Manager informs the Supervisor by sending a TaskResult message with status failed. If the Task Manager is requested to cancel the task which it is currently executing, it will (gracefully) stop that task and confirms it is stopping by informing the SUPERVISOR. To provide some insight in asset behavior that is associated with task execution by its TASK MANAGER, we give a few examples:

- Consider as asset a UGV, capable of moving covertly through a terrain with forests and open fields. If its TASK MANAGER receives a MOVE task that needs to be executed covertly, the terrain is analyzed to determine a path on which the probability of detection by other (non-friendly) assets is minimized. At the same time, the covert path length should not be much larger compared to a shortest path. Concretely, the path will be such that the asset is close to forest edges and/or through shrubbery.
- During the execution of a FOLLOW task, an asset will first execute a move towards to object to follow (if it is not yet in sight). After the object is observed, the asset will reposition itself (regularly) if needed to make sure the object will remain in sight.
- If an ENGAGEMENT task is received, the asset's goal is to eliminate a target, by using one or more of its weapon systems. Engagement continues until the target is no longer a threat, or if the target is no longer within line of fire or line of sight. If the target is eliminated, the task was executed successfully; otherwise, the task is considered to have been failed. Note that the execution of the ELIMINATE task involves not only repeatedly actuating weapons, but also the processing of sensor information.

As mentioned before, information involving the progression of tasks, the state of the asset and its perception of the world are shared with the SUPERVISOR during the mission, and, if desired, with other assets. For example, during the execution of a move task the intended path, the remaining distance and the estimated time of arrival can be shared using FEEDBACK messages. Another use for FEEDBACK messages involves sharing observations of other (non-team) assets with the SUPERVISOR and other TASK MANAGERS. Communication between TASK MANAGERS allows the SUPERVISOR to send a task to multiple assets that require alignment during it execution. For instance, the search for suspect contacts or loitering ammunition are tasks that can be executed with multiple assets that work together, without the need for detailed coordination from the SUPERVISOR.

6. Implementation and Demonstration

6.1. Implementation

The structure of the plan, its construction by the PLANNER, and the coordinated execution by the SUPERVISOR and asset TASK MANAGERS were implemented in `Python`, according to the architecture as given in Figure 9. The TASK MANAGER was implemented as a `ROS2` node. Python packages that were used included `SimPy` as the discrete event simulation framework, `pyzmq` for messaging, and `NetworkX` and `Matplotlib` for visualization of task templates and elements of the plan. The execution of the basic tasks by the TASK MANAGERS are simulated within the game engine `Unity`.

The interactions between the SUPERVISOR and TASK MANAGERS were implemented using a publish–subscribe messaging system to transfer ICTUS messages. For the communication between the TASK MANAGERS and the physical representation of the assets in the `Unity` scene, a connection via `ROS2` was used.

Figure 9 shows the higher-level architecture of the implementation, that is, the main components and the connections between these components. The inputs to the PLANNER include the operator-provided mission parameters, together with the in the design-phase-determined mission templates and world models. The inputs contain information on the military doctrine, including a hierarchy of tasks and reasoning capabilities on, e.g., terrain and enemy behavior. The output of the PLANNER is the plan, which consists of tasks (which in turn can contain subtasks). The SUPERVISOR receives the plan and has an execution mediator, called the *Executor*, which allows it to interact with one or more TASK MANAGERS via messages containing tasks. The TASK MANAGERS interact with the physical representation of the assets in the simulation environment, the `Unity` scene. Information that the TASK MANAGERS share with the SUPERVISOR, summarized here as *Task feedback*, is forwarded by the *Executor* to the ASSESSOR, which uses this information to perform *Situation assessment* and *Plan Assessment*. This assessment of "what is observed" and "what is achieved" results in runtime data that are shared via the *Blackboard* object.

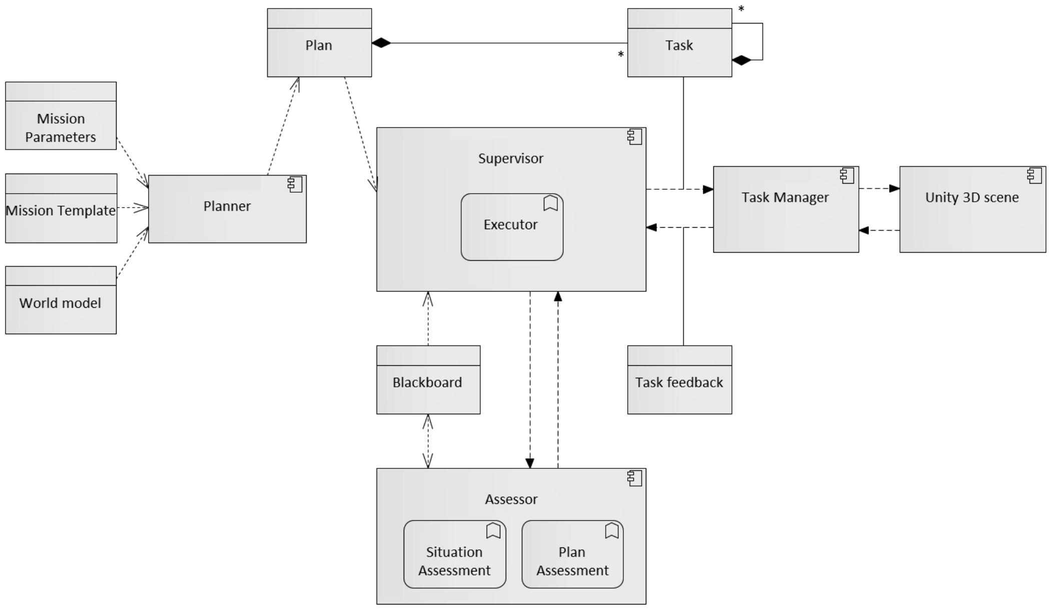

Figure 9. The ICTUS system architecture that has been implemented. The plan object specifies, for each platform, the actions and the order in which they are to be performed, see Figure 10.

6.2. Demonstration

6.2.1. Demonstration: An Area Defense Operation

To demonstrate the ICTUS approach towards autonomous planning and execution of tactical missions using autonomous assets, an area defense operation is considered as use case. For this use case, a `Unity` scene was setup. The terrain in this scene contains forests that are considered non-traversable, bushes that provide concealment, and a river. Roads and bridges provide ways to navigate through forests and to cross the river. See Figure 11 for a 2D (top-view) and 3D visualization of the generated terrain.

The *blue* team of autonomous assets encompasses four UGVs, which are all equipped with heavy machine guns. The objective of the defense operation is to deny the opposing forces access to the area west to (in Figure 11, to the left of) the river. In addition to the UGVs, a unmanned aerial vehicle (UAV) (or, drone) searches for opposing forces and shares its observations of contacts with the assets and SUPERVISOR. The observations of the UAV are used for timely regrouping of the UGVs.

To cross the river, there are three bridges available in the terrain, which create so-called chokepoints that need to be passed by the opposing, *red*, team, in order to reach its objectives. The red team consists of two vehicles, which can either cross the same bridge (simultaneously) or cross two different bridges (either simultaneously or shortly after one another). Thus, the red course of action involves deciding on when to cross which bridge or bridges with how many vehicles. This means that a total of six plans are available: in three cases, both vehicles aim to cross the same bridge and in three cases two bridges are approached (by one vehicle each).

The vehicles of the red team each represent a small platoon, and it is assumed that two blue UGVs, at a concealed position, are able to deflect and/or eliminate a red vehicle. Equivalently, four UGVs are required for countering an attack executed by two red vehicles.

Mission templates for the required tasks and reasoning were implemented. Both the blue forces and the opposing red forces use an ICTUS plan object, a SUPERVISOR module, and TASK MANAGER modules for their respective missions.

6.2.2. The Mission Plan

The mission plan is such that its execution results in the following behavior of the assets in the blue team:

- First, three UGVs position themselves at so-called *post* locations, close to the three bridges. A fourth vehicle positions itself at a *reserve* location, which allows it to move covertly and quickly on any of the three "post" locations (see the bottom right picture in Figure 11). In parallel, the UAV searches the area east of the river. The post locations are such that they provide concealment while ensuring that any contact approaching one of the three bridges will be observed.
- The three UGVs continuously scan their respective sectors, while receiving updates on locations of contacts—that is, red vehicles—from the UAV. When the ASSESSOR establishes which bridge(s) will be crossed, the UGVs are requested to regroup themselves as follows: If a single red vehicle is approaching a bridge, one UGV is tasked to help in the defense. If the UGV at the reserve location is available, this asset is involved. If in parallel (or shortly after) a second red vehicle is approaching another bridge, the UGV that is guarding the remaining (third) bridge will be requested to join the UGV at the post near that second bridge. If, on the other hand, both red vehicles are approaching the same bridge, all four UGVs position themselves at the post near the soon-to-be-crossed bridge. The (early) detections of the UAV ensure that the UGVs have ample time to adjust their positions before the red vehicles are within firing range.
- The red vehicles aim to enter blue territory via the bridges, and a fight between blue and red takes place. After a successful repulsion of an attack, the UGVs distribute themselves again. The three posts near the bridges are populated first, and if a fourth asset is available, which means that there are no losses for the blue team, this remaining asset will position itself at the reserve location.

A visual representation of a key part of the blue mission plan is given in Figure 10. In this figure, tasks are represented by rounded shapes. The horizontal ordering of tasks indicates causal dependencies between tasks, reading from left to right. At the top of the figure, the compound tasks are given. These tasks are executed by the SUPERVISOR. The box around the three ELIMINATE "X" tasks indicates that these tasks start in parallel, and in this case, directly after the start of DEFEND POSITION NORTH task. Directly below the compound tasks are so-called swim lanes located, one for each asset. In the swim lanes, it is indicated which basic tasks are executed. Tasks that are partially white indicate that during planning it is not yet known which asset will execute that task: the asset corresponding to that swim lane serves as an example asset. The bottom part visualizes the external events that might be triggered during execution by the ASSESSOR module. Arrows pointing from these events to tasks indicate that a task starts if the external event is triggered.

When a red vehicle is observed and it is established which bridge is approached, the ASSESSOR triggers the right `Attack` event, and an eliminate phase is started. The ELIMINATE FROM POST and ELIMINATE BY HELPER tasks start directly. This results in the repositioning, a MOVE to the post near the bridge, of an—at runtime—determined asset, referred to as the *helper*. If two vehicles will approach that same bridge, the `Backup` event is triggered. This means that the ELIMINATE BY BACKUPS starts and that the two remaining blue assets, the backups, will also position themselves near the bridge. Since at most two red vehicles are present, the backups can (temporarily) leave their respective post locations.

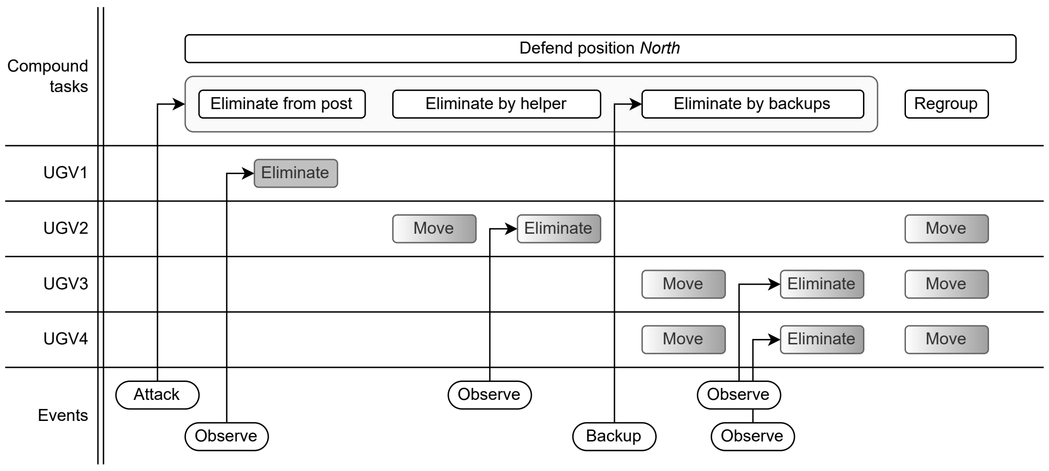

Figure 10. A part of the plan for the blue area defense operation: the defense phase involving one of the three bridges is given.

The asset that occupies the post and the repositioned assets wait until they observe the red vehicles, and as soon as these vehicles are within weapon range, the blue assets execute a (repeating) ELIMINATE task. The ASSESSOR assigns the target vehicle using runtime data and the SUPERVISOR sends the task with correct parameters to the assets. The ELIMINATE task continues until the asset itself is eliminated, if the target vehicle is eliminated or if the target vehicle is no longer visible and/or within attack range. Since the vehicles do not retreat in the current red course of action, the latter situation does not occur.

6.2.3. Plan Execution with Simulated Assets

The execution of the blue and red mission plans is illustrated with six snapshots that were taken from the simulation and are given in Figure 11. We describe the rationale of the snapshots, reading left to right, top to bottom:

- In the top-left sub-figure, the four blue assets are moving towards their respective positions: the three post locations near the bridges and the reserve location. The red vehicles are on the far-east side of the terrain and start approaching the blue territory.
- After some time, the blue ASSESSOR has established—through the UAV's observations—that the bridges in the north and in the south will be approached by one red vehicle. The `Attack` events are triggered, and as a result, the SUPERVISOR instructs two of the blue UGVs to reposition themselves. In the top-right sub-figure, the asset at the reserve location has joined the asset in the north, and the asset that guarded the bridge in the center left its post location and is moving towards the southernmost bridge.
- In the middle two sub-figures, the encounter between red and blue near the bridge in the south can be seen. As soon as the red vehicle is observed by the blue assets, the ENGAGEMENT tasks start. The red vehicle returns fire, but by virtue of the concealment of the blue assets, its disadvantage is significant. For that reason, the red vehicle is unable to eliminate the blue assets and is eliminated.
- In the bottom-left sub-figure, the attack in the south has finished, and the encounter in the north is bound to happen. This is rendered in the visualization by graying out (the symbol of) the eliminated red vehicle.
- After the attack by red is deflected, the defense phase is finished, and the remaining blue assets redistribute themselves in the regroup phase of the plan, cf. Figure 10.
- In the bottom right sub-figure, the blue assets have returned to their starting positions.

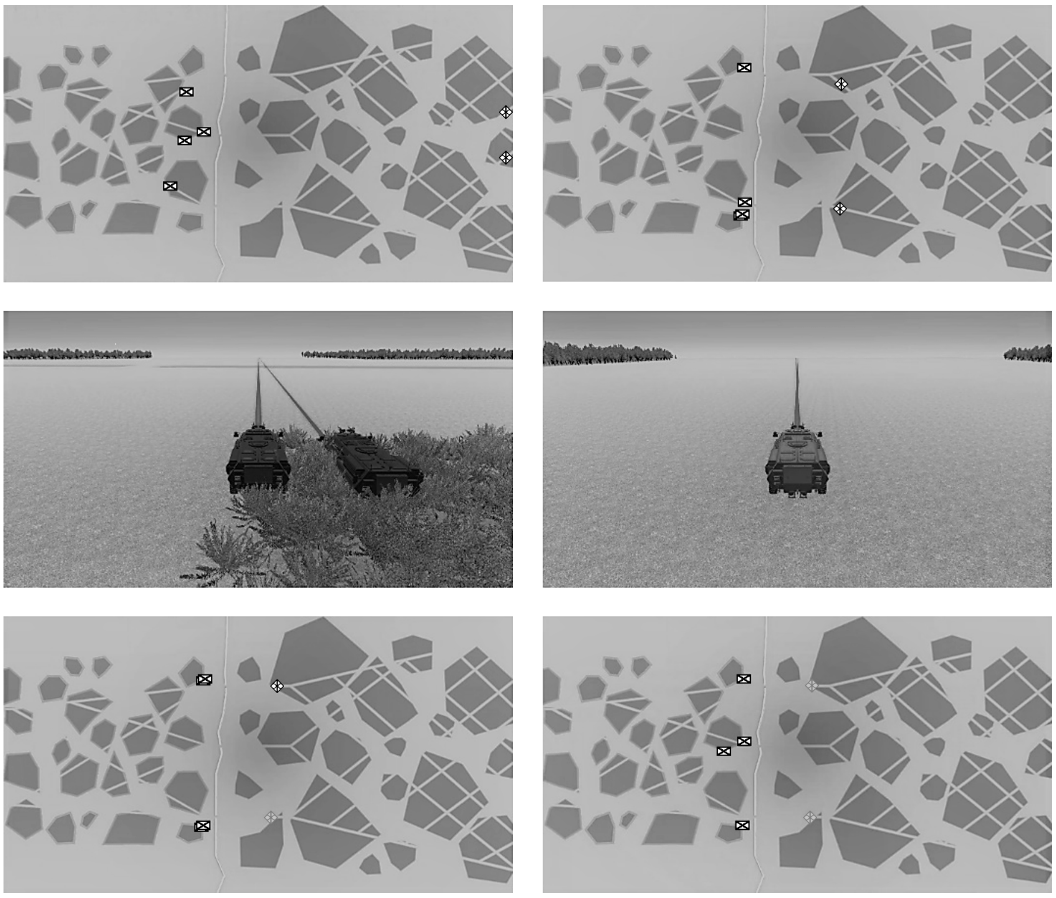
Figure 11. A few snapshots of the simulation involving the mission plan execution of the blue team.

7. Conclusions and Future Work

In the previous section, we discussed the implementation of our work and showed screenshots from the demonstrator to indicate its behavior. In this final section, we conclude the paper by highlighting the most important achievements of our work. We also suggest directions for future work and briefly discuss the proverbial road ahead.

7.1. Summary

As stated, the reported work was driven by practical needs and undertaken in close collaboration with the NLMinDef. A generic approach towards autonomous tactical mission planning and execution to control the operation of UGV has been developed.

As discussed in the opening of the manuscript, mission planning in the military domain has to meet certain requirements (**R**, stated in Section 2.2) and satisfy important constraints (**C**, see Section 2.3): planning has to be done for executable missions in dynamic environments (**R1**) and by providing a set of choices (plans) for final decision making by a human decision maker (**R2**). This means that the offered plans must be both short and concise to enable the human decision maker to make an informed choice (**R3**).

These requirements lead to four constraints: a plan must be such that it can be executed in a dynamic environment (**C1**) and by a number of platforms working as a coordinated team (**C2**). The platforms may only operate within a given doctrine; their behavior must be predictable and accountable (**C3**). Their deployment and the decision on which mission to deploy them for must be subject to human approval (**C4**).

These requirements and constraints were met. See Section 7.2 for a discussion.

Core elements of the approach involve the notion of an iteratively decomposed hierarchy of tasks, inspired by the HTN formalism (see Section 2.4.2), and an event driven approach towards execution, based on DES (see Section 2.5).

As a key distinction, our approach does not require replanning. Instead of having to return to the drawing board, the hierarchy of tasks offered by our system is equipped with sub-plans for (foreseen) contingencies.

The feasibility of the approach was demonstrated successfully in a simulation using the implemented use-case for a defense operation. This relied on the developed reasoning modules and the modeled mission templates. A hierarchical mission plan was constructed, providing tasks and subtasks for the mobile platforms during mission execution. The (sub)tasks specified the desired team and asset behaviors. Using military doctrine, the acceptable reactions to specific anticipated situations or events were defined. Contained within the plan were *contingencies* for these anticipated events. The resulting plan constituted a flexible blueprint for the operation that is suitable for a wide range of possible variations of circumstances and capable of defining collective actions for multiple platforms.

7.2. Revisiting the Constraints

The presented approach and its implementation were designed and developed with the aforementioned constraints (cf. Section 2.3) and requirements (cf. Section 2.2), in mind. Generally speaking, we argue to have met these constraints as follows:

- **C3 (accountability):** the design phase provides a mechanism (a doctrine model) to ensure that military doctrine is always followed by the unmanned systems.
- **C3 (predictability):** To further insure predictable behavior in line with the rules of engagement, mission templates and reasoning modules (determined in the design phase) can, and should, be constructed under the guidance of a subject-matter expert.
- **C1 (planning for dynamic environments):** The construction of plans, based on design-time determined templates is such that plans can span a long period of time, whereas the automated planning process does not take a long time.
- **C4 (human approval):** This enables feedback from the human operators in advance, allowing them to understand the behaviors of the unmanned systems and to adjust mission parameters and ultimately provide consent for the execution of the plan.
- **C1 (execution in dynamic environments):** Mission execution can take place even when external conditions have changed. The approach was specifically designed for dynamic environments: the availability of contingencies and placeholders (runtime data) in the plan, together with the event based execution of the plan, allows for incorporating and responding to information that becomes available during execution.
- **C2 (cooperative execution):** The hierarchically structured plan and the explicit causal relations between tasks in that plan enable coordinated and collaborative execution of the mission plan. In addition, including the execution of the higher-level tasks, in parallel with the basic tasks, helps coordinating the autonomous systems and keeping track of overall mission progress.
- **C2 (cooperative execution):** The unmanned systems execute their assigned tasks individually, and team level behavior, that is, coordinated and collaborative execution of the overall mission, is guaranteed.

7.3. Conclusions

We can conclude that the presented approach is capable of constructing plans and contingency plans for unmanned vehicles, allowing for the execution of a plan on the basis of perceived changes in the environment. One of the main achievements of this work is that the plan, constructed before deployment, was versatile enough to enable a group of autonomous vehicles to operate together despite that neither the details of the environment nor the exact behavior of an opposing force were known in the preparation phase.

Equally important was the ability to combine a smart-structured high-level approach with specific guidelines and protocols for the handling of the individual tasks. This combines the military's approach to have standard responses to standard situations with the flexibility required to make in situ decisions based on the situation at the time. Due to this, the behavior of the team of autonomous vehicles will be expected and contextually understood by the human operator, while leaving the team with the ability to tailor their joint behavior to the situation on the ground. While a human operator will not be able to predict reliably what an autonomous vehicle will do once deployed, they can predict with absolute certainty how any vehicle would react to any specific set of circumstances. This combines the power of autonomous assets with the hard requirement to understand and contain the behavior before it occurs.

7.4. Future Work

While many of the challenges related to autonomous or semi-autonomous weapon systems are outside the scope of our expertise (and mandate), there is no shortage of next steps that involve practical engineering and are well within the scope of our work. Some of these have already been mentioned in the article itself and are considered for projects in the near future (or already ongoing); others are more long-term plans. Among these are:

- In the mission-preparation phase, the plan is constructed based on the iterative decomposition of tasks into subtasks. The current process of decomposition uses the provided mission parameters to tailor the plan to the specific situation, but does not include an optimization towards, e.g., measures of effectiveness for acting in that situation. Including an optimization process is certainly part of the future work that will be implemented in follow-up projects.
- In Section 5.1.3, we pointed out that the reasoning in the assessment module is assumed to be directly available in that module itself. Extending the concept of a task such that it can be used to provide this information to the module is considered (and ongoing), but it will require tuning to the specific application, and thus our preliminary insights are omitted here.
- Providing feedback to, and more generally, involving human operators during the execution of the mission by the autonomous vehicles, will receive more attention in future projects. As mentioned in Section **??**, information on the current state of the unmanned systems, and their perceived information, together with the assessed situation and progress of the mission, are already available, and this information can be shared with operators. In addition, explicit consent from human operators for the execution of certain tasks during the mission might be required in specific cases and hence should be provided for.
- More broadly speaking, there is ample room for additional and more fleshed-out scenarios. We can go a number of ways in that, including (a) more advanced terrain models, possibly including new features; (b) increasing the number of vehicles and vehicle classes, and their capabilities; or (c) increasing the complexity of the modelled rules of engagement. None of these extensions would require fundamental changes in the developed system, but any of them means a significant increase in the time required to build the reasoning component.
- We have conceded (see Section 1.1) that we cannot offer a performance evaluation or a comparison of our approach to the state of the art on the basis of numerical results. This is due to the specialized nature of our work and because implementing reference approaches for bench marking and comparison is not feasible (in part because we aim at a different functionality, namely, one that can do without replanning). We do understand, however, that in the long run, the approach has to be further refined, amended, and augmented with the state of the art. While we do not claim to plan this for our own immediate future work, we do acknowledge the need for a more formal performance evaluation and encourage the interested reader to contact us.
- Finally, as pointed out in Section **??**, the abstract nature of our work allowed us to gloss over the details regarding the communication for tactical operations. Suffice it to say that a lot of efforts are directed at this challenge; and indeed, current operations of the RNLA and the Royal Netherlands Navy (RNLN) make use of state-of-the-art communication infrastructure. The approach presented here will operate within this

existing infrastructure. As this is a fundamental aspect of the system, these details will have to be discussed and presented to the field in due time.

Author Contributions: Writing—original draft, J.d.G., J.B. and H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The work presented here is part of ongoing larger efforts, building on previous projects and programs. Our work is to be seen in the context of, and would not be possible without, the work of our colleagues at TNO and that of our collaborators and customers. We would like to specifically mention Cdr. P. Blank, an innovator at the RNLN, who suggested the name ICTUS.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Broad, W.J. The U.S. Flight from Pilotless Planes. *Science* **1981**, *213*, 188–190. [CrossRef] [PubMed]
2. Hildmann, H.; Kovacs, E. Review: Using Unmanned Aerial Vehicles (UAVs) as Mobile Sensing Platforms (MSPs) for Disaster Response, Civil Security and Public Safety. *Drones* **2019**, *3*, 59. [CrossRef]
3. Dousai, N.M.K.; Loncaric, S. Detection of Humans in Drone Images for Search and Rescue Operations. In Proceedings of the 2021 3rd Asia Pacific Information Technology Conference (APIT 2021), Bangkok, Thailand, 15–17 January 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 69–75. [CrossRef]
4. Mandirola, M.; Casarotti, C.; Peloso, S.; Lanese, I.; Brunesi, E.; Senaldi, I. Use of UAS for damage inspection and assessment of bridge infrastructures. *Int. J. Disaster Risk Reduct.* **2022**, *72*, 102824. [CrossRef]
5. Nooralishahi, P.; Ibarra-Castanedo, C.; Deane, S.; López, F.; Pant, S.; Genest, M.; Avdelidis, N.P.; Maldague, X.P.V. Drone-Based Non-Destructive Inspection of Industrial Sites: A Review and Case Studies. *Drones* **2021**, *5*, 106. [CrossRef]
6. Bucknell, A.; Bassindale, T. An investigation into the effect of surveillance drones on textile evidence at crime scenes. *Sci. Justice* **2017**, *57*, 373–375. [CrossRef]
7. Mohd Daud, S.M.S.; Mohd Yusof, M.Y.P.; Heo, C.C.; Khoo, L.S.; Chainchel Singh, M.K.; Mahmood, M.S.; Nawawi, H. Applications of drone in disaster management: A scoping review. *Sci. Justice* **2022**, *62*, 30–42. [CrossRef]
8. Saffre, F.; Hildmann, H.; Karvonen, H.; Lind, T. Self-Swarming for Multi-Robot Systems Deployed for Situational Awareness. In *New Developments and Environmental Applications of Drones*; Lipping, T., Linna, P., Narra, N., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 51–72.
9. Sabino, H.; Almeida, R.V.; de Moraes, L.B.; da Silva, W.P.; Guerra, R.; Malcher, C.; Passos, D.; Passos, F.G. A systematic literature review on the main factors for public acceptance of drones. *Technol. Soc.* **2022**, *71*, 102097. [CrossRef]
10. Wu, H.; Wu, P.F.; Shi, Z.S.; Sun, S.Y.; Wu, Z.H. An aerial ammunition ad hoc network collaborative localization algorithm based on relative ranging and velocity measurement in a highly-dynamic topographic structure. *Def. Technol.* **2022**. [CrossRef]
11. Husodo, A.Y.; Jati, G.; Octavian, A.; Jatmiko, W. Switching target communication strategy for optimizing multiple pursuer drones performance in immobilizing Kamikaze multiple evader drones. *ICT Express* **2020**, *6*, 76–82. [CrossRef]
12. Calcara, A.; Gilli, A.; Gilli, M.; Marchetti, R.; Zaccagnini, I. Why Drones Have Not Revolutionized War: The Enduring Hider-Finder Competition in Air Warfare. *Int. Secur.* **2022**, *46*, 130–171. [CrossRef]
13. Aksu, O. *Potential Game Changer for Close Air Support*; Joint Air Power Competence Centre: Kalkar, Germany, 2022; Volume 3.
14. Mgdesyan, A. Drones A Game Changer in Nagorno-Karabakh. *Eurasia Rev.* **2020**, *2*.
15. Lesire, C.; Bailon-Ruiz, R.; Barbier, M.; Grand, C. A Hierarchical Deliberative Architecture Framework based on Goal Decomposition. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 9865–9870. [CrossRef]
16. Shriyam, S.; Gupta, S.K. Incorporation of Contingency Tasks in Task Allocation for Multirobot Teams. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 809–822. [CrossRef]
17. Ghallab, M.; Nau, D.; Traverso, P. The actor's view of automated planning and acting: A position paper. *Artif. Intell.* **2014**, *208*, 1–17. [CrossRef]
18. Peters, F.; Den Breejen, E. Integrating autonomous system of systems in the Royal Netherlands Navy. In Proceedings of the International Ship Control Systems Symposium, Delft, The Netherlands, 5–9 October 2020. [CrossRef]
19. Konert, A.; Balcerzak, T. Military autonomous drones (UAVs)—From fantasy to reality. Legal and Ethical implications. *Transp. Res. Procedia* **2021**, *59*, 292–299. [CrossRef]
20. Azar, A.T.; Serrano, F.E.; Koubaa, A.; Ibrahim, H.A.; Kamal, N.A.; Khamis, A.; Ibraheem, I.K.; Humaidi, A.J.; Precup, R.E. Chapter 5—Robust fractional-order sliding mode control design for UAVs subjected to atmospheric disturbances. In *Unmanned Aerial Systems: Theoretical Foundation and Applications*; Advances in Nonlinear Dynamics and Chaos (ANDC) Series; Koubaa, A., Azar, A.T., Eds.; Academic Press: Cambridge, MA, USA, 2021; pp. 103–128. [CrossRef]

21. Leonhard, R. *The Art of Maneuver: Maneuver Warfare Theory and Airland Battle*; Random House Publishing Group: New York, NY, USA, 2009.
22. von Clausewitz, C.; Werner, H. *Vom Kriege*; Nikol: Hamburg, Germany, 2011.
23. Sun, T.; Sawyer, R. *The Art of War*; Basic Books: New York, NY, USA, 1994.
24. Capek, K.; Novack-Jones, C.; Klima, I. *R.U.R. (Rossum's Universal Robots)*; Penguin Classics; Penguin Publishing Group: London, UK, 2004.
25. Takayama, L.; Nass, C. Beyond dirty, dangerous and dull: What everyday people think robots should do. In Proceedings of the HRI '08: International Conference on Human Robot Interaction, Amsterdam, The Netherlands, 12–15 March 2008; pp. 25–32. [CrossRef]
26. Goddard, M.A.; Davies, Z.G.; Guenat, S.; Ferguson, M.J.; Fisher, J.C.; Akanni, A.; Ahjokoski, T.; Anderson, P.M.L.; Angeoletto, F.; Antoniou, C.; et al. A global horizon scan of the future impacts of robotics and autonomous systems on urban ecosystems. *Nat. Ecol. Evol.* **2021**, *5*, 219–230. [CrossRef]
27. Haider, A.; Schmidt, A. *Defining the Swarm*; Joint Air Power Competence Centre: Kalkar, Germany, 2022; Volume 34.
28. Dorigo, M. SWARM-BOT: An experiment in swarm robotics. In Proceedings of the 2005 IEEE Swarm Intelligence Symposium (SIS), Pasadena, CA, USA, 8–10 June 2005; pp. 192–200. [CrossRef]
29. Ropero, F.; Muñoz, P.; R-Moreno, M.D. TERRA: A path planning algorithm for cooperative UGV–UAV exploration. *Eng. Appl. Artif. Intell.* **2019**, *78*, 260–272. [CrossRef]
30. Marvin, S.; While, A.; Kovacic, M.; Lockhart, A.; Macrorie, R. *Urban Robotics and Automation: Critical Challenges, International Experiments and Transferable Lessons for the UK*; EPSRC UK Robotics and Autonomous Systems (RAS) Network: London, UK, 2018. . [CrossRef]
31. der Sterren, W.V. *Hierarchical Plan-Space Planning for Multi-unit Combat Maneuvers*; Game AI Pro 360: Guide to Architecture; CRC Press: Boca Raton, FL, USA, 2013; Chapter 13; pp. 169–183.
32. West, A.; Tsitsimpelis, I.; Licata, M.; Jazbec, A.; Snoj, L.; Joyce, M.J.; Lennox, B. Use of Gaussian process regression for radiation mapping of a nuclear reactor with a mobile robot. *Sci. Rep.* **2021**, *11*, 13975. [CrossRef] [PubMed]
33. Beautement, P.; Allsopp, D.; Greaves, M.; Goldsmith, S.; Spires, S.; Thompson, S.G.; Janicke, H. Autonomous Agents and Multi-agent Systems (AAMAS) for the Military—Issues and Challenges. In *Proceedings of the Defence Applications of Multi-Agent Systems*; Thompson, S.G., Ghanea-Hercock, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–13.
34. Chithaluru, P.; Stephan, T.; Kumar, M.; Nayyar, A. An enhanced energy-efficient fuzzy-based cognitive radio scheme for IoT. *Neural Comput. Appl.* **2022**, *34*, 19193–19215. [CrossRef]
35. Chithaluru, P.; Al-Turjman, F.; Kumar, M.; Stephan, T. Computational Intelligence Inspired Adaptive Opportunistic Clustering Approach for Industrial IoT Networks. *IEEE Internet of Things J.* **2023**, 1. [CrossRef]
36. Hunjet, R. Autonomy and self-organisation for tactical communications and range extension. In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–5. [CrossRef]
37. Van Wiggen, O.; Pollaert, T. *Tactics, Made Easy*; Onkenhout Groep: Diemen, The Netherlands, 2009.
38. Ghallab, M.; Nau, D.; Traverso, P. *Automated Planning and Acting*; Cambridge University Press: Cambridge, UK, 2016. [CrossRef]
39. Campbell, M.; Hoane, A.; Hsiung Hsu, F. Deep Blue. *Artif. Intell.* **2002**, *134*, 57–83. . [CrossRef]
40. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
41. Hildmann, H. Computer Games and Artificial Intelligence. In *Encyclopedia of Computer Graphics and Games*; Lee, N., Ed.; Springer International Publishing: Cham, Switzerland, 2018; pp. 1–11. [CrossRef]
42. Lohatepanont, M.; Barnhart, C. Airline Schedule Planning: Integrated Models and Algorithms for Schedule Design and Fleet Assignment. *Transp. Sci.* **2004**, *38*, 19–32. [CrossRef]
43. Georgievski, I.; Aiello, M. HTN planning: Overview, comparison, and beyond. *Artif. Intell.* **2015**, *222*, 124–156. [CrossRef]
44. Mahmoud, I.M.; Li, L.; Wloka, D.; Ali, M.Z. Believable NPCs in serious games: HTN planning approach based on visual perception. In Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games, Dortmund, Germany, 26–29 August 2014; pp. 1–8. [CrossRef]
45. Fishman, G. *Discrete-Event Simulation: Modeling, Programming, and Analysis*; Springer Series in Operations Research and Financial Engineering; Springer: New York, NY, USA, 2013.
46. Bagchi, S.; Buckley, S.; Ettl, M.; Lin, G. Experience using the IBM Supply Chain Simulator. In Proceedings of the 1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274), Washington, DC, USA, 13–16 December 1998; Volume 2, pp. 1387–1394. [CrossRef]

47. Franke, J.; Hughes, A.; Jameson, S. Holistic Contingency Management for Autonomous Unmanned Systems. In Proceedings of the AUVSIs Unmanned Systems North America, Orlando, FL, USA, 29–31 August 2006.
48. MahmoudZadeh, S.; Powers, D.M.W.; Bairam Zadeh, R. State-of-the-Art in UVs' Autonomous Mission Planning and Task Managing Approach. In *Autonomy and Unmanned Vehicles: Augmented Reactive Mission and Motion Planning Architecture*; Springer: Singapore, 2019; pp. 17–30. [CrossRef]