Human-Machine Teaming with small Unmanned Aerial Systems in a MAPE-K Environment

JANE CLELAND-HUANG, Computer Science and Engineering, University of Notre Dame, IN, USA
THEODORE CHAMBERS, Computer Science and Engineering, University of Notre Dame, IN, USA
SEBASTIAN ZUDAIRE, Instituto Balseiro – Universidad Nacional de Cuyo, Argentina
MUHAMMED TAWFIQ CHOWDHURY, Computer Science and Engineering, Univ. of Notre Dame, IN, USA
ANKIT AGRAWAL, Dept. of Computer Science, Saint Louis University, MO, USA
MICHAEL VIERHAUSER, LIT Secure and Correct Systems Lab, Johannes Kepler University Linz, Austria

The Human Machine Teaming (HMT) paradigm focuses on supporting partnerships between humans and autonomous machines. HMT describes requirements for transparency, augmented cognition, and coordination that enable far richer partnerships than those found in typical human-on-the-loop and human-in-the-loop systems. Autonomous, self-adaptive systems in domains such as autonomous driving, robotics, and Cyber-Physical Systems, are often implemented using the MAPE-K feedback loop as the primary reference model. However, while MAPE-K enables fully autonomous behavior, it does not explicitly address the interactions that occur between humans and autonomous machines as intended by HMT. In this paper, we, therefore, present the *MAPE-K$_{HMT}$* framework which utilizes runtime models to augment the monitoring, analysis, planning, and execution phases of the MAPE-K loop in order to support HMT despite the different operational cadences of humans and machines. We draw on examples from our own emergency response system of interactive, autonomous, small unmanned aerial systems to illustrate the application of *MAPE-K$_{HMT}$* in both a simulated and physical environment, and discuss how the various HMT models are connected and can be integrated into a MAPE-K solution.

CCS Concepts: • **Human-centered computing → Collaborative interaction**; **HCI theory, concepts and models**.

Additional Key Words and Phrases: Self-Adaptive Systems, Human-Machine Teaming, Autonomous Systems, MAPE-K

ACM Reference Format:
Jane Cleland-Huang, Theodore Chambers, Sebastian Zudaire, Muhammed Tawfiq Chowdhury, Ankit Agrawal, and Michael Vierhauser. 2024. Human-Machine Teaming with small Unmanned Aerial Systems in a MAPE-K Environment. 1, 1 (July 2024), 34 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ANNOTATION FOR REVIEW PURPOSES:

Text shown in blue represents new or significantly edited text in this revision versus the originally submitted TAAS paper; while text shown in maroon represents a mix of modified and unmodified text within a paragraph of the revision.

Authors' addresses: Jane Cleland-Huang, Computer Science and Engineering, University of Notre Dame, IN, USA, JaneHuang@nd.edu; Theodore Chambers, Computer Science and Engineering, University of Notre Dame, IN, USA, tchambe2@nd.edu; Sebastian Zudaire, Instituto Balseiro – Universidad Nacional de Cuyo, Argentina, sebastian.zudaire@ib.edu.ar; Muhammed Tawfiq Chowdhury, Computer Science and Engineering, Univ. of Notre Dame, IN, USA, mchowdhu@nd.edu; Ankit Agrawal, Dept. of Computer Science, Saint Louis University, MO, USA, ankit.agrawal.1@slu.edu; Michael Vierhauser, LIT Secure and Correct Systems Lab, Johannes Kepler University Linz, Austria, michael.vierhauser@jku.at.

1 INTRODUCTION

The MAPE-K feedback loop has been broadly adopted as a reference model for controlling autonomous and self-adaptive systems [8, 74]. It has been used across diverse domains, including autonomous driving and traffic management [54], small Unmanned Aerial Systems [89, 94], Smart Home and IoT applications [7, 61], and assistive robots [68]. In the area of robotics, the MAPE-K control loop, and other similar frameworks, have enabled a shift from human-controlled robots to fully autonomous systems, capable of making independent decisions and (self-)adapting their own behavior according to their current state and their perception of the world around them. These systems are referred to as "Human-on-the-Loop" (HotL) [49] systems and take full advantage of machine autonomy to perform tasks independently, efficiently, and quickly. In contrast, traditional "Human-in-the-Loop" (HitL) systems primarily rely on humans to make decisions at key points in the system's execution.

In our work, we focus on a third paradigm referred to as *Human-Machine Teaming* (HTM) [77], in which machines, while capable of fully autonomous behavior, collaborate with other machines, as well as humans, to optimize task efficiency. Damacharia illustrated the benefits of HMT through an example of a chess game played in 2005 by a team of two inexperienced chess players and three PC machines against a less organized grouping of grand-masters and supercomputers [36, 59]. The HM team leveraged the cognitive skills of the humans and the data-mining abilities of the PCs, to outperform the less coordinated but clearly superior abilities of the grand-masters and supercomputers. The main purpose of HMT, as illustrated through this chess game, is to facilitate effective teamwork through emphasizing *interactions* and *partnership*, and through leveraging the *individual strengths of both humans and machines*, while compensating for their potential shortcomings and limitations [77, 87]. In effective Cyber-Physical HMT environments, the flexible, creative, and empathetic capabilities of humans are therefore combined with the processing power, data analysis, and physical capabilities of machines to achieve mission goals more efficiently.

HMT systems incorporate aspects of Cyber-Physical Systems (CPS) [120], Socio-Technical Systems (STS) [43, 131], and collaborative cognitition and decision-making [70, 79]. In the context of HMT, systems are still expected to operate fully autonomously, with all the capabilities that MAPE-K is designed to support. In fact, not only are the machines capable of performing their tasks autonomously, but they are perceived as true *partners* and not just "tools" in achieving mission goals. To facilitate this transition from the HotL paradigm to HMT, humans and machines need to interact more closely – not in a way that reduces or curtails the autonomous behavior of the machine, but in one that leverages that behavior to create meaningful partnerships where decisions are made collaboratively.

The primary goal of any feedback control system is to remove humans from the loop, and, therefore, MAPE-K tends to focus upon autonomous decision-making and self-adaptation rather than emphasizing the human aspects of a CPS. This was reflected in a recent systematic literature review [14] which reported that runtime models associated with self-adaptation primarily target the architecture, structure of the system, and/or its goals, but hardly incorporate any human-related factors or activities, such as user interaction or situational awareness [44]. This creates a gap between the existing MAPE-K framework and the capabilities necessary for an autonomous system to fully collaborate with human partners in an HTM environment. Kephart [73] proposed bridging this gap through creating highly interactive "dialogs" between humans and machines; however, he drew examples from information systems and not real-time robotic environments and, therefore, assumed that plenty of human "thinking" time would exist in the interactive dialogs. To the best of our knowledge, other prior work that discusses integrating humans into faster acting MAPE-K loops assume either a HiTL model, where the human is making all key decisions, or a HoTL model in which the human supervises the machine and decides when and where to intervene. Our HMT approach is far more interactive, as both

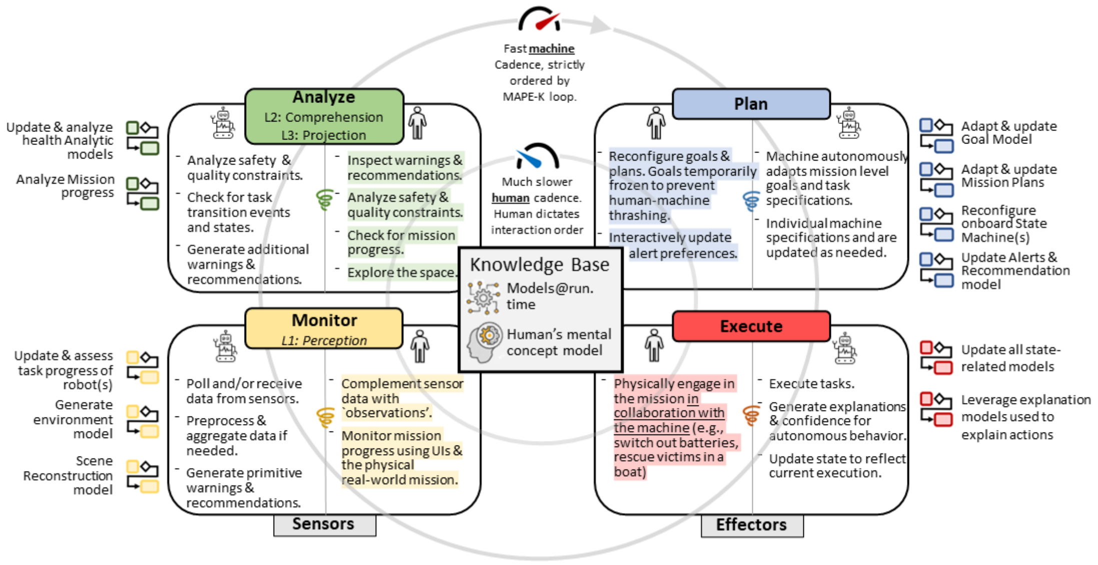

Fast machine Cadence, strictly ordered by MAPE-K loop.
Analyze
L2: Comprehension
L3: Projection
Update & analyze health Analytic models
Analyze Mission progress
Analyze safety & quality constraints.
Check for task transition events and states.
Generate additional warnings & recommendations.
Inspect warnings & recommendations.
Analyze safety & quality constraints.
Check for mission progress.
Explore the space.
Much slower human cadence. Human dictates interaction order
Plan
Reconfigure goals & plans. Goals temporarily frozen to prevent human-machine thrashing.
Interactively update alert preferences.
Machine autonomously adapts mission level goals and task specifications.
Individual machine specifications and are updated as needed.
Adapt & update Goal Model
Adapt & update Mission Plans
Reconfigure onboard State Machine(s)
Update Alerts & Recommendation model
Knowledge Base
Models@run.time
Human's mental concept model
Monitor
L1: Perception
Update & assess task progress of robot(s)
Generate environment model
Scene Reconstruction model
Poll and/or receive data from sensors.
Preprocess & aggregate data if needed.
Generate primitive warnings & recommendations.
Complement sensor data with 'observations'.
Monitor mission progress using UIs & the physical real-world mission.
Sensors
Execute
Physically engage in the mission in collaboration with the machine (e.g., switch out batteries, rescue victims in a boat)
Execute tasks.
Generate explanations & confidence for autonomous behavior.
Update state to reflect current execution.
Update all state-related models
Leverage explanation models used to explain actions
Effectors

Fig. 1. An overview of MAPE-K$_{HMT}$ showing machine activities (outer circle), and human activities (inner circle) [34]. Phases are mapped to Situational Awareness levels (L1-L3). Examples of runtime models are shown for each phase. The knowledge base is supported by HMT-models, and augmented by each humans' runtime conceptual model of the current state and its capabilities. The figure is borrowed, with minor modifications from [34].

parties (human and machine) request help from the other based on their current beliefs of the situation, their own uncertainty, and their knowledge of the other partners' capabilities. Therefore, each partner must monitor and analyze each others' behavior, and make contextualized decisions according to this knowledge.

In our earlier paper [34], published at SEAMS 2022, we proposed MAPE-K$_{HMT}$ as a way to align *teaming factors*, as described in the HMT literature [92], with the different phases of the MAPE-K loop in order to support HMT. We presented examples of runtime models for use in MAPE-K$_{HMT}$ and described their support for bidirectional human-machine interactivity and decision-making. These examples were taken from our own multi-agent system of autonomous small Unmanned Aerial Systems (sUAS) for supporting emergency response missions [33, 42]. Finally, based on these examples, we presented a lightweight, reusable process for integrating HMT into MAPE-K. In this journal paper, we extend our prior work in several important ways. We introduce a clearer distinction between traditional models@runtime and our proposed HMT models. Models@runtime provide direct reflection mechanisms between runtime models and the underlying system [14, 88], often supporting runtime activities such as verification or analysis. In contrast, while a causal relationship still exists between HMT models and the underlying system, it is less direct and reflects the broader relationship between the system, models, and humans. We have also extended the description of our case system, to clearly describe its use of MAPE-K and its implementation of a managed system at two different levels – first as a traditional autonomous system onboard each sUAS, and second at the ecosystem level where humans partner interactively with multiple sUAS. The core extension is in the form of an extended set of models for supporting HMT teaming factors in real-world scenarios, where each new model emphasizes the bidirectional nature of the HMT partnership. For example, one of our new models enables sUAS to understand the current cognitive load of humans in order to support their own autonomous adaptation and decision-making processes, while another new model supports collaborative decision-making between humans and sUAS and tackles more complex cases in which disagreements occur. We additionally show how a set of HMT models can be combined, creating an integrated HMT-coordination plan to guide *bi-directional* monitoring decisions, form pipelines that enable models to share inputs, and to generally improve integration efficiency. Finally, several of the MAPE-K_{HMT} models described in this paper have been deployed in either our high-fidelity simulator and/or in physical-world deployments with teams composed of multiple users and multiple sUAS. We, draw from these experiences to provide real-world insights into HMT challenges and solutions.

The remainder of this paper is structured as follows. Section 2 introduces *MAPE-K_{HMT}* and its extensions to the MAPE-K loop. Section 3 introduces our case study system, while Section 4 presents nine examples of HMT-related runtime models and their integration into *MAPE-K_{HMT}*. Section 5 then briefly discusses an approach for eliciting domain-specific requirements, and testing and deploying *MAPE-K_{HMT}* in a MAPE-K system. Finally, Sections 6 to 8 present threats to validity, related work, and conclusions.

2 AUGMENTING THE MAPE-K LOOP TO SUPPORT HMT

The involvement of humans in the context of the MAPE-K feedback loop so far has mainly focused on either monitoring human behavior [25], notifying users [82], or triggering actions/adaptations based on input from the user, e.g, through novel types of user interfaces [136]. The intention of our *MAPE-K_{HMT}* framework is to incorporate all aspects of MAPE-K and to support human engagement in decision-making and task enactment, while simultaneously preserving the autonomy of the self-adaptive system. The literature describes three goals of transparency, augmented cognition, and coordination that must be achieved in effective HMT environments.

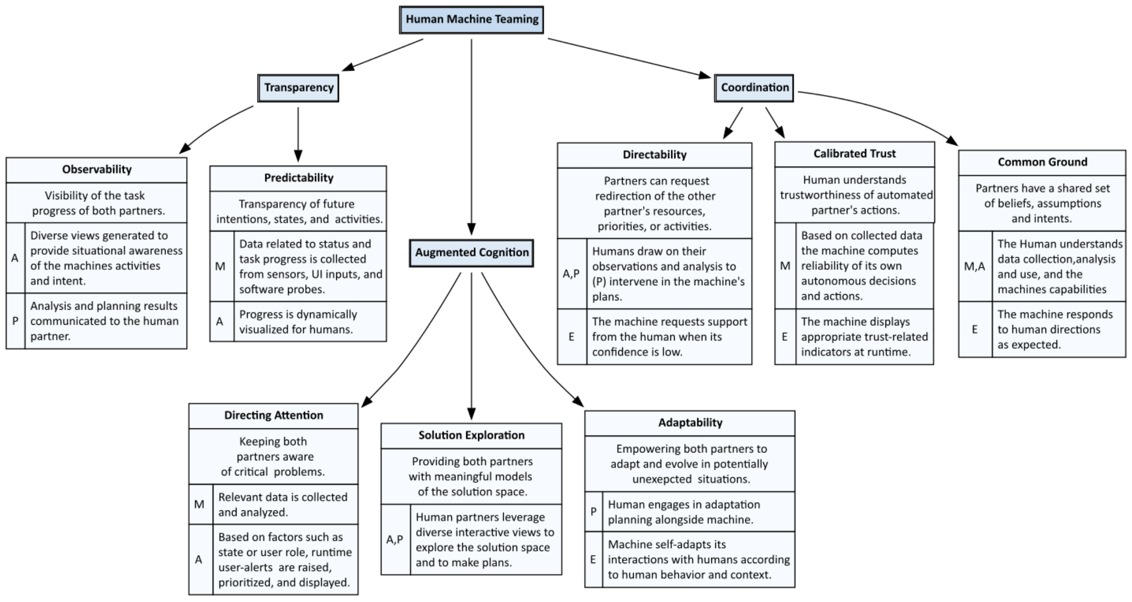
Transparency refers to the bidirectional communication that fosters a shared understanding between humans and machines. It emphasizes the need for both partners to maintain a clear picture of each other's actions, plans, and status to ensure effective cooperation [100, 110]. For example, humans should understand the actions and rationales of the machine, while the machine should be aware of the human's current mental and physical workload. This mutual understanding can guide the machine's decisions, particularly when determining whether to act autonomously or to seek human intervention in situations where uncertainties exist.

Augmented Cognition refers to the machine's role in enhancing human cognitive capabilities and supporting human decision-making. Machines can offer additional perspectives, such as providing physical views of a scene, increasing data processing capabilities, providing decision support, and offloading routine or complex tasks. The machine can potentially adapt according to the human's cognitive state, workload, and preferences, assisting the human partner in identifying critical situations and exploring a broader range of possible strategies. The ultimate goal of Augmented Cognition is to improve the outcomes of the human-machine team by balancing human and machine responsibilities, and complementing each of their cognitive limitations or weaknesses through the partners' strengths.

Finally, *Coordination* in HMT involves establishing shared knowledge and trust to facilitate effective joint actions between humans and machines. In the context of *MAPE-K_{HMT}*, we define human trust in the machine, as the expectation that the machine will act according to the best interests of the mission, perform reliably within its perceived capabilities, and will make decisions based on sound and transparent rationales [16, 52, 86]. Trust is therefore built over time as humans observe correct machine behavior in diverse settings. Given human trust in the machine, coordination is the process through which both partners work together in pursuit of shared goals, with the ability to request assistance from each other when needed. Unlike traditional adaptation capabilities, the coordination in an HMT environment extends the machine's self-adaptation capabilities to allow dynamic, continual configuration and reconfiguration of its interactions with human partners. This ensures that the interaction remains responsive and adaptable to the current context and the human's needs, fostering a more effective and equal partnership.

MAPE-K$_{HMT}$ builds upon MAPE-K by augmenting the four standard phases of monitoring, analysis, planning, and execution. The MAPE-K loop with HMT enhancements is shown in Fig. 1. In the basic MAPE-K loop [74] (indicated by the robot icons in the figure), data is collected from the environment in the *Monitoring* phase and then *analyzed* to determine if adaptations need to be performed. Corresponding actions and adaptations are then *planned* and ultimately *executed*. The "K" represents an underlying knowledge base, which is typically supported by runtime models and accessible by all parts of the MAPE loop [55].

Fig. 2. Key impacts of HMT factors upon phases of the MAPE-K loop. The eight factors are proposed by McDermott *et al.* [91] and augmented to reflect the bidirectional partnerships proposed in this paper. Given the dependencies that exist across all phases of the MAPE-K cycle, each factor is directly or indirectly integrated into every phase. Examples are given for phases of particular importance to each HMT factor.


Human Machine Teaming
Transparency
Augmented Cognition
Coordination
Observability
Visibility of the task progress of both partners.
A
Diverse views generated to provide situational awareness of the machines activities and intent.
P
Analysis and planning results communicated to the human partner.
Predictability
Transparency of future intentions, states, and activities.
M
Data related to status and task progress is collected from sensors, UI inputs, and software probes.
A
Progress is dynamically visualized for humans.
Directability
Partners can request redirection of the other partner's resources, priorities, or activities.
A,P
Humans draw on their observations and analysis to (P) intervene in the machine's plans.
E
The machine requests support from the human when its confidence is low.
Calibrated Trust
Human understands trustworthiness of automated partner's actions.
M
Based on collected data the machine computes reliability of its own autonomous decisions and actions.
E
The machine displays appropriate trust-related indicators at runtime.
Common Ground
Partners have a shared set of beliefs, assumptions and intents.
M,A
The Human understands data collection,analysis and use, and the machines capabilities
E
The machine responds to human directions as expected.
Directing Attention
Keeping both partners aware of critical problems.
M
Relevant data is collected and analyzed.
A
Based on factors such as state or user role, runtime user-alerts are raised, prioritized, and displayed.
Solution Exploration
Providing both partners with meaningful models of the solution space.
A,P
Human partners leverage diverse interactive views to explore the solution space and to make plans.
Adaptability
Empowering both partners to adapt and evolve in potentially unexepcted situations.
P
Human engages in adaptation planning alongside machine.
E
Machine self-adapts its interactions with humans according to human behavior and context.

However, to support human-machine teaming, we extend the MAPE-K capabilities with the human-related factors summarized in Fig. 2. These factors are grouped into three categories of transparency, augmented cognition, and coordination. *Transparency* includes "Observability" (TF1) of the autonomous partner's task progress and "Predictability" (TF2) of its future plans. *Augmented Cognition* includes the ability to "Direct Attention" (TF3) to critical problems, for example by raising meaningful alerts to increase situational awareness. Traditionally, the emphasis has been on the way a machine supports the human's situational awareness; however, we also consider ways in which the human contributes to the machine's situational awareness. Cognition enables "Solution Exploration" (TF4) and "Adaptability" (TF5) by both humans and machines, by imbuing them with the knowledge and capabilities they need to make and enact decisions. Finally, bidirectional *Coordination* is supported through "Directability" (TF6), "Calibrated Trust" (TF7), and the establishment of "Common Ground" (TF8) that together enable informed, trustworthy, and trusted partnerships.

Each of these capabilities is supported by HMT models that collect, aggregate, and visualize information in order to enable humans and machines to engage in meaningful interactions and work together to accomplish their joint goals. In Fig. 1 we provide a high-level view of how these teaming factors can be integrated across each phase of the MAPE-K loop as M^+, A^+, P^+, E^+, and K^+ respectively.

The subsequent phase enhancements are discussed in more detail in the following sections.

2.1 MAPE-K Phases

The Monitoring Phase (M) focuses on *collecting data* from the self-adaptive system and the environment in which it operates. Raw data related to attributes such as temperature, distance to potential obstacles, video streams, or GPS locations, is collected directly from hardware sensors, while runtime data such as resource usage, response times, and information about currently executing tasks is collected using software probes [62]. This data is continually collected for use in runtime models to guide the sUAS' analysis and self-adaptation decisions [69, 126].

M^+: To forge effective human-machine partnerships, situational awareness must be bidirectional, meaning that not only should the human understand what the machine is doing, but the machine must have certain degrees of awareness about what the human is doing – or not doing. Given the dissonance between the way humans and machines perceive the world [138], data is collected to support both the human's and machine's situational awareness from the machine, the environment, and human inputs. Therefore, the machine not only collects and publishes data about its own state, but also collects human-initiated data reflecting human goals, directives, workload, and response times [44]. Human-related data is collected via Graphical UIs (GUIs) and hardware interfaces such as radio controllers (RC), audio devices, pointing devices, and even eye-trackers [25, 51, 99, 102] or brain interfaces [53, 97]. MAPE-K's data collection process is therefore expanded accordingly and the collected data is used across subsequent analysis, planning, and execution phases to support the HMT goals of transparency, cognition, and coordination, with an emphasis on the teaming factors of *observability*, *directing attention*, *calibrated trust*, and *common ground*.

The Analysis Phase (A) is concerned with determining whether adaptation actions are required, based on the current and predicted state of the system, the environment it is operating in, and its defined goals, safety constraints, and quality of service specifications. Automated analysis enables timely and fast reactions to changes in the environment and emergent situations.

A^+: Within an HMT environment, both humans and machines are capable of analytical "thought". Transparency goals provide humans with situational awareness about the current state of the mission and status of each sUAS, thereby supporting their analysis. However, HMT also benefits from augmenting machine analysis capabilities with human perspectives and inputs [138]. Human-facing interfaces provide humans with situational awareness of the autonomous machine's current status, intent, performance, plans, and reasoning processes' [66, 116]; but also enable humans to provide information to the machine in the form of *real world* observations, as well as human-initiated directives. Humans observe the machine in two particular different ways. First, they use their physical senses to directly observe the machine's physical behavior in real-time. These direct observations enable humans to intervene quickly and directly through either a hardware or software interface, potentially causing a temporary interruption to the MAPE-K loop. However, much of the detailed analysis of the machine (e.g., its flight plans, health, autonomy levels) are supported through GUI representations where communication and processing latency of the data displays, and the time humans take to mentally process the data and formulate decisions, means that humans often act upon stale data. GUI support for HMT must, therefore, include dynamic runtime views that reflect current system states and historical information about past actions, rather than providing static snapshots of the system. Views are therefore often supported by HMT models with continually refreshed data. Historical views that enable the human to explore *why* a machine made a past decision (e.g., [108, 113]), or *what* it plans to do next, are also needed. The *MAPE-K*$_{HMT}$ analysis has a broad impact across almost all of the teaming factors as analysis is a precursor to human engagement.

In the **The Planning Phase (P)** the machine plans self-adaptation actions such as switching states to perform different tasks, reconfiguring existing features, activating or deactivating sensors, or modifying polling frequencies to preserve power or to collect additional information about the system or its environment.
P$^+$: HMT introduces two additional considerations to the planning phase. First, the human leverages their observations of the machine, its operating environment, interactions with other team members, and profound experience as a "human knowledge base" to engage directly in the planning process. Humans might reconfigure the mission's goals or plans, or temporarily intervene in the operation of the machine, for example by assuming manual control of a task that the machine is not able to perform autonomously or when the machine malfunctions. However, this introduces a potential tug-of-war that can occur when humans and machines create competing plans [37]. This was catastrophically illustrated in the crash of Lion Air Flight 610 and Ethiopian Airlines Flight 302 in which the MCAS (Maneuvering Characteristics Augmentation System) incorrectly perceived the angle of attack to exceed predefined limits and therefore pushed the nose of the plane down, whilst pilots struggled to push it back up [50]. The system was not designed to detect and mitigate this type of tug-of-war, and ultimately the machine "won", causing the planes to crash. Achieving effective coordination between humans and machines is a challenging problem which we discuss further in Section 4.3.

HMT has a second major impact on the planning phase, as the machine may make additional self-adaptation plans targeted at enhancing the human's interactive experience. For example, the machine might self-adapt its internal alert system to adjust the type and frequency of human alerts if the system perceives that human response time is starting to lag [3].

During the **The Execution Phase (E)**, the previously generated plan or adaptation strategy is executed on the physical machine or device.
E$^+$: In an HMT setting, both the machine and the human partners enact plans – sometimes closely coordinating their work whilst at other times working more independently on tasks that each partner is best suited to perform.

The Knowledge Base (K): MAPE-K employs diverse runtime models that represent the structure, behavior, and/or goals of a system at runtime. These models are pivotal for guiding autonomous adaptation decisions [9]. In a recent systematic literature review Bencomo *et al.*, [14] reported that *Models@run.time* have been used in numerous ways – for example to depict the current state of the system [46] and its behavioral dynamics specifying exactly what the system is able to do from its current state [48], model system goals [27, 112, 124] and functional and non-functional requirements [30, 71], and to depict product variability [132]. They provide a bidirectional reflection layer, such that changes in the runtime model trigger changes in the goals, structure, and/or behavior of the underlying system, whilst changes in the system are reflected in the model. As such, runtime models support system autonomy, imbuing the system with the ability to sense, analyze, predict, and make independent decisions.
K$^+$: In *MAPE-K*$_{HMT}$ runtime models must also support the HMT goals of transparency, augmented cognition, and coordination between human and machine partners. Depending on the type of system, application domain, or the type of missions that are executed, this requires different types of runtime models that are explicitly designed to provide information to the user or to relay critical information from the user to the system so that informed adaptation

decisions can be made. The need for transparency, cognition, and coordination across all phases of the *MAPE-K*$_{HMT}$ loop means that runtime models are not only executable, but must also be closely interconnected [129, 130]. This is achieved by modeling them with clearly specified *Inputs*, *Outputs*, and *Human Interaction Points* as illustrated in Fig. 3. We distinguish three distinct forms of inputs that include (1) human commands and/or feedback, (2) sensor inputs from the flight controller and/or onboard and off-board IoT devices, and (3) input values generated from other runtime models. Similarly, there are three distinct forms of outputs. As depicted in Fig. 3, outputs A and B are directed at the human. A initiates a human-facing request for feedback, whereas B displays information for purposes of situational awareness and has no expectation of a response. Finally, output C is not directly human facing, and produces data that potentially serves as an input value for other runtime models. Unlike the standard models@runtime which provide a bidirectional reflection layer between the underlying system and the models, the HMT-related models have a less direct causal relationship with the underlying system due to focus on human interaction. Changes in the model are therefore reflected in human-facing aspects of the system through modifications to the GUI or through observable machine behaviors such as task delays that allow humans time to respond. In turn, human inputs are reflected in the system, which in turn leads to model updates.

Figure 1 emphasizes the collaborations that occur between the partners. For example, in the Monitoring phase, the machine monitors the environment, including the human's behavior, whilst also accepting direct inputs from the user to complement its own sensing abilities. A symmetrical process also occurs, whereby the human monitors the machine through graphical and hardware user interfaces, and uses natural senses (such as eyes and ears) to observe the machine's behavior and to accept direct requests for help. Similar synergies, as described above, occur across other *MAPE-K*$_{HMT}$ phases.

3 CASE PROJECT: A MULTI-SUAS SYSTEM

The examples and experiments described in this paper are all based on our own *Drone Response* system (see https://youtu.be/DyKqxkesgg0). Many of the presented HMT models have therefore been deployed in real-world settings with physical sUAS, while others have been tested in our high-fidelity simulation environment.

3.1 The Drone Response Ecosystem

Drone Response is a distributed multi-user, multi-sUAS system, developed as a product line for supporting diverse operations such as search-and-rescue [2, 4, 33] and aerial data collection and analysis [1]. Each sUAS in the *Drone Response* system is equipped with onboard compute capabilities that support self-adaptive autonomous behavior. A Ground Control Station (GCS) is responsible for centralized activities such as global mission configuration, airspace leasing, and task coordination, while several graphical and hardware user interfaces facilitate bidirectional communication and interactions between multiple sUAS and humans.

Drone Response represents a managed system [133] implemented using the MAPE-K loop at two distinct levels. First, the onboard autonomous system represents a relatively traditional MAPE-K loop, while the broader ecosystem, which includes the onboard autonomous pilot, the ground control system, and humans interacting with the system through various UIs, represents a broader *MAPE-K*$_{HMT}$ loop. We discuss these two perspectives again after describing each of *Drone Response*'s individual elements.

figures/runtime_model_overview_v2.pdf

Fig. 3. All Runtime Models accept inputs from (1) humans, (2) onboard sensors, including sensors integrated into sUAS flight controllers, and (3) values generated from other runtime models. They generate output values used (A) as part of direct user requests, (B) used in GUIs, (C) fed to other runtime models, or (D) sent to actuators in the flight controller or IoT devices.

• *sUAS Onboard Autonomous Pilot:* The *Onboard Autonomous Pilot* (OAP) is responsible for the sUAS' autonomous and self-adaptive behavior. Its core architecture is centered around a state machine which is dynamically configured with states and transitions for the current mission. The OAP communicates with the ground-based GCS and potentially with other sUAS via its onboard MQTT client. At the start of a mission, it receives a mission specification in a JSON format and uses this specification to self-configure its onboard state machine. Each individual state represents a specific task such as *takeoff* or *search*; and task progression during the mission is enabled by state transitions, triggered by events that the sUAS independently detects using its onboard sensors through messages received from the GCS or other sUAS. Some states and subsequent transitions are simplistic in nature. For example, in the *takeoff* state, the sUAS ascends to a predefined altitude, uses its onboard sensors to detect when it reaches that altitude, and then transitions to a subsequent state such as "fly-to-waypoints" or "survey" according to the activated state transition. Other states and their transitions leverage more sophisticated AI-supported intelligence. For example, a *search* state utilizes onboard computer vision to continually search for a person on the ground or in the water. It monitors outcomes of the CV model, analyzes the results, and with it identifies the victim with some degree of confidence and certainty [1], it coordinates with other sUAS via the GCS to determine which sUAS should track the victim whilst waiting for human rescuers.

• *Ground Control Station:* The Control Station is built around a set of microservices loosely connected via a message broker [95]. Each microservice supports a specific capability such as airspace leasing, multi-sUAS coordination, or a task sequencing validity checker [64]. Microservices subscribe to different topics in order to receive messages from three unique sources that include human operators, other sUAS, and runtime models within the MAPE-K knowledge base. In turn, they publish outputs as messages to MQTT using topics subscribed to by other microservices, GUIs, or sUAS. In effect, MAPE-K's knowledge base is distributed across a set of runtime models distributed across microservices, client-side components, and onboard each sUAS. These models communicate via MQTT and are supported by a shared in-memory database, reducing the need to store duplicate runtime data. Status data (e.g., GPS location, battery, health) and task progress updates (e.g., current task, potential adaptations), are sent to the GCS by both humans and sUAS in support of MAPE-K's mission-level monitoring, analysis, and planning.

• *Graphical and Hardware Interfaces: Drone Response* leverages both graphical and hardware user interfaces to enable human-machine interactions. Under normal operating conditions, interactions are primarily GUI-based; however, in case of emergency, or to temporarily assume control for tasks that the machine has not yet been trained to perform, humans can directly issue commands to sUAS via hand-held radio controllers. The *Drone Response* GUI is a centrally hosted *web application* allowing human-to-human interactions across multiple devices. Communication is primarily asynchronous with data sent over a mesh radio via MQTT brokers situated on the GCS and onboard each sUAS [3]. MQTT offers three QoS (Quality of Service) levels of which we leverage two. Status messages are sent from the sUAS to the ground at the lowest level QoS-0, which provides no guarantee of delivery, while human commands sent to each sUAS use QoS-2 which guarantees that the message is received exactly once. By default, Microservices publish to other microservices using QoS-0.

Many of the $MAPE\text{-}K_{HMT}$ runtime models are coupled with one or more GUIs that provide situational awareness to the human for planning and analysis purposes while also potentially monitoring aspects of human interaction behavior. A physical handheld controller serves as a backup and allows a human operator to assume direct control of an sUAS should it be deemed necessary in an emergency situation, or in order to execute a series of tasks and/or maneuvers that the sUAS OAP is not yet capable of performing independently.

3.2 *Drone Response* as a Managed System

We illustrate the runtime environment of *Drone Response*, and its support for HMT in Fig. 4. The "edge" component, shown at the top part of the diagram, incorporates a traditional MAPE-K loop in which the sUAS monitors changes in mission or environment state through data it collects directly from its own sensors or indirectly as messages sent via MQTT from other sUAS, microservices, or the GUI. Analysis occurs within various components such as the Computer Vision module (e.g., when a person is detected with a certain degree of confidence), planning occurs within the *currently active state* in the state machine (e.g., to determine whether a transition should be made to another task), and the plan is executed either when the state machine transitions to another state and the sUAS enacts the new task (e.g., transition to hover) or through adaptations to its plans within the current task (e.g., adjust altitude and speed). Traditional human-on-the-loop scenarios play out within this MAPE-K loop when human inputs are received via MQTT messages.

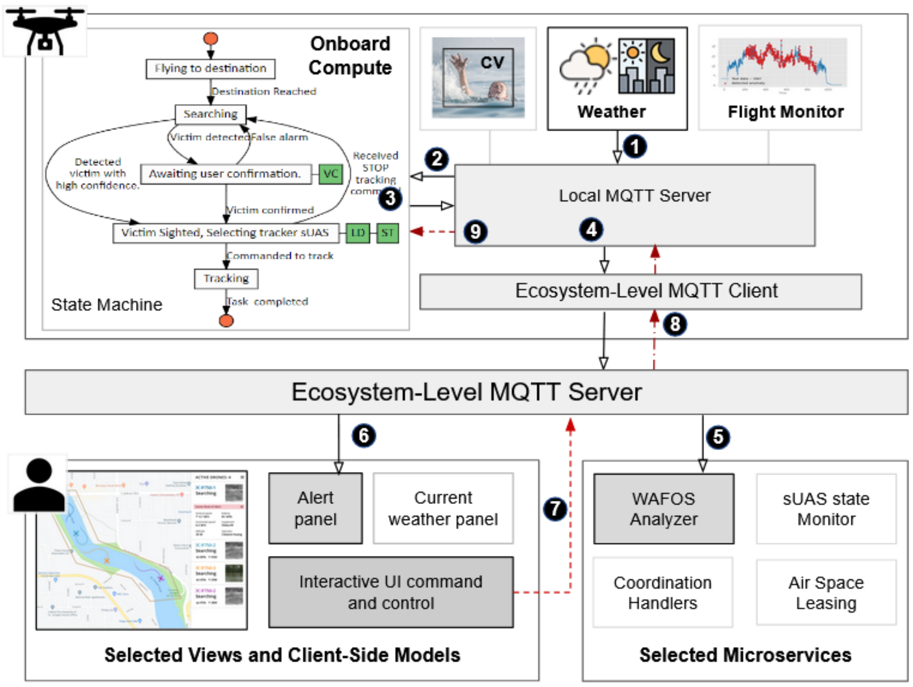


Fig. 4. Human-Machine Teaming is supported in *Drone Response* by hardware and graphical UIs and a set of closely integrated runtime models distributed across the onboard computer and the Ground Control Station. The onboard autopilot and flight controller form a managed system, while the entire ecosystem of GUIs, Ground Control Station, and onboard compute capabilities forms a managed HMT System.

In addition, the entire *MAPE-K*$_{HMT}$ loop acts as a managed system at the ecosystem level, as illustrated through an annotated example in Fig. 4 which depicts the onboard autonomous pilot (hosting its traditional managed system), the GCS hosting MQTT and various microservices, and the humans supported by various UIs. In this example, the onboard runtime weather model detects poor visibility due to foggy conditions and publishes an alert to the local MQTT server (Step 1). The onboard pilot's state machine subscribes to weather-related events and is therefore notified of the fog (Step 2), adapts its behavior to fly lower and slower in order to compensate for the low visibility, and publishes a message explaining its autonomous action (Step 3). This message is forwarded to the GCS's MQTT broker (Step 4), which in turn forwards it to the WAFOS Microservice to determine what kind of alert (if any) should be raised (Step 5). Assuming an alert is needed, the alert messages are sent via the MQTT broker to the front-end GUI (Step 6) and displayed on various screens and panels (e.g., Alert panel, Weather panel). The loop is closed when the user sees that an adaptation has occurred, but based on the user's personal knowledge of the current weather conditions, is aware that a storm is approaching. They, therefore, direct the sUAS to re-establish its original speed whilst retaining altitude (Steps 7-9), and the sUAS adapts accordingly. We discuss this type of exchange between humans and sUAS later in the paper, especially in light of tug-of-war scenarios in which the human and sUAS make conflicting decisions. However, this example, illustrates how humans and sUAS can collaborate on common tasks within the *MAPE-K*$_{HMT}$ managed system.

4 APPLYING MAPE-K HUMAN-MACHINE TEAMING

In this section, we present a number of different HMT models from our *Drone Response* system that support HMT within the *MAPE-K$_{HMT}$* loop, explore how each of the HMT goals is satisfied from both the human and machine partners' perspectives, and highlight the types of human-machine interactions that are intrinsic to each model. While in practice, individual models often support more than one HMT goal, we select examples that emphasize aspects of the three HMT goals of transparency, augmented cognition, and human-machine coordination.

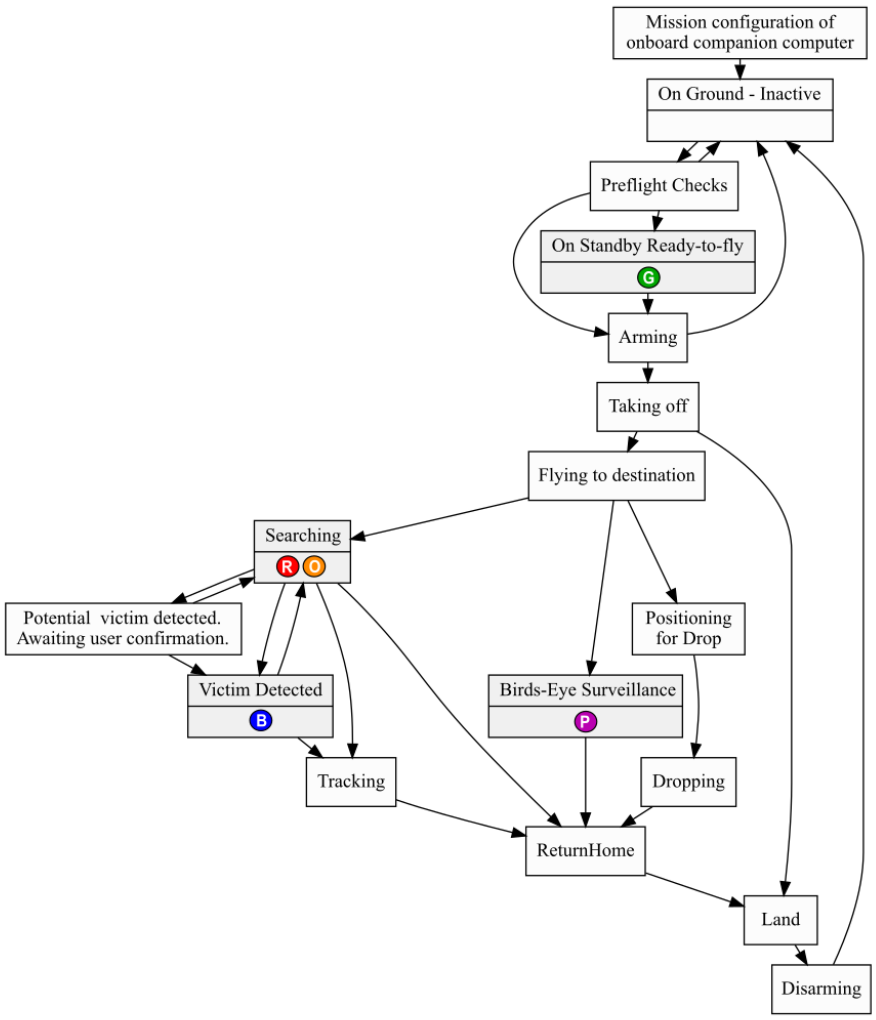


Fig. 5. This 'Multi-Agent Tracking' view displays the task progress of all active UAVs. It is managed by a dedicated microservice, which continually aggregates states and transition paths for all active UAVs and uses each UAV's uniquely colored token to mark their current state.

4.1 HMT supporting bi-directional Transparency between Machines and Humans

Transparency goals center around observability and predictability and align closely with Endsley's Situational Awareness goals of *observing* and *understanding* [44, 45]. Traditionally, observability has enabled humans to maintain awareness of what their autonomous partner is doing, including its goals, current tasks, future intentions, status, progress, uncertainties, ability to adapt to changing contexts, rationales for adaptations, and challenges or constraints that impact its ability to solve the current problem. However, for HMT purposes, transparency must be bidirectional as the machine also needs transparency into the human's current cognitive workload. In addition to observability, *predictability* helps to eliminate potential emergent surprises introduced by the machine's decisions, and provides insights into uncertainties [13, 55] and ways in which reliability of the autonomous partner changes over time, under what circumstances it changes, and rationales for decisions it is making. Given the importance of the transparency goal and the many facets of observability and predictability, systems will likely have several transparency related HMT models pertaining to these aspects. These models tend to process runtime data depicting the machine's progress and its environment, and visualize that data in ways that support the human throughout various stages of the mission. For example, in the preflight stage, *Drone Response* displays the current state of each sUAS and the progress of its preflight checks so that the human can monitor the sUAS' flight readiness and mitigate problems as they occur. During the mission, *Drone Response* provides several different views including maps that depict the current location and status of each sUAS, task-based views that show the current action that each sUAS is performing, and environment views that offer insights into current operational conditions such as weather. We illustrate transparency with two *Task Awareness Models*. The first provides situational awareness to the human, enabling them to track the state of each individual sUAS, as well as the overall mission; while the second provides situational awareness to the machine (in this case the sUAS) concerning the current workload of the human from which it can infer human availability for helping in decision making.

- *Task Awareness Model:* In a multi-agent system, the human needs to understand exactly what each individual agent is currently doing and its progress towards the overall mission plan. Fig. 5 provides one example of a task-centric view generated from the underlying task awareness HMT model. The main intention of the model is to track the current state (e.g., `Land`, `Searching`) of each sUAS. *Drone Response* manages the model using a dedicated microservice that subscribes to all sUAS status messages, checks them for state changes, and publishes state changes as they occur. In turn, a second runtime model, residing on the client side, tasked with visualizing a merged view of all active sUAS state transition models, subscribes to state changes and displays each sUAS' current task as a colored token assigned to a specific node. Currently, the view shows that the Green (G) delivery sUAS is *on standby*, Red (R) and Orange (O) sUASs are *searching*, the Purple (P) sUAS is performing *surveillance*, and the Blue (B) sUAS has *detected a victim*. This model represents the case in which the system provides information for the human – Machine-to-Human (M2H) – but does not expect a direct response. As the human uses the data to acquire and maintain situational awareness and may make decisions based on their observations, it is critical that the generated information is continually refreshed with low latency.

- *WAFOS Visibility Model:* Traditionally, when we talk about concepts such as observability, predictability, and situational awareness, the focus has been on human awareness. However, in an HMT environment, the machine can also benefit from understanding what the human is doing and/or is likely to do. We, therefore, present an HMT WAFOS model (Workload, Anxiety, Fatigue, and Other Stressors) [29, 45], designed to measure if and how well a human is maintaining situational awareness. This, in turn, pertains to the ability of the human to interact with the sUAS in a timely manner. Monitoring human behavior can take many forms, but if used inappropriately can be perceived as violating personal

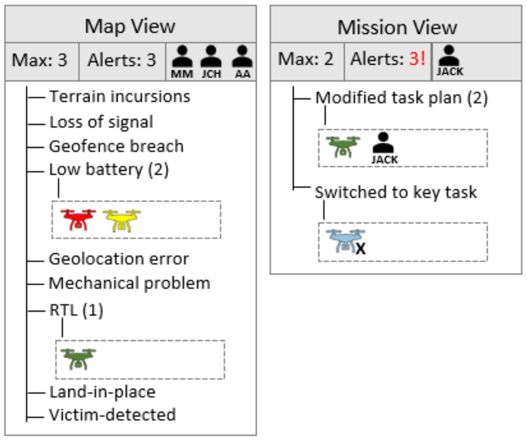


Fig. 6. The Alert Views model tracks the currently displayed alerts on each open GUI screen. While users can sometimes share GUIs, it serves as a proxy for user workload.

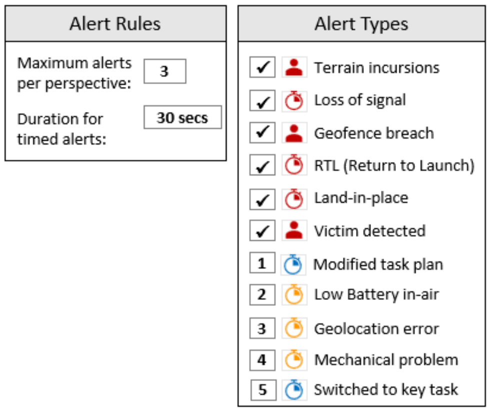


Fig. 7. The Alert Rules model contains the current rules for triaging alerts. These rules can by dynamically modified at runtime if users become less responsive or request fewer alerts.

privacy [119]. In this setting, *w*orkload refers to the tasks that need to be managed. The NASA Task Load Index (NASA-TLX) [57], which has been broadly adopted, measures workload through aggregating scores for mental, physical, and temporal demands, effort, and frustration levels. *A*nxiety is seen as a psychological state that can influence an individual's ability to accurately perceive, understand, and predict situational elements. Fatigue impacts the ability of a user to perceive, understand, and project future states of the system -- all of which are key abilities for maintaining situational awareness, while other stressors, such as anxiety, toxic working environment, or extreme flying conditions, can exacerbate these problems and impair human capabilities. Anxiety is often measured using psychological assessment tools, such as the State-Trait Anxiety Inventory (STAI) [121], while stress can be measured using tools such as the Perceived Stress Scale (PSS) or by measuring physiological indicators such as cortisol levels or heart rate [107]. The use of questionnaires and other psychological tools during an sUAS mission is impractical, while direct physiological measurements could be used, especially in military or extreme fire-fighting operations, but are otherwise likely, not appealing. Our model therefore estimates WAFOS through indirect means.

In a multi-sUAS system, each additional sUAS that an operator is responsible for monitoring, and each alert raised against that sUAS, can increase workload and hence stress levels [134]. The operator not only carries responsibility for monitoring the safe operation of the sUAS, but must be prepared to intervene or interact with each sUAS when needed [26, 98]. We therefore adopt a lightweight approach that collects a limited amount of data for each user during a single session, but does not attempt to store or aggregate user data across sessions to permanently "profile" users. Our WAFOS model simply maps each user to their active views, records the number of currently *monitorable items* in each active GUI, and measures recent response times of the human. The rationale for using workload as an indicator of performance has been studied and documented in numerous papers [26, 98].

The WAFOS model is supported by the *Alert Views* model in Fig. 6, which tracks the number of sUAS displayed in each view and the number of alerts that are currently raised. In addition, whenever an individual sUAS requests help from the human (e.g., confirmation of a victim sighting in a search-and-rescue mission), the WAFOS model monitors and tracks the time that a human takes to respond. All of this information is collected and aggregated in a WAFOS

Table 1. A Partial WAFOS (Workload, Anxiety, Fatigue, and Other Stressors) model showing recent activity for one human operator

Time Period												User Action		
10	9	8	7	6	5	4	3	2	1	0	**Event**			**Action Description**
🟨											D1 battery at 30%	○		Operator observes the alert, but takes no action. They trust the system level failsafes to manage the situation.
	🟩										D2 self-adapts to optimize flying altitude for CV task.	○		Operator observes, but accepts the sUAS' decision to optimize altitude for the current CV task.
			🟨	🟨	🟨						D3 raises lost signal alert and then regains signal.			No action is possible because communication is lost. Operator trusts inbuilt failsafe mechanisms, and signal is reactivated.
					🟥	🟥	🟥	🟥	🟥	🟥🧍	D2 detects victim and requests user confirmation	●	►	User fails to respond. sUAS waits allotted period of time for response and then makes independent decision to track victim.
							🟨🧍				High critical. D1 battery at 15%.	○	►	Operator intervenes and advises sUAS to RTL.
							🟩				D1 Returns to launch	○		No action required by operator except to monitor RTL.
								🟨			D4 preps to replace D2	○		Operator notified that drone D4 is being prepped to replace D1.
								🟥🧍			D4 requests launch. approval	●	►	Operator approves launch.

User Response: ●=Required ○=Optional; ►=Action taken, 🧍=Time at which operator responds.

model, and a WAFOS score is computed. This score can be used by the sUAS or other parts of the system, such as microservices or a GUI, to decide what information to present to human operators at any given time. The WAFOS model is therefore an example of Human-to-Machine transparency.

For illustrative purposes, data for a single sUAS operator is shown in Table 1 for 11 time units, where 0 is the current time (t), and 10 represents t-10. Each alert is classified using predefined tags as low (L, green), medium (M, yellow), or high (H, red) criticality. Some alerts require (REQ) responses, while others allow optional (OPT) responses, and still others are informational only. In this example, the user responds to three alerts, two REQ alerts, and one OPT alert. Our initial formula for computing WAFOS scores is shown below:

- Current time T_i, and T_j, where j is T_p units from i, with T_p being the time window.
- Weight w_1 for N_{sUAS}, where N_{sUAS} represents the maximum number of sUAS in the air during the period.
- Weights w_2, w_3, w_4 for the count of each criticality alert: $N_L(j)$, $N_M(j)$, $N_H(j)$, at time step T_j. Each critical alert is multiplied by a decay function λ, which we define as $\lambda(x) = (0.7)^x$. This ensures that we prioritize more recent alerts. Higher weights will be accrued the closer to current time i.
- Weight w_5 for the duration in time for each required and optional alert that was delayed. For a required or optional alert π we define $\delta(\pi)$ as the response time it took the user to answer the alert (if the alert has not been answered then we assume that they will answer in the next time step), and $Req(j)$ as the set of required and optional alerts at time T_j.

Workload can then be derived as:

$$W = w_1 N_{sUAS} + \sum_{j=i-T_p}^{i} \left(\lambda(i-j)\Big(w_2 N_L(j) + w_3 N_M(j) + w_4 N_H(j)\Big) + w_5 \cdot \sum_{\pi \in Req(j)} \delta(\pi) \right) \tag{1}$$

which computes to a score of 34.135 for the example shown in Table 1, considering $(w_1, w_2, w_3, w_4, w_5) = (4, 1, 2, 3, 1)$. In a mission-critical environment, where it is not expedient for the sUAS to wait indefinitely or for long periods for input from an overloaded human, if the WAFOS scores rise above a predefined threshold (e.g., 60), the sUAS could potentially increase its own autonomy level and replace requests for human input with informational alert. The human still has the opportunity to intervene, but the potential bottleneck is removed. Determining proper thresholds for WAFOS triggers and appropriate adaptations of human and machine rights and responsibilities under different circumstances remains an open research issue that we are currently addressing. While this model is inspired by our prior quantitative analysis of WAFOS-related stressors [3], and supported through our own experiences of many hours of multi-sUAS flights, the model has not yet been validated in human studies.

4.2 HMT Models that Augment Cognition for both Humans and Machines

The HMT goal of augmented cognition extends far beyond basic transparency and is traditionally intended to help humans understand emerging problems and their causes, provide insights into the decisions and actions taken by the autonomous partner, and allow the user to explore different perspectives and solutions [81]. *MAPE-*K_{HMT} runtime models support cognition goals by enabling the right information to be presented or available to the human at the right time, without overloading their cognitive abilities [3, 76, 135]. We extend this notion in the opposite direction, where the human provides pertinent information at runtime for use by the machine in its own decision-making. The two examples of Machine-to-Human (M2H) models we present here focus on *triaging runtime alerts* and *explaining autonomous actions* of the machine by generating human-readable explanations of autonomous behavior; however, our previously presented task-centric model (cf. Fig. 5) provides an additional example, as it's interactive GUI also provides interactive and more detailed views of individual sUAS's tasks and progress. The Human-to-Machine (H2M) model that we provide focuses on an example in which the human updates the knowledge base to guide the machine's computer vision tasks.

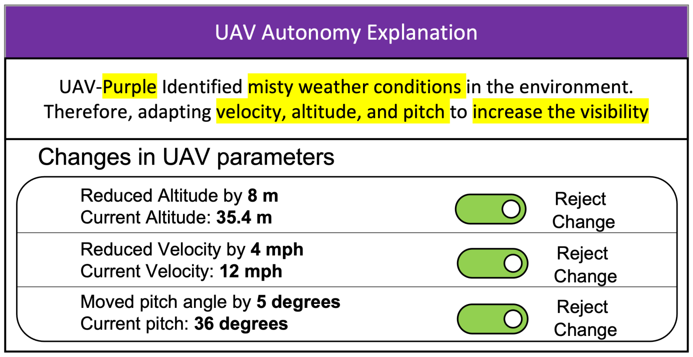
- *Alert Triage Model:* The *Drone Response* alert triage model is designed to support cognition through avoiding the situational awareness "design demon" of information overload [44], and is built upon a formal meta-model for human-sUAS collaborations [5]. It focuses particularly on the HMT goal of *augmenting cognition* with an emphasis on directing human attention to important messages. Fig. 7 illustrates the alert rules and priorities, which are initially provided as default values by human stakeholders but could be dynamically adapted at runtime by human requests or by the machine if WAFOS values indicate that the human operator becomes overloaded. They specify essential alerts which must always be displayed (✓), and prioritized alerts (1-5) which will only be displayed if they don't cause the maximum threshold to be exceeded. As previously explained, the *triage* part of the model (e.g., Map View), is dynamically maintained by the system at runtime and is responsible for managing alerts in each active GUI view. It is notified whenever an alert is generated by a runtime model hosted on the sUAS or on a GCS microservice. It is also notified by the GUI server whenever a new GUI is activated or deactivated. The alert prioritization model thus builds upon services already available in the basic MAPE-K loop, by collecting, aggregating, and processing the data they produce, in order to support the HMT-focused capability of triaging alerts.

- *Adaptation Explanation Model:* The explanation model generates explanations for sUAS autonomous decisions so that humans can gain insights into the machine's reasoning and assess the appropriateness of individual adaptations [3, 78]. Autonomy in *Drone Response* occurs at two levels. At the most basic level, default failsafe mechanisms are built into the flight controller and configured for each sUAS. Examples include setting failsafe actions, such as RTL (return to launch) or LAND commands, that the sUAS must take when the battery reaches critically low levels or breaches a geofence. However, the onboard autonomy software makes more sophisticated adaptation decisions, which must be communicated or queryable by the user.

These insights potentially strengthen the human's trust and confidence in its machine partner. The predefined *explanation templates* shown in Table 8a are used to dynamically generate human-readable textual explanations for all adaptations performed by an sUAS. Whenever an sUAS self-adapts, it collects three types of information. These are

Type	H/M	Explanation Template
Ext	M	UAV-{id/color} identified {Event} in the environment. Therefore, adapting {Action - internal changes} to {Rationale}
Ext	H	UAV-{id/color} identified {Event} in the environment. Therefore, need {Desired Changes} to {Rationale}
Int	M	UAV-{id/color} observed {Event}. Therefore, {Action - internal changes} to {Rationale}
Int	H	UAV-{id/color} observed {Event} due to {cause}. Therefore, need {Desired Changes} to {Rationale}

(a) Explanation templates for internally and externally triggered adaptation events initiated by either the human (H) or machine (M).


UAV Autonomy Explanation

UAV-Purple Identified misty weather conditions in the environment. Therefore, adapting velocity, altitude, and pitch to increase the visibility

Changes in UAV parameters

Reduced Altitude by **8 m** Current Altitude: **35.4 m**	Reject Change
Reduced Velocity by **4 mph** Current Velocity: **12 mph**	Reject Change
Moved pitch angle by **5 degrees** Current pitch: **36 degrees**	Reject Change

(b) An example explanation generated by the runtime model.

Fig. 8. The Adaptation Explanation Model generates an explanation for all major adaptation decisions.

explanation snippets describing relevant external events (e.g., "misty weather conditions", or "victim detected"), the sUAS's response to the events (e.g., "reduced altitude by 8 m", or "switched to tracking mode"), and finally, the rationale behind those actions (e.g., "limited visibility", or "high confidence in victim sighting"). An example of a weather-related adaptation explanation is shown in Fig. 8b. Upon receipt of the adaptation message, the runtime model selects the appropriate template and generates the explanation by filling in the missing parts with the data provided by the sUAS [3]. This model, therefore, utilizes the outputs of existing runtime adaptation models to provide explanations that are critical for HMT.

- *Human Updates to Shared Knowledge Model:* The human provides information to the sUAS in several different ways including (1) assigning high-level mission plans through the GUI, (2) responding to requests for help during decision-making, for example, when an sUAS seeks a human opinion in an object recognition task, and (3) through providing information that the sUAS is not otherwise able to sense. For example, during a search for a victim, the human might provide additional information that the victim is a white female wearing a red shirt and blue jeans. The sUAS can use this information to update its search parameters while enacting its CV-based search.

4.3 HMT Models for Coordinating Human and Machine Decisions and Actions

Coordinating the decision-making and subsequentactions of both humans and machines represents a challenging problem [82] that is exacerbated by the differing cadences of human and machine response times. Cutting-edge research in human-machine decision-making [70, 79], aims to empower machines with advanced abilities to reason, so that they can make increasingly complex autonomous decisions and actively collaborate with humans. Successful coordination of decisions and actions depends on many factors, including a shared conceptual model of the operating environment, respect for each partner's capabilities, and well-calibrated bidirectional trust, including humans' trust that the machine can operate autonomously when capable and request help when needed. In our current *Drone Response* system, machine intelligence is limited to two specific capabilities. First, onboard CV capabilities enable an sUAS to perceive its environment and make basic decisions related to detecting, surveying, and tracking objects of interest [1, 32]. Second, onboard analytics enable each sUAS to monitor its current state, detect anomalous sensor data that could impact flight safety, and make subsequent adaptations to avoid problems such as crashes, fly-aways, or other forms of flight deviations [65, 115]. Both cases provide opportunities for humans to engage in decision-making or to directly intervene in the flight.

4.3.1 Human-Machine Coordination.

We present three different HMT models covering the cases of machine-initiated and human-initiated coordination, as well as the particularly challenging case that occurs when humans and machines generate conflicting plans of action. Conflicts occur when two autonomous agents make different, incompatible, decisions. Our examples in this section focus on computer vision decisions when an sUAS detects an object of interest and makes subsequent enactment decisions.

- *Machine Initiated Coordination using Computer Vision:* In *Drone Response* much of the autonomous behavior of the sUAS is supported by its onboard CV capabilities [1]. For example, when searching for a drowning victim, the sUAS uses CV to continuously analyze the video stream and detect objects classified as "person". For each detected object, the CV module generates two scores. The *confidence* score represents the probability that the object is correctly classified, while the *reliability* score accounts for any *uncertainty* arising from image noise or mismatch between the current context and the training data [104]. Based on learned threshold values, the sUAS uses these scores to decide whether it can autonomously decide on its actions (e.g., track the object vs. continue to search) or whether it should request input from the human. Both, the CV model and the respective human-machine coordination mechanism are supported by runtime models which take a coordination specification as input (cf. Algorithm 1), instantiate a simple state machine (managed by a microservice on the GCS), and use the existing messaging system to choreograph human and machine tasks. In closely related work Li *et al.* [82], also proposed a form of choreography that provides users with advanced knowledge of tasks they would be requested to participate in.

- *Human Initiated Coordination Models:* In order for the human to engage in meaningful interactions with the machine, they must have high degrees of situational awareness and a clear understanding of what they can and cannot do at any point during a mission. Furthermore, the different cadences of human and machine response times within *MAPE-K*$_{HMT}$ create a significant coordination challenge resulting in problems such as the human making plans based on stale data or intervening after the machine has already completed an action. An example of a relevant runtime model determines currently available human interaction options and dynamically adapts each active GUI to activate and deactivate widgets (e.g., icons, buttons, menu options) according to the ways that humans can feasibly interact with the machine given the current state of the mission. For example, if the sUAS is currently in active search mode, it is reasonable for the human to request a view of the sUAS's annotated video stream; however, this option should not be available if the sUAS is in RTL mode with cameras turned off to preserve limited power. A suitable affordance (e.g., a button) should be activated and deactivated accordingly). Tracking these currently available human actions is handled by a dedicated runtime model.

- *Collaborative Decision Making:* A significant challenge in HMT is reconciling potentially conflicting actions taken by the human and machine, a problem that is again exacerbated by the different operating speeds of the machine and

Algorithm 1: Human-Machine coordinated decision-making addressing Computer Vision reliability problems

```
if Object detected at low reliability then
    UAV raises alert and requests help from human;
    if Human is available and responsive then
        Human evaluates video stream, makes decision, and selects CONFIRM or REJECT option;
        if Human confirms victim sighting then
            UI Server sends CONFIRMATION message to UAV;
        else
            Human refutes sighting;
            UI Server sends REFUTATION message to UAV;
        end
    if no response from human within waiting_period then
        NO RESPONSE message sent to UAV;
        'human failure to respond' event is logged;
        Responsibility reverts to UAV;
    end
end
```

human. As the machine has the advantage of a faster cadence, it will often win any disagreements with potentially devastating results, as illustrated by the case of the Lion Air Crash (cf., Section 2.1). Similar scenarios play out across other domains. For example, an sUAS may place itself close to the river to collect better imagery of the riverbank. On a sunny day, reflections from the light on the water could impact the sUAS's sensors causing sudden altitude fluctuations and triggering a land-in-place failsafe mechanism to activate. An alerted human might quickly intervene to prevent the sUAS from landing in the river directing it to ascend to a safe altitude; however, once the sUAS's autonomy kicks back into gear, the whole cycle could repeat itself – causing a tug-of-war with respect to the ideal altitude placement. We illustrate some of the dangers of tug-of-war with a recent crash that occurred with our own VTOL (vertical Takeoff and Landing) sUAS (see Fig. 9). While flying the VTOL manually in fixed-wing mode, the system unexpectedly raised a critical low-battery alert, and as a result, the pilot switched to QHOVER (quad hover) mode in order to immediately land the aircraft. However, due to the low battery event, the autopilot triggered an RTL (return to launch) failsafe causing the drone to gain height for its return home. The pilot switched back to QHOVER, overriding the RTL, but then had to land from a far greater altitude, which led to the quad motors failing – likely due to the battery problem. If the system had been configured to prevent such a tug-of-war scenario, the sUAS would have probably landed safely when the pilot initially attempted to land it.

These types of scenarios are not uncommon, and, therefore, HMT systems must allow humans and/or machines to detect and break interwoven cycles of human and machine actions that are indicative of a tug-of-war. Several potential solutions can be employed; however, finding the right approach for resolving conflicts is still a cutting-edge research problem with solutions at least partially dependent upon the cognition capabilities and intelligence of the sUAS.

Our current approach creates and utilizes an HMT model that we refer to as a *Conflict Decision Graph* (CDG) for each class of potentially conflicting decisions. There are two primary decision classes in *Drone Response*, namely decisions related to CV, and decisions related to runtime mitigation of detected anomalies. Building upon our previous example, we illustrate the CDG for a CV-related task of detecting a victim and performing a subsequent action (e.g., tracking). We configure the CDG by establishing different thresholds for categorizing confidence and certainty (i.e., reliability).

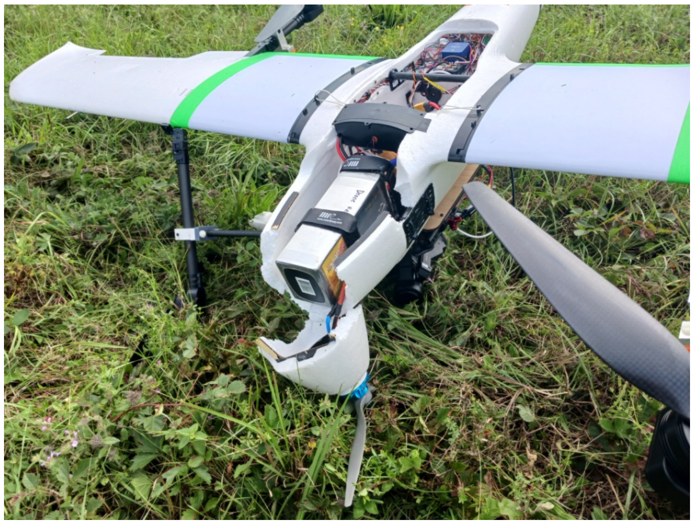


Fig. 9. Our VTOL drone crash-landed and broke its nose due to a real-life tug-of-war in which the autopilot initiated RTL, overriding the pilot's decision to land in QHOVER mode when a sudden drop in battery voltage occurred.

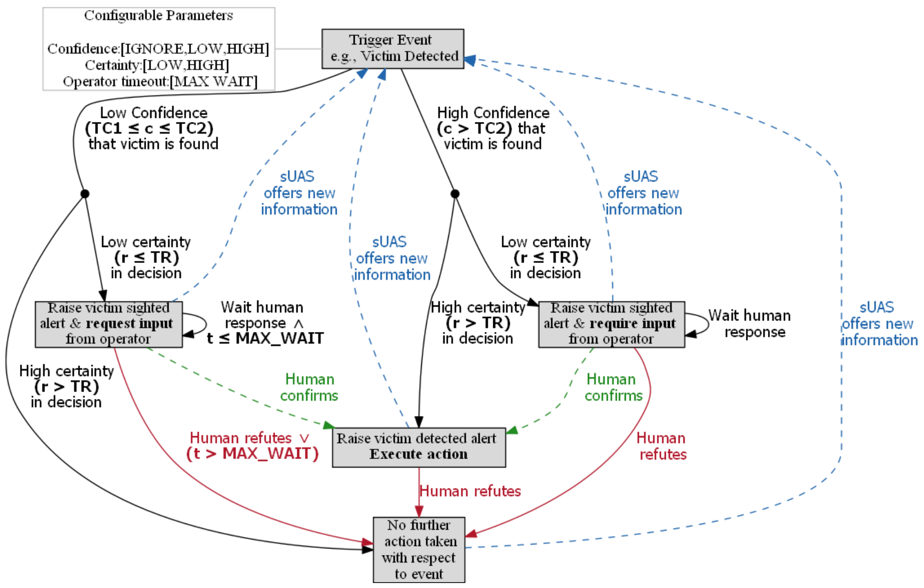

Configurable Parameters
Confidence:[IGNORE,LOW,HIGH]
Certainty:[LOW,HIGH]
Operator timeout:[MAX WAIT]
Trigger Event e.g., Victim Detected
Low Confidence (TC1 ≤ c ≤ TC2) that victim is found
High Confidence (c > TC2) that victim is found
sUAS offers new information
sUAS offers new information
sUAS offers new information
sUAS offers new information
Low certainty (r ≤ TR) in decision
Low certainty (r ≤ TR) in decision
Raise victim sighted alert & request input from operator
Wait human response ∧ t ≤ MAX_WAIT
High certainty (r > TR) in decision
Raise victim sighted alert & require input from operator
Wait human response
High certainty (r > TR) in decision
Human confirms
Human confirms
Human refutes ∨ (t > MAX_WAIT)
Raise victim detected alert Execute action
Human refutes
Human refutes
No further action taken with respect to event

Fig. 10. Conflict Decision Graph (CDG) for a computer vision tasks. The graph is configured for different tasks to specify threshold values for different confidence and certainty ranges and maximum time to wait for an operator if a constraint is specified for a specific task. All feedback is fed into the knowledge base and used to update the model itself and/or considered in future decisions to prevent tug-of-war scenarios. A similar graph is created for each type of collaborative decision. In *Drone Response* we have graphs for CV-related and onboard health-related human-machine decisions.

We categorize confidence (c) into three groups divided by two threshold values (TC1, and TC2) as follows: $IGNORE$: $\{c | 0 <= c < TC1\}$; $LOW\,CONFIDENCE$: $\{c | TC1 <= c <= TC2\}$; $HIGH\,CONFIDENCE$: $\{c | TC2 < c <= 1\}$ and split certainty into two groups where outcomes are determined by certainty level (r) as follows: $LOW\,CERTAINTY$: $\{r | 0 <= r < TR\}$; $HIGH\,CERTAINTY$: $\{r | TR <= r <= 1\}$. One final parameter MAX_WAIT, specifies the duration in seconds (t) that the sUAS should wait for the human to respond. This parameter is initialized with a default value and then updated according to WAFOS scores computed by the HMT WAFOS model.

These parameters are used to configure the CDG to support diverse relevant tasks. For example, detecting a drowning victim is a life-critical task and so TC1, TC2, and TR thresholds may all be set quite low, triggering more alerts and increasing human engagement. In contrast, a less critical CV task that labels objects in a scene may be configured with higher thresholds to limit the number or requests sent to the operator. Furthermore, these configurable parameters support dynamic learning based on human feedback.

In Fig. 10, an event is triggered when the underlying CV algorithm detects a potential victim at confidence level c and certainty r determining the actions taken. For example, a low confidence, high certainty detection results in no action, whereas, low confidence with low certainty leads triggers a request for human response. If the human confirms the sighting within MAX_WAIT, a "victim detected" alert is raised, and a subsequent action is executed (e.g., "track the victim"), otherwise no action is taken. The the CDG therefore specifies exactly when and where a human can provide feedback that overrides decisions previously made by the sUAS in its rapid decision-making mode.

A tug-of-war can occur when an sUAS makes a decision, the decision is overridden by the operator, and the sUAS then offers new information and retriggers the initial event. However, all feedback and new information is fed to the MAPE-K knowledge base and considered when computing confidence scores. In this case, for example, instead of only considering output from the CV pipeline, we also consider geolocation of the sighting and recent feedback to determine confidence levels. Confidence scores < TC1 are ignored, eventually (possibly temporarily) halting the event cycle for a specific victim sighting.

4.3.2 Human Mental Models and Calibrated Trust. Finally, we discuss the critical role of the human's conceptual model, which adds an additional layer to the way humans and machines coordinate their actions and calibrate trust levels [44, 122]. As depicted in Fig. 1, each human builds and maintains their own mental model of the current state of the mission and the capabilities of the machines, and the human's trust in the system is highly correlated with their current conceptual model. It would be näive to assume that this mental model is purely informed by explicit runtime models and GUIs that are intentionally designed to support human-machine partnerships. Instead, the human's conceptual model is informed by everything they know about the system, including their physical interactions with the machine in the physical world, their prior experiences with the system, and their knowledge of its underlying capabilities.

We illustrate this with a simple example from *Drone Response*. To provide flexibility in our multi-sUAS environment whilst avoiding in-air collisions, each sUAS must request and lease exclusive air tunnels that match their planned flight paths prior to any flight. During initial tests of the air-leasing system, before it was validated, we required status messages for each and every lease request and authorization, and we avidly observed each sUAS flight in order to quickly intervene if anything unexpected happened. However, now that this feature is validated, we do not need status messages about lease requests, nor do we want to be asked for permission each time an sUAS requests a new flight path. Instead, we now trust the system to perform this task autonomously, and the runtime air-leasing model is now entirely hidden from the humans, unless an explicit query is requested to view current air leases. This, and other similar examples, indicate that human trust is a clear driver in determining how human-machine roles and autonomy levels should be balanced at any time, and that degrees of autonomy and coordination should be adapted dynamically according to human preferences and comfort levels. *MAPE-K*$_{HMT}$ must therefore accommodate human requests for raising or lowering autonomy levels according to current degrees of calibrated trust.

4.4 Integrating HMT Models

While we have presented several examples of HMT models as independent entities, many of them share common data inputs (e.g., data read from sensors and software probes), or rely on data/outputs from other runtime models. For example, in order to generate meaningful explanations where multiple sUAS are involved, information is collected from the individual sUAS onboard runtime models. The Autonomy Explanation Model relies upon "adaptation notification events" provided by each individual sUAS, which in turn employs its own state machine. This information is then aggregated and post-processed, so that the Alert Triage Model (cf. Fig. 7) can select and display critical alerts from individual sUAS or global alerts from the system. While the knowledge base and supporting runtime models are distributed across the respective agents they share data as needed. This means that each sUAS retains full autonomy when a person is detected, and adaptation decisions are made onboard, but at the same time alerts are raised to allow human partners to participate in the decision-making process. This dependency upon accurate and appropriate information passing between the different models calls for careful design and planning to ensure that the right information is available at the right time in a potentially resource-constrained operating environment.

In Fig. 11 we provide an overview of the relations between several of the models discussed in this paper, as well as the sUAS and humans involved in the mission. Here we can see that some models (e.g., State-change Monitor, Alert Triage Model, Computer Vision) produce outputs that are used by several other components in the system. Similarly, other components (e.g., Alert Triage Model) require inputs that are generated by multiple runtime models. Three of the models produce outputs that are used by the GUI, which support human decision-making, albeit at far slower cadence than machine decisions. Finally, three of the models have outputs to sUAS which directly impacts the sUAS autonomy. Given these complex relations between models, the sampling and control times involved in each of the components must be carefully determined to avoid oversampling (potentially wasting resources) and undersampling (generating latencies and filtering effects such as aliasing). Time steps for sampling and control are determined as follows:

- *Human/Machine inputs:* For each model that produces an output for the human or machine, determine an upper bound for the time step, taking into account the relevance for the individual's autonomy (e.g., failsafe alerts in the UI need to be updated at least every second, CV requests can be generated no faster than every 10 seconds to give time for the human to respond, and flight mode commands for the sUAS need to be issued faster than 0.5 s in case of a tug-of-war scenario).
- *Inter-model dependencies:* For each model A that produces an output that is used by another runtime model B whose time step upper bound has already been estimated, determine the time step that is required for the model A to be in compliance with the input sampling time needed by model B. Repeat this step until all inter-model dependencies have been resolved. Note that this process works under the assumption that there are no cycles or deadlocks of runtime models that would prevent the generation of outputs for the human or sUAS.
- *Feasibility:* Once all the models have upper bounds for the time steps, verify that the human and sUAS inputs for the models can satisfy the sampling time requirements for the models (e.g., if the State-change Monitor is designed to receive status from the sUAS with at most 10 Hz given the current communication baud rate and available sensors, but the tug-of-war mitigator requires a faster rate during sensitive operations, then the required

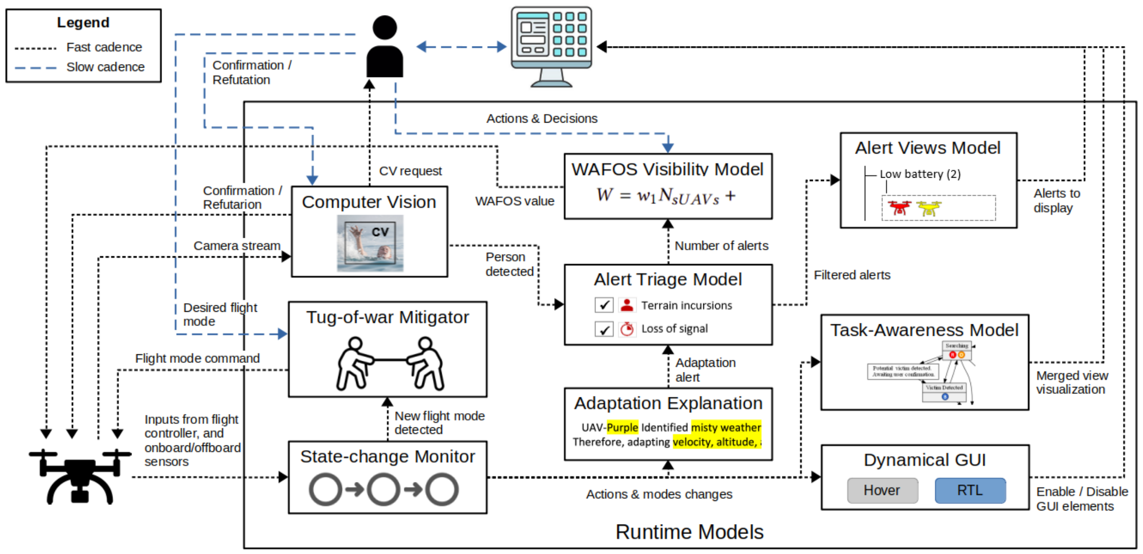


Fig. 11. Integration of the RunTime Data Probes, showing the fast and slow cadence relations.

sampling time is not feasible without making modifications). For the infeasible scenarios where the sUAS or human cannot produce the required sampling time, we have to revisit the *Human/Machine inputs* item and redo the procedure with higher upper bounds, trying to identify the cause of the high sampling requirements.

5 ANALYSIS AND TAKE-AWAYS

Our work can be positioned at the intersection of autonomous systems and human-machine teaming in general, and human-sUAS interaction in particular. As part of this, we have proposed augmenting the MAPE-K loop with dedicated support for HMT at each of its key phases. *MAPE-K_{HMT}* assumes that both humans and machines are capable of autonomous behavior and decision-making, and that mission goals are achieved jointly, through an interactive partnership. Based on our own experiences in integrating aspects of *MAPE-K_{HMT}* into our *Drone Response* system, we discuss important issues related to designing and deploying *MAPE-K_{HMT}* and report on potential challenges one has to consider along the way.

5.1 Requirements and Stakeholders

HMT systems inherently involve humans, and therefore, it is important to identify CRACK (Collaborative, Representative, Authorized, Committed, Knowledgeable) [18] stakeholders serving as direct users and domain experts. Engaging with stakeholders, who will ultimately become the human partners, helps to uncover interactions and expectations users have on the system. This, for example, requires carefully exploring the human-machine interactions related to mission-related scenarios [123]. Based on focus group interviews with emergency responders, as key stakeholders of our *Drone Response* system, we have observed varying levels of trust and sophistication in their expectations for working with sUAS as mission partners. This not only needs to be considered when eliciting requirements, but also needs to be accounted for in subsequent system design. For example, when asked to imagine working with autonomous sUAS, one firefighter said "The autonomous part is so new to us [. . .], how (would) we know that the drone is recognizing a person on the water?", and when presented with a simulation of a mission asked "Why is it (the sUAS) flying there?" Therefore, one key capability of HMT systems is related to the customizability of the user interaction components. Depending on the (technical) background and capabilities of the stakeholders and the tasks they will perform different functionality or decisions may need to be provided. This in turn affects runtime models, the type of information they need to provide, and the coordination mechanisms which largely determine the degrees of autonomy allowed on the sUAS. As a starting point for collecting HMT-related requirements, McDermott *et al.* [92] created a detailed elicitation process that included a list of key questions associated with each of the HMT factors [91]. For example, to understand "predictability" requirements, analysts must discover (a) automation goals, abilities, and limitations, (b) how the human partner's goals and priorities are tracked, (c) reliability of different automated tasks within different contexts, and (d) the types of changes that are expected to occur and trigger subsequent adaptations. The elicited requirements are subsequently analyzed to negotiate and reconcile trade-offs, and to identify and specify requirements that support human-machine interactions [111]. While our process is agnostic to specific techniques, the domains in which MAPE-K operates generally dictate that the requirements process includes a rigorous safety analysis (e.g., [40, 80, 127]), and determines that requirements should be specified sufficiently formally to capture timing and other performance constraints and/or to establish formal goal models. In many cases, the requirements specification (e.g., Goal Models, or state transition diagrams) provides the foundation for the respective runtime models [6, 14, 23].

In *MAPE-K$_{HMT}$*, it is particularly important to elicit requirements for tasks at the intersection between humans and machines, where human input or feedback is needed. These need to be carefully analyzed to avoid problems that arise from the mismatched cadence. This requires specifying what should happen if the human fails to provide timely input, as well as defining runtime monitoring capabilities with guarantees that necessary information can be provided to the user without excessive latency. This might include processing time for data aggregation or providing, and frequent updates to the data to avoid data becoming stale and outdated before a decision is made.

Integration and Flow of information through the HMT runtime models: *MAPE-K$_{HMT}$* emphasizes the importance of HMT-related runtime models. We, therefore, start the design process by taking an inventory of existing runtime models, identifying gaps where HMT requirements are not adequately supported by existing models, designing new runtime models as needed, and then finally composing models into workflows to service each HMT requirement. This involves assessing *who* (machine or human-role), *when*, and *where* each model will be used and updated, and *what* data sources are required as inputs (e.g., probes, message subscriptions, or human-initiated data and events). Furthermore, as actions are performed at different speeds, the required refresh frequencies must be determined for each constituent element of each runtime model, and a system-wide plan established so that each collected data attribute satisfies the refresh frequencies of all relevant models.

5.2 Testing and Deployment of HMT systems

The final steps involve implementing the system, verifying that required runtime models, UIs, and supporting features are implemented as intended, and finally validating that the deployed system satisfies its stated requirements and supports the desired HMT. As human-machine interaction plays a central role in our work, Human-Machine Interfaces are of particular importance. This not only includes graphical user interfaces (GUIs), but also hardware interfaces. These interfaces need to be designed to provide interactive support for humans. Depending on the runtime model, the data that is displayed, and the input that is expected from the human, different means of conveying information or requesting input may be necessary. Examples include the previously discussed Alert and Adaptation Explanation

Models but further include hardware interfaces such as Joysticks or radio controllers. Mission- and safety-critical information needs to be provided to the human in a timely manner without creating cognitive overload, and must not be “hidden” in sub-menus or views that require multiple steps to access [44].

In testing and deploying HMT systems such as *Drone Response*, it is important to validate not just the functional requirements and typical non-functional ones (NFRs) such as safety, performance, and basic usability; but to validate NFRs related to each of the HMT factors such as calibrated trust, augmented cognition, and transparency.

6 THREATS TO VALIDITY

Our work is subject to three primary threats to validity. First, all of the runtime models described in this paper are designed to support HMT in our own *Drone Response* system, representing a multi-agent, multi-human system operating in a mid-level safety-critical domain [85, 125]. As types of interactions are influenced by the operating domain, the human-machine partnerships in *Drone Response* may differ significantly from other application domains – for example, those with a single operator or a single machine, or operating in a higher or lower level of safety-criticality. Therefore, instead of proposing a specific set of HMT-related runtime models, *MAPE-K*$_{HMT}$ provides a simple approach for identifying, developing, and integrating context-specific models in support of transparency, augmented cognition, and coordination goals, which are known to be applicable across diverse operating environments [77, 91, 92]. Further, our approach could be easily extended – for example, by integrating a more formal safety analysis for more critical domains. Given our own emphasis on the sUAS domain, we leave this as an open research challenge for *MAPE-K*$_{HMT}$ to be applied to more varied systems.

Second, developing *Drone Response* at a professional level for real-world deployment is extremely time-consuming. Therefore, while many of our runtime models have been prototyped and/or employed within *Drone Response*, several are still only conceptualized. In particular we have implemented the state change monitor [33, 64], task-awareness model [33, 64], and dynamic GUI models, and have partially implemented the two alerts models [3], the decision conflict graphs [1] and deployed other features mentioned in this paper such as the computer vision pipeline and air leasing. However, other models such as the tug-of-war mitigator and the WAFOS model are planned but not yet implemented into the full version of *Drone Response*. This paper has also discussed the creation of shared data probes that support the monitoring frequencies of each individual HMT model. We have developed monitoring capabilities to achieve this, and models, such as the state change monitor, have been implemented as microservices that seamlessly forward their outputs via MQTT to other models; however, full deployment of monitors and the integration of all models is still ongoing.

Finally, *Drone Response* has been deployed in the physical world, where our own team has experienced the challenges of multi-sUAS operations. However, formal evaluations of HMT user interfaces have relied upon user studies conducted in *Drone Response*’s simulator (e.g., [2, 3, 126]). While the *Drone Response* GUI is identical for both physical and simulated sUAS, and our past experiences have shown that many findings from these user studies have held when deployed in physical field tests, it is impossible to replicate the full immersive experience of operating multiple sUAS in the real-world through a simulated environment. An sUAS crash or fly-away in a simulator is entirely stress-free; whereas a similar event in the physical world has real consequences and therefore radically changes the way humans may wish to interact with the system and the levels of trust they may be willing to place in sUAS autonomy. Therefore, further experiments targeted specifically at HMT need to be conducted in real-world settings where humans collaborate with sUAS under far more noisy, volatile, and potentially stressful conditions.

7 RELATED WORK

Given that we combine concepts from both HCI and self-adaptive systems in conjunction with safety-related aspects, we discuss related work pertaining to human-machine collaboration, human engagement, and safety assurance in the context of self-adaptive systems.

Human-Machine Collaboration: A significant body of work, primarily in the HCI community has focused on designing UIs to support situational awareness of Cyber-Physical Systems (e.g., [2, 5, 17, 106, 117, 122]). The focus has primarily been on enabling users to perceive, understand, and make effective decisions [44, 45]. While this related work supports HMT goals, it puts little emphasis on integration with underlying runtime models which are needed in order to deliver accurate, timely, and often aggregated data for use in the UIs. Kephart advocated for increased interactivity between humans and users in adaptive decision-making [73]; however, their examples were all taken from domains in which humans had plenty of time to consider their decisions. Integrating HMT into the MAPE-K loop for real-time robotics systems introduces additional, and very challenging, timing constraints.

In the HMT domain, researchers have explored many facets of human-machine teaming. For example, Klein *et al.* [75] identified challenges associated with achieving shared goals, preventing breakdowns in team coordination, and fostering communication and collaboration. Furthermore, Schmid *et al.* [118] studied ways to adjust system automation in complex, safety-critical environments in order to better support human operators. While their work has significant relevance to *MAPE-K$_{HMT}$*, it perceives humans as “operators” rather than true partners.

Calhoun *et al.* [24] proposed a flexible architecture that allows the degree of automation to vary according to the human’s current engagement and workload. These goals are reflected in our discussion on adaptation in *MAPE-K$_{HMT}$*, which embraces the notion of adapting for improved human collaboration and performance. To increase context awareness in order to better engage humans in the decision-making process, Li *et al.* [82] proposed a formal framework based on probabilistic reasoning to determine when advanced notifications are useful for humans interacting with self-adaptive systems. Their work specifically addresses the cadence problem in which humans may be required to respond quickly but require “thinking time”. However, their evaluation was conducted in a robotics goods delivery domain, which has a lower decision cadence than a multi-sUAS system.

Self-Adaptation & Models@Runtime : Various researchers have proposed sUAS-related self-adaptation frameworks. For example, Braberman *et al.* [19, 20] presented a MAPE-K reference architecture for unmanned aerial vehicles, while Yu *et al.* [137] proposed a self-adaptive framework for sUAS forensics. Neither of these explored the HMT aspects of adaptation. Finally, in more closely related work Lim *et al.* [83] explored ways to adapt human-robot interactions in a multi-sUAS system, with a focus on modulating automation support according to the cognitive states of the human operator. This creates a subtle, but important difference from our *MAPE-K$_{HMT}$* goal, as it centers primarily around the needs of the operator rather than optimizing teamwork goals. Ignatius and Bahsoon [63] have proposed the SOA-HITLCPS, an ontology model for human-machine collaboration in service-oriented, self-adaptive systems. Similar to our work, their model focuses on how humans and machines can help each other and extend their capabilities, focusing on “human-as-a-service” capabilities. While they have created a comprehensive ontology, describing human capabilities, with *MAPE-K$_{HMT}$* we focus on the different types of models and more specifically, their use at runtime to augment both human and machine capabilities. Parra-Ullauri *et al.* [101] proposed a temporal feedback mechanism for MAPE-K architectures to provide historical information for HiTL self-adaptive systems. Similarly, our work allows for the ability to include temporal data in human-machine teaming efforts at runtime as demonstrated with our alert and weather models. However, the authors present a singular feedback and explainability layer based on temporal history, whereas we combine historical data with categorized HMT models for increased transparency. Other authors explore self-adaptive MAPE-K frameworks with human-in-the-loop capabilities for general user applications. For example, Evers *et al.* [47] augment a system to allow users to initialize goals at runtime for application supervision along with methods to override or self-adaptions by measuring utility. While this is similar to our workload monitoring, their system does not have detailed explanation capabilities built into each portion of the MAPE-K feedback loop (as seen in our adaptation explanation model), or diverse runtime model evaluations. Our system incorporates a variety of user-centered interactions that support self-adaption which may include alerts, explanations, and runtime modeling for each step of a complicated mission [82] focus on preparatory notifications, introducing a formal framework for self-adaptive systems involving human operators. Their intention is to raise human attention via task notifications. . In the context of Digital Twins, Yigitbas *et al.* [136] presented an approach for enhancing HiTL via virtual reality (VR) interfaces. As part of this, their framework uses head-mounted displays in two different scenarios, where humans can either control adaptations through VR interactions or specify a goal state in conjunction with an AI planner. Going even one step further, Lloyd *et al.* [84] present a concept of a self-adaptive system using brain-computer interaction. While this approach facilitates immersive human involvement in the self-adaptation loop, their focus differs from our *MAPE-*K_{HMT} insofar as we focus on the various different interaction types during complex sUAS missions and the underlying runtime data that is collected and processed.

Runtime Monitoring and Adaptive Monitoring of Adaptive Systems: To collect runtime data from a system, which serves as the basis for further analysis and decision-making, various different approaches have been explored. This, for example, includes adaptive sampling techniques [12, 41, 96], as well as (model-based) adaptive monitoring [21, 22] In this context, goal models are frequently used to capture the state of a system at runtime and support (runtime) adaptation and decision-making [11, 103]. Reynolds *et al.* [109] present an approach for tracking the behavior of self-adaptive systems using provenance graphs and a runtime model to analyze and explain the runtime behavior of a system. Similarly for robotic applications, a model-based framework [35] by Corbato *et al.* adapts robot control architectures at runtime. It targets ROS-based systems and uses the MAPE-K loop to trigger reconfigurations of the system. Hili *et al.* [58] proposed a model-based architecture for interactive runtime monitoring using model-based techniques and Sakizloglou *et al.* [114] used runtime models supported by the Viatra model query engine [15] to check constraints as part of adaptation rules. Other frameworks, such as Plato-RE [105] for generating monitoring configurations at run time in response to changing conditions or DYNAMICO [128] provide explicit support for runtime data collection, analysis, and subsequent decision-making. With our proposed *MAPE-*K_{HMT} extension and the respective models, we leverage these existing capabilities and add the additional dimension of human interaction and, more importantly, collaborative decision-making to these systems.

Safety Assurance: Finally, several papers have explored self-adaptation and/or human interactions in the sUAS and autonomous systems domain, in conjunction with safety-related aspects. As part of our own previous work [127], we have identified hazards explicitly related to human-sUAS interaction, and have created a dataset of hazard trees with corresponding mitigation examples. Similarly, Miller *et al.* [93] explored different techniques by which users could interact with multiple sUAS. Hagele *et al.* [56] introduced a simplified real-time environmental situation risk assessment approach for determining the reliability of autonomous systems. They further describe emergency behavior control for autonomous systems' safe behavior assurance. McAree *et al.* [90] discussed the development of a semi-autonomous inspection drone that can maintain a fixed distance and relative heading to a particular object, as well as a Model-Based Design (MBD) framework enabling any candidate control system to be tested in a high-fidelity simulation environment prior to any real-world flights. Related to self-adaptive systems, Jahan *et al.* [67] have created MAPE-SAC a framework for incorporating Safety Assurance Cases into MAPE-K, by introducing an additional MAPE-SAC loop. Other work by Cheng *et al.* [31] has explored the application of Safety Assurance Cases for ROS-based systems allowing runtime adaptation. Furthermore, contract-based has also been explored in the context of Safety Assurance, for example, Kelly developed Modular GSN to handle the situation where one goal needs to be supported by a goal from another module linking multiple safety case modules [72]. And work by Denney and Pai [38, 39] explored several aspects of modular safety cases for UAVs, facilitating the capture and maintenance of safety-related UAV behavior. However, most of the work in this area focuses on either system-related hazards and/or Human-on-the-loop systems rather than active collaboration and HMT. Furthermore, in the context of autonomous driving, human interaction/collaboration and safety assurance has been explored extensively [1, 10, 60] which could serve as a foundation to further extend our own runtime models.

8 CONCLUSION

In this paper, we have described our approach for augmenting MAPE-K to support partnerships between humans and machines – both of which are capable of fully autonomous behavior. We have shown how *MAPE-K$_{HMT}$* can address the three HMT goals of transparency, augmented cognition, and coordination by delivering a self-adaptation framework in which humans and machines collaborate together to achieve a common goal. We have mapped HMT factors to the MAPE-K phases and then used a series of examples to illustrate how carefully designed and integrated runtime models enable meaningful support for HMT. The runtime models, drawn from our own *Drone Response* system, have provided examples of both human-to-machine, and machine-to-human transparency, cognition, and coordination. Whereas prior work on HMT has focused primarily on ways in which the human interacts with the machine, we have equally emphasized models that recognize machine autonomy – for example by providing awareness to the machine of the human's current availability and workload. Rather than diminishing the autonomy of machines, HMT draws upon the autonomous abilities of both humans and machines to deliver an even stronger solution that allows autonomy levels to be autonomously adapted through the mission based on human trust levels, availability, and machine capabilities.

In conducting this work we have identified three key challenges. The first challenge stems from the very different cadences at which humans and machines operate. This discrepancy creates the potential for humans to make decisions based upon stale data that no longer reflects current state. Existing HMT solutions, such as turn-taking [28], are only effective in scenarios that can tolerate slower response times. In *Drone Response* we partially address this challenge by dynamically and frequently adapting the UIs to reflect currently available human interaction options. Furthermore, from a technical perspective we have implemented a reliable messaging system that ensures that messages are only sent to an sUAs when it is in a state to handle them; however, the problem is a complex one which warrants further exploration.

The second challenge relates to designing and supporting the integrated *MAPE-K$_{HMT}$* environment. Diverse HMT runtime models, needed for effective human-machine teaming, must be identified, designed, integrated, deployed, and effectively maintained at runtime. We have therefore shown how the runtime models described in this paper can be integrated to optimize monitoring rates and information flow, and how models that directly support human interfaces need to be carefully designed to account for the slower cadence of human interactions.

The third key challenge relates to how humans and machines reach consensus and deal with conflicts as they partner together in a mission. We have discussed our preliminary rule-driven approach for handling specific classes of conflicts, related to CV or onboard anomaly detection and mitigation; however, more intelligent approaches reflecting advances in psychology, sociology, CPS, and AI are emerging and better able to handle unforeseen types of conflicts. We plan to explore this issue in future work.

Beyond these three challenges, much additional work is needed to realize the vision of trusted, coordinated, and accountable teams of humans and machines; however, *Drone Response* has been implemented, deployed, and tested in the physical world as a multi-sUAS system supported by the HMT runtime models described in this paper. Deploying and flying *Drone Response* has enabled us to validated the ideas discussed in this paper, providing critical insights that have guided decisions for enhanced transparency, augmented cognition, and coordination. Our ongoing work will therefore focus on utilizing the *Drone Response* system to support continued field tests and studies with physical sUAS and end-users. Our findings will guide the evolution of the *Drone Response* system with the aim of tackling many of the open challenges described in this paper in order to achieve increasingly higher levels of HMT teaming.

ACKNOWLEDGMENT

The work in this paper has been funded by the US National Science Foundation (NSF) under Grant# CNS-1931962, a Fullbright Scholarship funding Sebastian Zudaire's visit to the University of Notre Dame, and by the Linz Institute of Technology (LIT-2019-7-INC-316).

REFERENCES

[1] Sophia J. Abraham, Zachariah Carmichael, Sreya Banerjee, Rosaura G. VidalMata, Ankit Agrawal, Md Nafee Al Islam, Walter J. Scheirer, and Jane Cleland-Huang. 2021. Adaptive Autonomy in Human-on-the-Loop Vision-Based Robotics Systems. In *1st IEEE/ACM Workshop on AI Engineering - Software Engineering for AI, WAIN@ICSE 2021, Madrid, Spain, May 30-31, 2021*. IEEE, 113–120. https://doi.org/10.1109/WAIN52551.2021.00025

[2] Ankit Agrawal, Sophia J. Abraham, Benjamin Burger, Chichi Christine, Luke Fraser, John M. Hoeksema, Sarah Hwang, Elizabeth Travnik, Shreya Kumar, Walter J. Scheirer, Jane Cleland-Huang, Michael Vierhauser, Ryan Bauer, and Steve Cox. 2020. The Next Generation of Human-Drone Partnerships: Co-Designing an Emergency Response System. In *Proc. of CHI Conference on Human Factors in Computing Systems*. ACM, New York, 1–13. https://doi.org/10.1145/3313831.3376825

[3] Ankit Agrawal and Jane Cleland-Huang. 2021. Explaining Autonomous Decisions in Swarms of Human-on-the-Loop Small Unmanned Aerial Systems. In *Proceedings of the Ninth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2021, virtual, November 14-18, 2021*, Ece Kamar and Kurt Luther (Eds.). AAAI Press, 15–26. https://ojs.aaai.org/index.php/HCOMP/article/view/18936

[4] Ankit Agrawal and Jane Cleland-Huang. 2021. RescueAR: Augmented Reality Supported Collaboration for UAV Driven Emergency Response Systems. *CoRR* abs/2110.00180 (2021). arXiv:2110.00180 https://arxiv.org/abs/2110.00180

[5] Ankit Agrawal, Jane Cleland-Huang, and Jan-Philipp Steghöfer. 2020. Model-Driven Requirements for Humans-on-the-Loop Multi-UAV Missions. In *10th IEEE International Model-Driven Requirements Engineering, MoDRE@RE 2020, Zurich, Switzerland, August 31, 2020*. IEEE, 1–10. https://doi.org/10.1109/MoDRE51215.2020.00007

[6] Amal Ahmed Anda and Daniel Amyot. 2019. Arithmetic Semantics of Feature and Goal Models for Adaptive Cyber-Physical Systems. In *27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019*, Daniela E. Damian, Anna Perini, and Seok-Won Lee (Eds.). IEEE, 245–256. https://doi.org/10.1109/RE.2019.00034

[7] Paolo Arcaini, Raffaela Mirandola, Elvinia Riccobene, Patrizia Scandurra, Alberto Arrigoni, Daniele Bosc, Federico Modica, and Rita Pedercini. 2020. Smart home platform supporting decentralized adaptive automation control. In *Proc. of 35th Annual ACM Symposium on Applied Computing*. ACM, New York, 1893–1900.

[8] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. 2015. Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation. In *Proc. of 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 13–23. https://doi.org/10.1109/SEAMS.2015.10

[9] Uwe Aßmann, Sebastian Götz, Jean-Marc Jézéquel, Brice Morin, and Mario Trapp. 2014. *A Reference Architecture and Roadmap for Models@run.time Systems*. Springer International Publishing, Cham, 1–18. https://doi.org/10.1007/978-3-319-08915-7_1

[10] Stuart Ballingall, Majid Sarvi, and Peter Sweatman. 2020. Safety assurance concepts for automated driving systems. *SAE International Journal of Advances and Current Practices in Mobility* 2, 2020-01-0727 (2020), 1528–1537.

[11] Luciano Baresi, Liliana Pasquale, and Paola Spoletini. 2010. Fuzzy goals for requirements-driven adaptation. In *Proc. of the 18th IEEE Int'l Requirements Engineering Conf.* IEEE, 125–134.

[12] Ezio Bartocci, Radu Grosu, Atul Karmarkar, Scott A Smolka, Scott D Stoller, Erez Zadok, and Justin Seyster. 2012. Adaptive runtime verification. In *Proc. of the Int'l Conf. on Runtime Verification*. Springer, 168–182.

[13] Nelly Bencomo and Amel Belaggoun. 2014. A world full of surprises: bayesian theory of surprise to quantify degrees of uncertainty. In *Proc. of the 36th International Conference on Software Engineering, Companion Proceedings.* ACM, 460–463. https://doi.org/10.1145/2591062.2591118

[14] Nelly Bencomo, Sebastian Götz, and Hui Song. 2019. Models@run.time: a guided tour of the state of the art and research challenges. *Software and Systems Model* 18, 5 (2019), 3049–3082. https://doi.org/10.1007/s10270-018-00712-x

[15] Gábor Bergmann, István Dávid, Ábel Hegedüs, Ákos Horváth, István Ráth, Zoltán Ujhelyi, and Dániel Varró. 2015. Viatra 3: A reactive model transformation platform. In *Proc. of the Int'l Conf. on Theory and Practice of Model Transformations.* Springer, 101–110.

[16] Jason M Bindewald, Christina F Rusnock, and Michael E Miller. 2018. Measuring human trust behavior in human-machine teams. In *Proc. of the AHFE 2017 International Conference on Human Factors in Simulation and Modeling.* Springer, 47–58.

[17] Francesco N Biondi, Monika Lohani, Rachel Hopman, Sydney Mills, Joel M Cooper, and David L Strayer. 2018. 80 MPH and out-of-the-loop: Effects of real-world semi-automated driving on driver workload and arousal. In *Proc. of Human Factors and Ergonomics Society Annual Meeting*, Vol. 62. SAGE Publications, 1878–1882.

[18] Barry W. Boehm and Richard Turner. 2004. Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. In *Proc. of the 26th International Conference on Software Engineering.* IEEE Computer Society, 718–719. https://doi.org/10.1109/ICSE.2004.1317503

[19] Victor Braberman, Nicolas D'Ippolito, Jeff Kramer, Daniel Sykes, and Sebastian Uchitel. 2015. Morph: A reference architecture for configuration and behaviour self-adaptation. In *Proc. of 1st International Workshop on Control Theory for Software Engineering.* ACM, New York, 9–16.

[20] Victor Braberman, Nicolas D'Ippolito, Jeff Kramer, Daniel Sykes, and Sebastian Uchitel. 2017. An Extended Description of MORPH: A Reference Architecture for Configuration and Behaviour Self-Adaptation. In *Software Engineering for Self-Adaptive Systems III. Assurances.* Springer International Publishing, Cham, 377–408.

[21] Thomas Brand and Holger Giese. 2018. Towards software architecture runtime models for continuous adaptive monitoring. In *Proc. of the 13th Int'l WS on Models@run.time.* 72–77.

[22] Thomas Brand and Holger Giese. 2019. Modeling approach and evaluation criteria for adaptable architectural runtime model instances. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS).* IEEE, 227–232.

[23] Jennifer Brings, Marian Daun, Thorsten Weyer, and Klaus Pohl. 2020. Goal-based configuration analysis for networks of collaborative cyber-physical systems. In *Proc. of 35th Annual ACM Symposium on Applied Computing.* ACM, New York, 1387–1396.

[24] Gloria L. Calhoun, Heath A. Ruff, Kyle J. Behymer, and Elizabeth Marie Mersch. 2017. Operator-Autonomy Teaming Interfaces to Support Multi-Unmanned Vehicle Missions. In *Advances in Human Factors in Robots and Unmanned Systems.* Springer, 113–126.

[25] Radu Calinescu, Naif Alasmari, and Mario Gleirscher. 2021. Maintaining driver attentiveness in shared-control autonomous driving. In *Proc. of the 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* IEEE, 90–96.

[26] Benjamin Camblor, Jean-Marc Salotti, Charles Fage, and David Daney. 2022. Degraded situation awareness in a robotic workspace: accident report analysis. *Theoretical Issues in Ergonomics Science* 23, 1 (2022), 60–79.

[27] Lorena Castañeda, Norha M. Villegas, and Hausi A. Müller. 2014. Self-Adaptive Applications: On the Development of Personalized Web-Tasking Systems. In *Proc. of 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* ACM, New York, 49–54.

[28] Crystal Chao and Andrea Thomaz. 2016. Timed Petri nets for fluent turn-taking over multimodal interaction resources in human-robot collaboration. *The International Journal of Robotics Research* 35, 11 (2016), 1330–1353.

[29] Jessie Chen, Michael Barnes, and Michelle Harper-Sciarini. 2011. Supervisory Control of Multiple Robots: Human-Performance Issues and User-Interface Design. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 41 (07 2011), 435–454. https://doi.org/10.1109/TSMCC.2010.2056682

[30] Tao Chen and Rami Bahsoon. 2017. Self-Adaptive and Online QoS Modeling for Cloud-Based Software Services. *IEEE Transactions on Software Engineering* 43, 5 (2017), 453–475. https://doi.org/10.1109/TSE.2016.2608826

[31] Betty HC Cheng, Robert Jared Clark, Jonathon Emil Fleck, Michael Austin Langford, and Philip K McKinley. 2020. AC-ROS: assurance case driven adaptation for the robot operating system. In *Proc. of the 23rd acm/ieee international conference on model driven engineering languages and systems.* 102–113.

[32] Muhammed Tawfiq Chowdhury and Jane Cleland-Huang. 2023. Engineering Challenges for AI-Supported Computer Vision in Small Uncrewed Aerial Systems. In *Proc. of the 2nd International Conference on AI Engineering – Software Engineering for AI.* IEEE, Melbourne, Victoria, Australia, 12 pages.

[33] Jane Cleland-Huang, Ankit Agrawal, Md Nafee Al Islam, Eric Tsai, Maxime Van Speybroeck, and Michael Vierhauser. 2020. Requirements-driven configuration of emergency response missions with small aerial vehicles. In *Proc. of 24th ACM International Systems and Software Product Line Conference.* ACM, New York, 1–12. https://doi.org/10.1145/3382025.3414950

[34] Jane Cleland-Huang, Ankit Agrawal, Michael Vierhauser, Michael Murphy, and Mike Prieto. 2022. Extending MAPE-K to support Human-Machine Teaming. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2022, Pittsburgh, PA, USA, May 22-24, 2022.* IEEE, 120–131. https://ieeexplore.ieee.org/document/9800015

[35] Carlos Hernandez Corbato, Darko Bozhinoski, Mario Garzon Oviedo, Gijs van der Hoorn, Nadia Hammoudeh Garcia, Harshavardhan Deshpande, Jon Tjerngren, and Andrzej Wasowski. 2020. MROS: Runtime Adaptation For Robot Control Architectures. *arXiv:2010.09145* (2020).

[36] Praveen Damacharla, Ahmad Y. Javaid, Jennie J. Gallimore, and Vijay K. Devabhaktuni. 2018. Common Metrics to Benchmark Human-Machine Teams (HMT): A Review. *IEEE Access* 6 (2018), 38637–38655. https://doi.org/10.1109/ACCESS.2018.2853560

[37] Rogério de Lemos. 2020. Human in the Loop: What is the Point of No Return?. In *Proc. of IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* ACM, New York, 165–166. https://doi.org/10.1145/3387939.3391597

[38] Ewen Denney and Ganesh Pai. 2012. A lightweight methodology for safety case assembly. In *Proc. of the International Conference on Computer Safety, Reliability, and Security*. Springer, 1–12.

[39] Ewen Denney, Ganesh Pai, and Ibrahim Habli. 2015. Dynamic safety cases for through-life safety assurance. In *Proc. of the 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 587–590.

[40] Ewen Denney, Ganesh Pai, and Josef Pohl. 2012. Heterogeneous aviation safety cases: Integrating the formal and the non-formal. In *Proc. of the 17th International Conference on Engineering of Complex Computer Systems*. IEEE, 199–208.

[41] Rui Ding, Hucheng Zhou, Jian-Guang Lou, Hongyu Zhang, Qingwei Lin, Qiang Fu, Dongmei Zhang, and Tao Xie. 2015. Log2: A Cost-Aware Logging Mechanism for Performance Diagnosis. In *Proc. of the 2015 USENIX Technical Conference*. 139–150.

[42] Dronology. 2020. Research Incubator and Dataset. https://dronology.info. [Last accessed 01-01-2022].

[43] F.E. Emery and E.L. Trist. 1960. Socio-technical systems. In *In: Churchman, C.W., Verhulst, M. (Eds.), Management Science Models and Techniques*, Vol. 9. Pergamon, 83–97.

[44] Mica R. Endsley. 2011. *Designing for Situation Awareness: An Approach to User-Centered Design, Second Edition* (2nd ed.). CRC Press, Inc., Boca Raton, FL, USA.

[45] Mica R Endsley. 2017. Autonomous driving systems: A preliminary naturalistic study of the Tesla Model S. *Journal of Cognitive Engineering and Decision Making* 11, 3 (2017), 225–238.

[46] Naeem Esfahani, Eric Yuan, Kyle R. Canavera, and Sam Malek. 2016. Inferring Software Component Interaction Dependencies for Adaptation Support. *ACM Transactions on Autonomous and Adaptive Systems* 10, 4 (2016), 26:1–26:32. https://doi.org/10.1145/2856035

[47] Christoph Evers, Romy Kniewel, Kurt Geihs, and Ludger Schmidt. 2014. The user in the loop: Enabling user participation for self-adaptive applications. *Future Generation Computer Systems* 34 (2014), 110–123. https://doi.org/10.1016/j.future.2013.12.010 Special Section: Distributed Solutions for Ubiquitous Computing and Ambient Intelligence.

[48] Antonio Filieri, Giordano Tamburrelli, and Carlo Ghezzi. 2016. Supporting Self-Adaptation via Quantitative Verification and Sensitivity Analysis at Run Time. *IEEE Transactions on Software Engineering* 42, 1 (2016), 75–99. https://doi.org/10.1109/TSE.2015.2421318

[49] Joel E Fischer, Chris Greenhalgh, Wenchao Jiang, Sarvapali D Ramchurn, Feng Wu, and Tom Rodden. 2021. In-the-loop or on-the-loop? Interactional arrangements to support team coordination with a planning agent. *Concurrency and Computation: Practice and Experience* 33, 8 (2021), e4082.

[50] Flight Safety Foundation. 2019. Preliminary Report B737-800MAX. https://flightsafety.org/preliminary-report-b737-800max-et-avj. [Last accessed 01-01-2022].

[51] Lex Fridman, Bryan Reimer, Bruce Mehler, and William T Freeman. 2018. Cognitive load estimation in the wild. In *Proc. of 2018 CHI Conference on Human Factors in Computing Systems*. ACM, New York, 1–9.

[52] Diego Gambetta. 2020. A Conceptual Reference Model for Human as a Service Provider in Cyber Physical Systems. *Department of Sociology, University of Oxford* (2020), 213–237. http://www.sociology.ox.ac.uk/papers/gambetta213-237.pdf

[53] Antoine Gaume, Gérard Dreyfus, and François-Benoît Vialatte. 2019. A cognitive brain–computer interface monitoring sustained attentional variations during a continuous task. *Cognitive Neurodynamics* 13, 3 (2019), 257–269.

[54] Ilias Gerostathopoulos and Evangelos Pournaras. 2019. TRAPPed in traffic? A self-adaptive framework for decentralized traffic optimization. In *Proc. of 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 32–38.

[55] Holger Giese, Nelly Bencomo, Liliana Pasquale, Andres Ramirez, Paola Inverardi, Sebastian Wätzoldt, and Siobhán Clarke. 2014. *Living with Uncertainty in the Age of Runtime Models*. 47–100. https://doi.org/10.1007/978-3-319-08915-7_3

[56] Georg Hägele and Dirk Söffker. 2017. A simplified situational environment risk and system reliability assessment for behavior assurance of autonomous and semi-autonomous aerial systems: A simulation study. In *Proc. of the 2017 Int'l Conf. on Unmanned Aircraft Systems*. IEEE, 951–960.

[57] Sandra G Hart. 1986. NASA task load index (TLX). (1986).

[58] Nicolas Hili, Mojtaba Bagherzadeh, Karim Jahed, and Juergen Dingel. 2020. A model-based architecture for interactive run-time monitoring. *Software and Systems Modeling* (2020), 1–23.

[59] Barbara Christine Hoekenga. 2007. Mind over machine : what Deep Blue taught us about chess, artificial intelligence, and the human spirit.

[60] Chao Huang, Hailong Huang, Junzhi Zhang, Peng Hang, Zhongxu Hu, and Chen Lv. 2021. Human-machine Cooperative Trajectory Planning and Tracking for Safe Automated Driving. *IEEE Transactions on Intelligent Transportation Systems* (2021).

[61] Muhammad Usman Iftikhar, Gowri Sankar Ramachandran, Pablo Bollansée, Danny Weyns, and Danny Hughes. 2017. Deltaiot: A self-adaptive internet of things exemplar. In *Proc. of 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 76–82.

[62] Didac Gil De La Iglesia and Danny Weyns. 2015. MAPE-K formal templates to rigorously design behaviors for self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 10, 3 (2015), 1–31.

[63] Hargyo TN Ignatius and Rami Bahsoon. 2021. A Conceptual Reference Model for Human as a Service Provider in Cyber Physical Systems. In *Proc. of the 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 1–10.

[64] Md Nafee Al Islam, Muhammed Tawfiq Chowdhury, Ankit Agrawal, Michael Murphy, Raj Mehta, Daria Kudriavtseva, Jane Cleland-Huang, Michael Vierhauser, and Marsha Chechik. 2023. Configuring mission-specific behavior in a product line of collaborating Small Unmanned Aerial Systems. *J. Syst. Softw.* 197 (2023), 111543. https://doi.org/10.1016/j.jss.2022.111543

[65] Md Nafee Al Islam, Yihong Ma, Pedro Alarcon Granadeno, Nitesh V. Chawla, and Jane Cleland-Huang. 2022. RESAM: Requirements Elicitation and Specification for Deep-Learning Anomaly Models with Applications to UAV Flight Controllers. In *30th IEEE International Requirements Engineering Conference, RE 2022, Melbourne, Australia, August 15-19, 2022*. IEEE, 153–165. https://doi.org/10.1109/RE54965.2022.00020

[66] Chen J. Y., Procci K., Boyce M., Wright J.and Garcia A., and Barnes M. 2014. Situation Awareness–Based Agent Transparency. *Report No. ARL-TR-6905. Aberdeen Proving Ground, MD: U.S. Army Research Laboratory.* (2014).

[67] Sharmin Jahan, Matthew Pasco, Rose Gamble, Philip McKinley, and Betty Cheng. 2019. MAPE-SAC: A framework to dynamically manage security assurance cases. In *Proc. of the 4th International Workshops on Foundations and Applications of Self* Systems*. IEEE, 146–151.

[68] Pooyan Jamshidi, Javier Cámara, Bradley Schmerl, Christian Käestner, and David Garlan. 2019. Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots. In *Proc. of 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 39–50.

[69] Yuchen Jiang, Shen Yin, and Okyay Kaynak. 2018. Data-driven monitoring and safety control of industrial cyber-physical systems: Basics and beyond. *IEEE Access* 6 (2018), 47374–47384.

[70] Jianxin Jiao, Feng Zhou, Nagi Gebraeel, and Vincent Duffy. 2020. Towards augmenting cyber-physical-human collaborative cognition for human-automation interaction in complex manufacturing and operational environments. *International Journal of Production Research* 58 (03 2020), 1–23. https://doi.org/10.1080/00207543.2020.1722324

[71] Donia Kateb, Nicola Zannone, Assaad Moawad, Patrice Caire, Grégory Nain, Tejeddine Mouelhi, and Yves Le Traon. 2015. Conviviality-Driven Access Control Policy. *RE Journal* 20 (11 2015). https://doi.org/10.1007/s00766-014-0204-0

[72] T. P. Kelly. 2001. *Concepts and Principles of Compositional Safety Cases, COMSA/2001/1/1.* Technical Report.

[73] Jeffrey Kephart. 2020. Keynote: Viewing Autonomic Computing through the Lens of Embodied Artificial Intelligence: A Self-Debate. In *Proc. of the 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE.

[74] J.O. Kephart and D.M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (2003), 41–50. https://doi.org/10.1109/MC.2003.1160055

[75] Glen Klien, David D Woods, Jeffrey M Bradshaw, Robert R Hoffman, and Paul J Feltovich. 2004. Ten challenges for making automation a" team player" in joint human-agent activity. *IEEE Intelligent Systems* 19, 6 (2004), 91–95.

[76] Torkel Klingberg. 2009. *The overflowing brain: Information overload and the limits of working memory.* Oxford University Press.

[77] Margarita Konaev and Husanjot Chahal. 2021. Building trust in human-machine teams. https://www.brookings.edu/techstream/building-trust-in-human-machine-teams. [Last accessed 01-01-2022].

[78] Jeamin Koo, Jungsuk Kwac, Wendy Ju, Martin Steinert, Larry Leifer, and Clifford Nass. 2015. Why did my car just do that? Explaining semi-autonomous driving actions to improve driver understanding, trust, and performance. *International Journal on Interactive Design and Manufacturing (IJIDeM)* 9, 4 (2015), 269–275.

[79] Shitij Kumar, Celal Savur, and Ferat Sahin. 2020. Survey of Human-Robot Collaboration in Industrial Settings: Awareness, Intelligence, and Compliance. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* PP (12 2020), 1–18. https://doi.org/10.1109/TSMC.2020.3041231

[80] Nancy G. Leveson. 2011. *The Use of Safety Cases in Certification and Regulation.* Technical Report. MIT.

[81] Nianyu Li, Sridhar Adepu, Eunsuk Kang, and David Garlan. 2020. Explanations for human-on-the-loop: A probabilistic model checking approach. In *Proc. of IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, New York, 181–187.

[82] Nianyu Li, Javier Cámara, David Garlan, Bradley Schmerl, and Zhi Jin. 2021. Hey! Preparing Humans to do Tasks in Self-adaptive Systems. In *Proc. of 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 48–58. https://doi.org/10.1109/SEAMS51251.2021.00017

[83] Yixiang Lim, Nichakorn Pongsakornsathien, Alessandro Gardi, Roberto Sabatini, Trevor Kistan, Neta Ezer, and Daniel J. Bursch. 2021. Adaptive Human-Robot Interactions for Multiple Unmanned Aerial Vehicles. *Robotics* 10, 1 (2021). https://doi.org/10.3390/robotics10010012

[84] Eric Lloyd, Shihong Huang, and Emmanuelle Tognoli. 2017. Improving human-in-the-loop adaptive systems using brain-computer interaction. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 163–174.

[85] Robert Loh, Y. Bian, and Tim Roe. 2009. UAVs in civil airspace: Safety requirements. *IEEE Aerospace and Electronic Systems Magazine* 24, 1 (2009), 5–17. https://doi.org/10.1109/MAES.2009.4772749

[86] Joseph B Lyons, Kevin T Wynne, Sean Mahoney, and Mark A Roebke. 2019. Trust and human-machine teaming: A qualitative study. In *Artificial intelligence for the internet of everything*. Elsevier, 101–116.

[87] Azad M. Madni and Carla C. Madni. 2018. Architectural Framework for Exploring Adaptive Human-Machine Teaming Options in Simulated Dynamic Environments. *Systems* 6, 4 (2018). https://doi.org/10.3390/systems6040044

[88] Pattie Maes. 1987. Concepts and Experiments in Computational Reflection. In *Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications* (Orlando, Florida, USA) *(OOPSLA '87)*. Association for Computing Machinery, New York, NY, USA, 147–155. https://doi.org/10.1145/38765.38821

[89] Paulo Henrique Maia, Lucas Vieira, Matheus Chagas, Yijun Yu, Andrea Zisman, and Bashar Nuseibeh. 2019. Dragonfly: a tool for simulating self-adaptive drone behaviours. In *Proc. of 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 107–113.

[90] Owen McAree, Jonathan M Aitken, and Sandor M Veres. 2016. A model based design framework for safety verification of a semi-autonomous inspection drone. In *Proc. of the 11th International Conference on Control*. IEEE, 1–6.

[91] Patricia McDermott, Cindy Dominguez, Nicholas Kasdaglis, Matthew Ryan, and Isabel Trahan. 2018. Human-Machine Teaming Systems Engineering Guide, Technical Report # MP180941. (2018).

[92] Patricia L McDermott, Katherine E. Walker, Cynthia O Dominguez, Alex Nelson, and Nicholas Kasdaglis. 2018. Quenching the Thirst for Human-Machine Teaming Guidance: Helping Military Systems Acquisition Leverage Cognitive Engineering Research, The MITRE Corporation. https://www.mitre.org/publications/technical-papers/quenching-the-thirst-for-human-machine-teaming-guidance-helping. [Last accessed 01-01-2022].

[93] Christopher A. Miller, Joseph Mueller, Christopher Geib, David LaVergne, Phillip Walker, and Joshua Hamell. 2019. User Interaction Approaches For Managing Multiple UAS In The National Airspace. In *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE, 1–16. https://doi.org/10.1109/ICNSURV.2019.8735205

[94] Gabriel Moreno, Cody Kinneer, Ashutosh Pandey, and David Garlan. 2019. DARTSim: An exemplar for evaluation and comparison of self-adaptation approaches for smart cyber-physical systems. In *Proc. of 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 181–187.

[95] Henry Muccini and Mahyar Tourchi Moghaddam. 2018. IOT Architectural Styles. In *Proc. of 2018 European Conference on Software Architecture*. Springer, 68–85.

[96] Harish Narayanappa, Mukul S Bansal, and Hridesh Rajan. 2010. Property-aware program sampling. In *Proc. of the 9th ACM SIGPLAN-SIGSOFT WS on Program analysis for software tools and engineering*. 45–52.

[97] Amin Nourmohammadi, Mohammad Jafari, and Thorsten O Zander. 2018. A survey on unmanned aerial vehicle remote control using brain–computer interface. *IEEE Transactions on Human-Machine Systems* 48, 4 (2018), 337–348.

[98] Olessia Ogorodnikova. 2008. Human weaknesses and strengths in collaboration with robots. *Periodica Polytechnica Mechanical Engineering* 52, 1 (2008), 25–33.

[99] Oskar Palinko, Andrew L Kun, Alexander Shyrokov, and Peter Heeman. 2010. Estimating cognitive load using remote eye tracking in a driving simulator. In *Proc. of 2010 Symposium on Eye-Tracking Research & Applications*. ACM, New York, 141–144.

[100] April Rose Panganiban, Gerald Matthews, and Michael D Long. 2020. Transparency in autonomous teammates: Intention to support as teaming information. *Journal of Cognitive Engineering and Decision Making* 14, 2 (2020), 174–190.

[101] Juan Parra-Ullauri, Antonio Garcia-Dominguez, Nelly Bencomo, and Luis Garcia-Paucar. 2022. History-aware explanations: towards enabling human-in-the-loop in self-adaptive systems. In *MODELS '22: 25th International Conference on Model Driven Engineering Languages and Systems*. ACM. http://dro.dur.ac.uk/37030/

[102] Colin Paterson, Radu Calinescu, Suresh Manandhar, and Di Wang. 2019. Using Unstructured Data to Improve the Continuous Planning of Critical Processes Involving Humans. In *Proc. of 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 25–31. https://doi.org/10.1109/SEAMS.2019.00013

[103] Xin Peng, Bihuan Chen, Yijun Yu, and Wenyun Zhao. 2010. Self-tuning of software systems through goal-based feedback loop control. In *2010 18th IEEE International Requirements Engineering Conference*. IEEE, 104–107.

[104] Alexandre Pouget, Jan Drugowitsch, and Adam Kepecs. 2016. Confidence and certainty: distinct probabilistic quantities for different goals. *Nature neuroscience* 19, 3 (2016), 366.

[105] Andres J Ramirez, Betty HC Cheng, and Philip K McKinley. 2010. Adaptive monitoring of software requirements. In *Proc. of the First International Workshop on Requirements@Run. Time*. IEEE, 41–50.

[106] Nicolas Regis, Frédéric Dehais, Emmanuel Rachelson, Charles Thooris, Sergio Pizziol, Mickaël Causse, and Catherine Tessier. 2014. Formal Detection of Attentional Tunneling in Human Operator-Automation Interactions. *IEEE Transactions on Human-Machine Systems* 44, 3 (2014), 326–336. https://doi.org/10.1109/THMS.2014.2307258

[107] R Siqueira Reis, AA Hino, and CR Añez. 2010. Perceived stress scale. *J. health Psychol* 15, 1 (2010), 107–114.

[108] Owen Reynolds, Antonio García-Domínguez, and Nelly Bencomo. 2020. Automated provenance graphs for models@run.time. In *Proc. of the ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems*. ACM, New York, 1–10. https://doi.org/10.1145/3417990.3419503

[109] Owen Reynolds, Antonio García-Domínguez, and Nelly Bencomo. 2020. Towards automated provenance collection for runtime models to record system history. In *Proc. of the 12th System Analysis and Modelling Conf.* 12–21.

[110] Karina A Roundtree, Michael A Goodrich, and Julie A Adams. 2019. Transparency: Transitioning from human–machine systems to human-swarm systems. *Journal of Cognitive Engineering and Decision Making* 13, 3 (2019), 171–195.

[111] Günther Ruhe, Armin Eberlein, and Dietmar Pfahl. 2003. Trade-off analysis for requirements selection. *International Journal of Software Engineering and Knowledge Engineering* 13, 04 (2003), 345–366.

[112] Luca Sabatucci and Massimo Cossentino. 2015. From Means-End Analysis to Proactive Means-End Reasoning. In *Proc. 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 2–12.

[113] Lucas Sakizloglou, Sona Ghahremani, Matthias Barkowsky, and Holger Giese. 2021. Incremental execution of temporal graph queries over runtime models with history and its applications. *Software and Systems Modeling* (2021).

[114] Lucas Sakizloglou, Sona Ghahremani, Thomas Brand, Matthias Barkowsky, and Holger Giese. 2020. Towards Highly Scalable Runtime Models with History. In *Proc. of the IEEE/ACM 15th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems* (Seoul, Republic of Korea). ACM, 188–194.

[115] Md Nafee Al Islam Jane Cleland-Huang Myra Cohen Salil Purandare, Urjoshi Sinha. 2023. Self-Adaptive Mechanisms for Misconfigurations in Small Uncrewed Aerial Systems. In *Proc. of the International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2023, Melbourne, Australia, 2023*.

[116] Lindsay Sanneman and Julie A. Shah. 2020. A Situation Awareness-Based Framework for Design and Evaluation of Explainable AI. In *Explainable, Transparent Autonomous Agents and Multi-Agent Systems*. Springer International Publishing, Cham, 94–110.

[117] Filippo Santoni de Sio and Jeroen van den Hoven. 2018. Meaningful Human Control over Autonomous Systems: A Philosophical Account. *Frontiers in Robotics and AI* 5 (2018), 15. https://doi.org/10.3389/frobt.2018.00015

[118] Daniela Schmid, Bernd Korn, and Neville Stanton. 2020. Evaluating the Reduced Flight Deck Crew Concept Using Cognitive Work Analysis and Social Network Analysis: Comparing Normal and Data-link Outage Scenarios. *Cognition, Technology & Work* 22 (02 2020). https://doi.org/10.1007/s10111-019-00548-5

[119] Karma Sherif and Omolola Jewisimi. 2018. Electronic Performance Monitoring Friend or Foe: Empowering Employees through Data Analytics. In *24th Americas Conference on Information Systems, AMCIS 2018, New Orleans, LA, USA, August 16-18, 2018*. Association for Information Systems. https://aisel.aisnet.org/amcis2018/CultureIS/Presentations/7

[120] J. Shi, J. Wan, H. Yan, and H. Suo. 2011. A survey of Cyber-Physical Systems. In *Proc. of 2011 International Conference on Wireless Communications and Signal Processing*. 1–6. https://doi.org/10.1109/WCSP.2011.6096958

[121] Charles D Spielberger, Fernando Gonzalez-Reigosa, Angel Martinez-Urrutia, Luiz FS Natalicio, and Diana S Natalicio. 1971. The state-trait anxiety inventory. *Revista Interamericana de Psicologia/Interamerican journal of psychology* 5, 3 & 4 (1971).

[122] Tim Claudius Stratmann and Susanne Boll. 2016. Demon Hunt - The Role of Endsley's Demons of Situation Awareness in Maritime Accidents. In *Human-Centered and Error-Resilient Systems Development*. Springer International Publishing, Cham, 203–212.

[123] Alistair G Sutcliffe, Neil AM Maiden, Shailey Minocha, and Darrel Manuel. 1998. Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering* 24, 12 (1998), 1072–1088.

[124] Aurélien Vialon, Kenji Tei, and Samir Aknine. 2017. Soft-Goal Approximation Context Awareness of Goal-Driven Self-Adaptive Systems. In *Proc. of 2017 IEEE International Conference on Autonomic Computing*. IEEE, 233–238. https://doi.org/10.1109/ICAC.2017.25

[125] Michael Vierhauser, Sean Bayley, Jane Wyngaard, Wandi Xiong, Jinghui Cheng, Joshua Huseman, Robyn R. Lutz, and Jane Cleland-Huang. 2021. Interlocking Safety Cases for Unmanned Autonomous Systems in Shared Airspaces. *IEEE Transactions on Software Engineering* 47, 5 (2021), 899–918. https://doi.org/10.1109/TSE.2019.2907595

[126] Michael Vierhauser, Jane Cleland-Huang, Sean Bayley, Thomas Krismayer, Rick Rabiser, and Pau Grünbacher. 2018. Monitoring CPS at runtime-A case study in the UAV domain. In *Proc. of 44th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 73–80.

[127] Michael Vierhauser, Md Nafee Al Islam, Ankit Agrawal, Jane Cleland-Huang, and James Mason. 2021. Hazard analysis for human-on-the-loop interactions in sUAS systems. In *Proc. of 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, New York, 8–19.

[128] Norha M Villegas, Gabriel Tamura, Hausi A Müller, Laurence Duchien, and Rubby Casallas. 2013. DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Springer, 265–293.

[129] Thomas Vogel and Holger Giese. 2012. A language for feedback loops in self-adaptive systems: Executable runtime megamodels. In *Proc. of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 129–138.

[130] Thomas Vogel, Andreas Seibel, and Holger Giese. 2010. The role of models and megamodels at runtime. In *Proc. of the International Conference on Model Driven Engineering Languages and Systems*. Springer, 224–238.

[131] Guy H. Walker, Neville A. Stanton, Paul M. Salmon, and Daniel P. Jenkins. 2008. A review of sociotechnical systems theory: a classic concept for new command and control paradigms. *Theoretical Issues in Ergonomics Science* 9, 6 (2008), 479–499. https://doi.org/10.1080/14639220701635470 arXiv:https://doi.org/10.1080/14639220701635470

[132] Martin Weißbach, Philipp Chrszon, Thomas Springer, and Alexander Schill. 2017. Decentrally Coordinated Execution of Adaptations in Distributed Self-Adaptive Software Systems. In *Proc. of 11th International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 111–120.

[133] Danny Weyns and Jesper Andersson. 2013. On the challenges of self-Adaptation in systems of systems. *Proc. of the 1st ACM SIGSOFT/SIGPLAN International Workshop on Software Engineering for Systems-of-Systems, SESoS 2013 Proceedings*, 47–51. https://doi.org/10.1145/2489850.2489860

[134] Choon Yue Wong and Gerald Seet. 2017. Workload, awareness and automation in multiple-robot supervision. *International Journal of Advanced Robotic Systems* 14, 3 (2017), 1–16. https://doi.org/10.1177/1729881417710463

[135] Robert H Wortham, Andreas Theodorou, and Joanna J Bryson. 2017. Robot transparency: Improving understanding of intelligent behaviour for designers and users. In *Annual Conference Towards Autonomous Robotic Systems*. Springer, 274–289.

[136] Enes Yigitbas, Kadiray Karakaya, Ivan Jovanovikj, and Gregor Engels. 2021. Enhancing human-in-the-loop adaptive systems through digital twins and VR interfaces. In *Proc. of the 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 30–40.

[137] Yijun Yu, Danny Barthaud, Blaine A. Price, Arosha K. Bandara, Andrea Zisman, and Bashar Nuseibeh. 2019. LiveBox: A Self-Adaptive Forensic-Ready Service for Drones. *IEEE Access* 7 (2019), 148401–148412. https://doi.org/10.1109/ACCESS.2019.2942033

[138] Zijian Zhang, Jaspreet Singh, Ujwal Gadiraju, and Avishek Anand. 2019. Dissonance Between Human and Machine Understanding. In *Proc. of ACM on Human-Computer Interaction*, Vol. 3. ACM, New York, 1–23. https://doi.org/10.1145/3359158