

Towards Transparent Marine Autonomy: Visual Mission Composition for Teleoreactive AUVs

Christopher Thierauf
Deep Submergence Laboratory
Woods Hole Oceanographic Institution
Woods Hole, Massachusetts, USA
christopher.thierauf@whoi.edu

Charlotte Stieve
Oceaneering International, Inc.
Work done at the Deep Submergence Laboratory
Woods Hole Oceanographic Institution
CStieve@oceaneering.net

Abstract—Despite the demonstrated utility of symbolic task-and-motion planning systems for safe and adaptable behavior, non-technical users of deployed marine autonomy generally do not use these methods because developers often assume they are complex to build and opaque to interpret. We present a mission-planning interface that addresses these concerns by abstracting a mission-planning problem for the operator into a set of composable goals with optional ordering constraints, which delegate to an onboard teleoreactive task and motion planner. Resulting behaviors can be viewed in 2D or 3D alongside plan decomposition, resource commitments, and anticipated state progression. We argue that this design places goal-level retaskable autonomy in the hands of non-technical users without demanding symbolic planning expertise. The design draws on the levels-of-robot-autonomy (LORA) and situation awareness-based agent transparency (SAT) frameworks, is deployed as a web application, and is released in part as open-source packages. We report preliminary evaluations as a cognitive walkthrough against the workflow currently used for AUV Sentry, in preparation for a more comprehensive on-vehicle user study. We also discuss the implications of this work for calibrated performance-based trust, identifying it, alongside relation-based trust and psychological safety, as a motivating concern.

Index Terms—human-robot interaction, autonomous underwater vehicles, mission planning, trust in automation, agent transparency, cognitive walkthrough

I. INTRODUCTION

Mission-planning user interfaces form the dominant control interaction between marine robots and their operators, yet the underwater human-robot interaction space has invested little into user interfaces rooted in cognitive science and HRI principles. Existing tooling either requires text configuration accessible only to expert users, or provides visual composition without cognitive grounding (Section II). One common method, such as that used by AUV Sentry, is text-file-based configuration that specifies a sequence of behaviors to be executed linearly (with some cases permitting triggers for non-linear behavior). Reasoning and planning systems accept goals and thus dramatically raise autonomy, but they require domain authoring and planning by system experts, which limits adoption. As a result, both require specialized training to produce and extend AUV behavior, especially for more complex planning systems. Many direct clients of these systems (in our case, the science users who define the mission

objectives) cannot specify a deployment directly and must instead work through an intermediary expert operator.

The underserved combination for marine deployments is high autonomy with high usability, partly because autonomy requires users to surrender specific behavioral expectations and let the autonomous system take the lead, which requires operator trust in the autonomous system even when autonomy can produce the better outcome. Symbolic task-planning architectures (such as the teleoreactive architecture built upon here [1]) raise autonomy by empowering the robot to reason toward complex goals; related work reports that letting a robot proactively reason and respond to corrected instructions eases operator effort in collaborative settings [2], [3]. However, these methods often require complex domains and may produce difficult-to-debug outputs. Without predictive interpretability of automated decisions, operators cannot calibrate expectation to robot capability before committing to a plan [4], while not having appropriate trust in the system leads to misuse or disuse [5]: users either commit to missions the AUV cannot complete, or under-utilize it because future behavior remains opaque.

The operator must rely on the autonomous system and remain responsible for its performance, yet must also communicate mission constraints to superiors or other clients who provide the high-level tasks the operator constructs deployments to satisfy. They are therefore both an operator and an interpreter, a relationship with implications for calibrated trust [4], [6] and psychological safety [7], which remains a motivation but underexplored here and in marine HRI in general.

We present a user interface addressing these issues via several key realizations: first, that domain construction (frequently referenced as a problem for the operator) can be considered an engineering problem abstracted away from the user; second, that the relevant task and motion planning problem can be constrained to a subset of the domain and represented as common ‘blocks’; third, that the desired output can be represented as an ordered block sequence with accompanying trajectory and resource cost graphs.

To this end, this work presents a ‘planner as a partner’ framework through a composition interface which separates what the operator decides from what the system reasons, generates, and communicates visually. Thus, the contributions

of this work are:

- A visual mission composition model which maps onto the levels of robot autonomy [8] (itself building on the levels-of-automation taxonomy [9]);
- a plan inspector grounded in the SAT model [10] exposing the planner’s decomposition, resource commitments, and predicted state evolution to provide agent transparency;
- a synchronized mission-timeline visualization supporting predictive visibility of resource and state evolution, with releases of related open-source packages; and
- a comparative cognitive walkthrough across five mission scenarios (three as capability contrasts), against the current expert-mediated waypoint workflow as baseline, identifying which composition steps a science user can perform unaided, and scoping the user study that these findings motivate.

II. BACKGROUND

Marine Mission Planning Interfaces. The widely used MOOS-IvP [11] requires text configuration files, as does Sentry via its Mission Controller (MC) [1], [12], which limits use to expert operators and conflicts with the highly visual nature of marine survey planning. Visual tools exist but lack concrete cognitive analysis demonstrating that they increase operator performance and assurance. Consider the Neptus graph-based planner [13], waypoint-based QGroundControl (see [14] for automated planning extensions), and various commercial vendor systems.

In each of these systems, autonomy is limited: human intelligence turns intent into vehicle behavior. Higher-autonomy planners that accept goals exist (surveyed for AUVs and UAVs in [15]), but they expose complex domains that are generally interpretable only to technical experts: T-REX [16] provided teleoperative execution for AUVs a decade before this work, but required mission specification in the New Domain Description Language (NDDL), which in practice kept authoring with specialist operators. Marine autonomy has received some attention from human-factors engineering, but generally targets teleoperation and supervisory control of UUVs [17], with a more recent push into human-diver interactions [18], [19]. This differs dramatically from the environment we target, where the AUV’s goal is to extend human reach beyond already-observable spaces, so the operator depends on the vehicle’s autonomy to accomplish its objectives and return safely. In these cases, no information may be available in real time (or, if an acoustic link is available, information is highly limited).

This work builds on a cognitive teleoperative mission planning and execution framework (not yet published in detail; an early version is discussed in [1]) in which missions are expressed as goals over symbolic fluents, grounded in optimizable numeric information, and decomposed into behaviors through reasoning. For example, when the user provides a series of survey zones, the system provides the required descent steps, optimizes their order, and then assembles the survey zones from the individual tracklines

needed to produce the overall behavior. When executed, these behaviors run teleoperatively and are constantly re-assessed to determine whether problem-solving behaviors should be employed. From the set of satisficing plans, an execution cost model considers runtime and battery to score and select the optimal plan based on the trajectories each would require. This combination of symbolic fluents, grounding, a cost model, and teleoperative execution enables goal delegation. It further enables plan inspection, AUV self-assessments and outcome predictions, and real-time re-tasking. In contrast, a fixed linear command sequence (such as produced by MC) has none of these. Here, the interface is a client of the planning API, which we consider independently of the vehicle deployment state for both our purposes and standard mission-planning contexts.

For comparison here, we focus on a new frontend for the onboard task and motion planner [1] and MC. MC is the existing mission planning system that has been central to AUV Sentry deployments for over a decade [12], and it provides expert operators with a function call that, when scripted, produces a linear sequence of commands the mission executive can perform in order. The planner being developed to replace it is a teleoperative cognitive architecture: it comes configured with a set of domain knowledge and, when an operator provides a set of goals, it produces a sequence of reactive behaviors (including motion plans) to achieve them. During execution, it self-checks to ensure it is making appropriate progress toward its goals and modifies its behaviors if necessary. These goals can also be changed in real time.

This is a dramatic increase in autonomy, but it also increases operator complexity if the interface is not managed effectively. We use MC as our comparison point for a standard mission planning and execution stack, and the onboard task and motion planner for a symbolic planning case.

Autonomous Deployments for Non-Technical Users, and Grounding in HRI. Block-based programming has been evaluated for adult novices, outperforming other methods for industrial programming approaches on speed, usability, and learnability [20], [21], and applied to drone control for novice programmers in an educational context [22]. A comparison of behavior-tree and state-machine mission-specification DSLs found that both offer reasonable abstraction over text-based languages but still require users to learn the underlying formalism [23] (the gap we target with this abstraction). Yet, although marine deployments are frequently long-horizon, unobservable during execution, and not iteratively debuggable, they have not consistently adopted these findings.

In contrast, this work grounds itself in core principles of HRI and HCI: the levels-of-automation taxonomy [9], despite originating in undersea teleoperation, has been most thoroughly developed in the HRI space rather than marine technology, and the resulting principles remain underused in marine mission planning. The LORA taxonomy [8] allocates sensing, planning, and acting between human and robot, while SAT [10], [24] outlines what facts an agent must provide to support operator situational awareness [25], namely: what it is doing (SAT-1), why (SAT-2), and what happens next

(SAT-3). This complements the types-and-levels view of automation [26]. Reflecting this observation, we map each inspector view to a distinct SAT level.

Trust and Safety. In addition to seeking an effective user interface for autonomous systems, this work aims to lay the foundation for appropriately calibrated trust in autonomy and for understanding the psychological and social factors around it. We follow [6], whose survey of trust definitions in HRI divides them into performance-based trust (“can I trust this agent to complete the task I specify?”) and relation-based trust (“can I trust this agent to behave appropriately as a social participant?”). Both forms are a willingness to accept vulnerability to the robot’s behavior, and both are distinct from reliance, the behavior that willingness informs [4]. Our evaluation here addresses the performance-based form and the reliance it produces: the operator of an AUV must rely on it to make sensible decisions in real time, without complete information about the deployment during operation.

Trust must be appropriately calibrated. In the case of under-trust, operators may disuse the system; in cases of over-trust, they may misuse it [5]. Implementations of deployed marine autonomy systems have generally erred on the side of minimal robot agency because of deployment risk, and much of that intuition is attributable to appropriate consideration of deployment cost and vehicle risk. But some of it is disuse: deployments held to minimal autonomy even when a more adaptable plan is viable and more effective. When the reason to avoid autonomy is opacity rather than unreliability, the gap is transparency: transparency in a plannable understanding of how and why something may fail improves outcomes for the agent and is necessary for the operator to become appropriately calibrated [27], [28].

The second cost of opaque autonomy is interpersonal, and it changes the recipient of trust. When an operator is instructed to produce a mission plan for a system that cannot communicate its own capability or intent, the operator takes on that responsibility and becomes an enabler, mediator, or obstacle between the client and the deployed asset; verification additionally runs through this system expert under time pressure. The client’s trust is then placed in the operator rather than the vehicle. If the operator is wrong, team members may lose trust in the operator or in the AUV directly. Whether the operator feels well equipped to voice potential shortcomings of the system, and better ways to manage objectives, is an indicator of psychological safety [7], which Edmondson identifies as a prerequisite for such voice rather than a consequence of it. Although [7] was developed for human work teams, its extension to a team that includes an autonomous agent with unobservable operations is of interest.

III. CONSTRUCTED SYSTEM

Architecture. The user interface is a web application so that no download is required, and provides the same HTTP API as on-vehicle for testing pre-deployment with a backend simulator. The same interface is available online and can be deployed locally and privately to eliminate connectivity

requirements. The Reflex framework produces dynamic web applications from Python; the backend is a FastAPI service that interprets the same commands provided to the vehicle and executes them on the simulator. Permitted users authenticate with bearer tokens and can save generated plans for sharing.¹

The frontend provides the visual tools to create shapes, goals, and constraints. These are the same tools used to represent plans from the mission executive’s Python API, so the frontend’s role is simply to construct the appropriate order of Python calls. Saving and export, then, become sharing this Python file, which lets deployments always use the most up-to-date version of vehicle behaviors while remaining fully introspectable, archivable, and editable. The same file drives all of the introspection views described below.

Geographic Workspace. The geographic workspace (Figure 1) provides a visual intuition of the current plan before and during construction. Users drag-and-drop coordinates or polygons directly onto the map, set their parameters (e.g., altitude or depth, speed, swath width) via inline fields, use them in behaviors (such as “survey” or “idle”), and view the generated result afterward (e.g., specific tracklines, as shown). Standard resets are provided as a one-click convenience. Exclusion zones are supported by marking a zone as not permitted, which turns the zone red, prevents plans from generating motions that enter that space, and triggers safety fallbacks (a geofence) if the vehicle approaches or enters it during execution. High-resolution bathymetry can be automatically imported into the 3D viewer, and tracklines will be projected onto the 3D space as appropriate.

Plan Construction. The plan constructor (Figure 2) abstracts the symbolic task planning problem and builds on previous findings highlighting the effectiveness of block-based programming modalities. Plan blocks pass symbolic goals to a task reasoning/planning system; survey blocks map to coverage goals; goto blocks map to waypoint goals; action blocks execute behaviors directly. The system also supports blocks to enforce or relax ordering. Forcing order means planning toward each goal independently, with one block’s start state being the prior block’s end state; relaxing order lets the task and motion planner optimize. The order block is therefore a constraint on the single planner: it lets the user require a sequence without hand-specifying the intervening behaviors, so the planner remains the sole source of plans. The abstract plan is grounded in numeric values (e.g., ‘waypoint 1’ becomes coordinates), a cost model optimizes the planning space, and the result is decomposed into plan steps, motion plans with coordinates, and cost over time. Infeasible plans are grounded in their reason (unsatisfiable constraints, no valid decomposition for a goal, or insufficient time or battery re-

¹Two open source packages are released as a companion to this work. To create these graphs, we adapted the JavaScript charting library ApexCharts for use with the Reflex platform, which lets web applications be authored in Python; this generally applicable integration is publicly available at <https://pypi.org/project/reflex-apexcharts/>. The mission timeline utility is released as a second package, enabling integration into other mission planning interfaces beyond this work: <https://pypi.org/project/reflex-mission-graphs/>

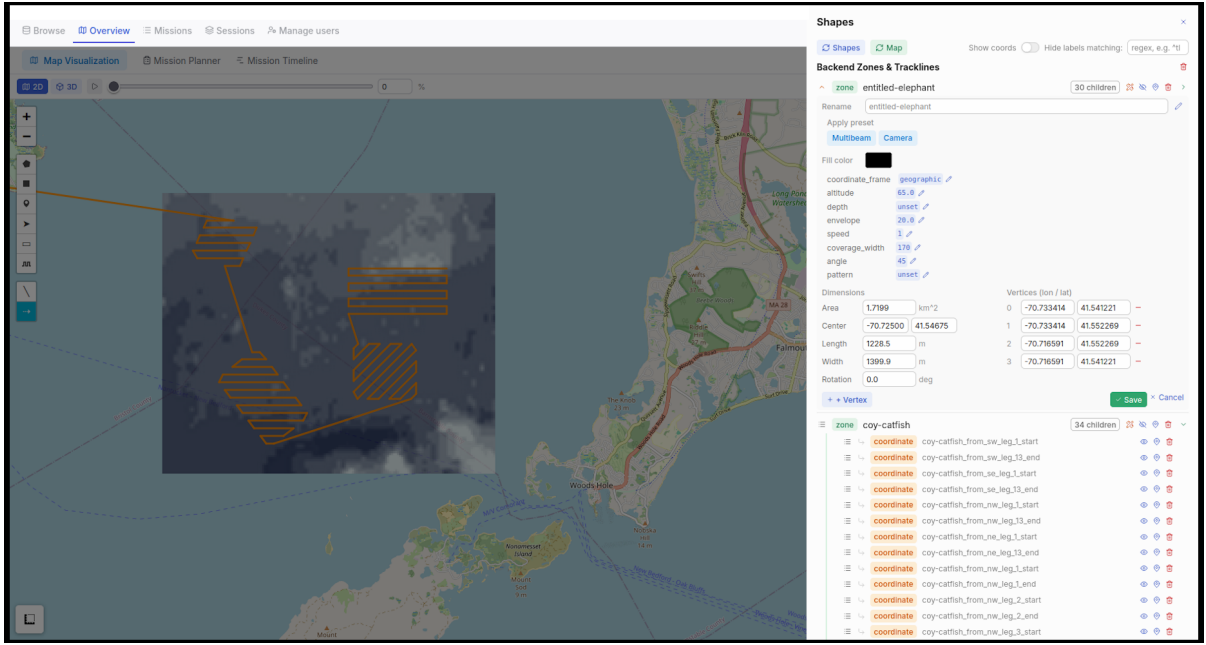


Fig. 1. Geographic workspace with shape editing window open. In the foreground: One zone edit is visible with preset buttons; one set of coordinates produced by a zone is also visible. In the background: Generated tracklines in orange, on top of imported bathymetry. Drawn zones and labels toggled off for visibility.

sources) and reported to the operator as a text-based error. This is the same planner available to the agent during operation.

Table I shows the translation between user-selectable blocks and their symbolic/programmatic counterparts. Each resolves to either an arbitrary conjunction or a specific type of fluents. The interface prevents ill-formed goals (e.g., a survey goal requires a zone to exist), but cannot block infeasible sequences until the planner runs. Here, we describe allocating decision authority using LORA. Because sensing is robot-allocated throughout, our blocks vary only the planning and acting allocation across LORA's levels: an action block is manual, direct behavior specification (near manual/tele-operation); goto and survey blocks are supervisory control (the operator states a goal and permits the planner to decompose, optimize, and act); a plan block does the same at the high-autonomy end, for all representable states; and an order block adds an ordering constraint, letting the operator enforce sequence without over-constraining behavior through action blocks. This authority allocation can be provided to either the expert operator or the non-expert client: mission construction authority can therefore be given to the client to specify mission objectives, while safety authority remains with the expert operator.

IV. PLAN INSPECTOR AND MISSION TIMELINE

The plan inspector (Figure 3) exists to provide observation of behavior before committing to it, to compensate for minimal observability post-deployment and to properly inform decision making while planning. It combines a hierarchical decomposition plan with the SAT model [10], [24]: the hierarchical step viewer addresses SAT-1 (*what is the robot doing?*), the step-breakdown addresses SAT-2 (*why?*), and timeline and battery charts address SAT-3 (*what next?*). The

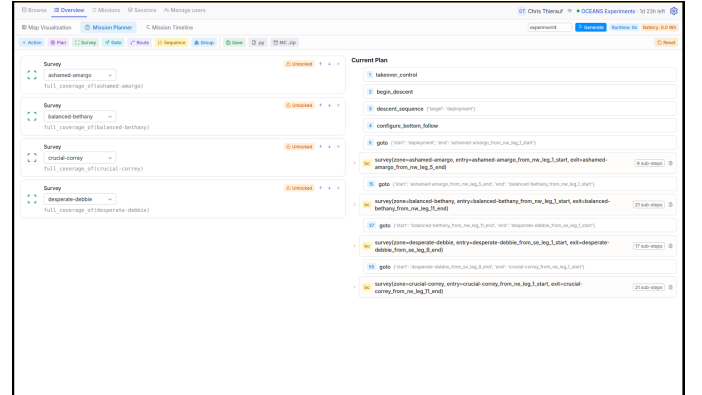


Fig. 2. Block-based plan constructor showing goals (left) and autonomously-produced behavior (right).

TABLE I
BLOCK TYPES, CORRESPONDING SYMBOLIC GOAL, OPERATOR-SET PARAMETERS, AND ALLOCATION OF PLANNING AND ACTING AUTHORITY UNDER LORA [8].

Block	Symbolic goal	Operator-set params.	LORA
action	(none; runs a behavior)	behavior's params	tele-op.
plan	conjunction of fluents	fluent rows (typed)	supervisory
survey	full_coverage_of(zone)	zone	supervisory
goto	visited(coord)	coordinate	supervisory
order	(none; ordered subseq.)	child blocks	(constraint)

three views are vertically aligned and share a common time axis. The task timeline shows the generated plan decomposed by task and annotated by time and resource consumption. Plans are hierarchical: a survey decomposes into a series of

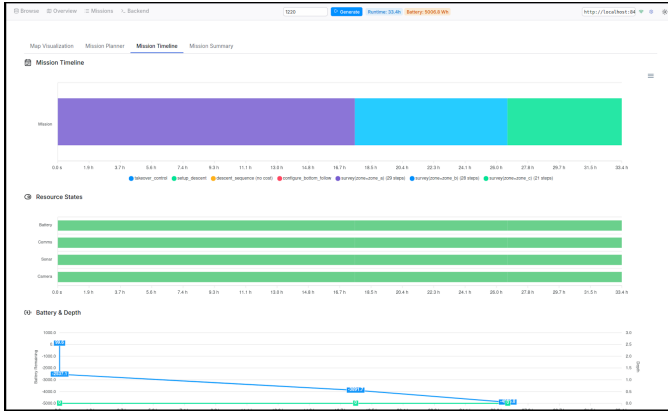


Fig. 3. Plan inspector: synchronized mission task timeline, resource state timeline, and continuous resource/state plot.

tracklines, which decompose into a series of *goto*'s; the decomposition granularity is set at the top level and can be expanded to show the individual sub-tasks that enable it. The resource timeline shows per-resource availability and commands populated again from the task and motion planner. Finally, the UI plots continuous battery and depth over mission time. With this, the operator can concisely answer “will this mission exhaust resources before completion” or “will this sub-task reduce probability of success more than other factors” using the built-in performance analysis tooling, which draws on per-behavior records of runtime, battery use, and success rate accumulated across prior missions [1]. We provide the planner’s decomposition and predicted trajectory as an explainable AI strategy during composition time.

V. COGNITIVE WALKTHROUGH

Method and Scope. Assessment is through a comparative cognitive walkthrough [29], [30]. We walk through two scenarios step-by-step in both our visual composition interface and the current expert-mediated textual workflow; we examine three further scenarios as capability contrasts rather than walk them in full: the AUV’s ability to plan and re-plan independently provides a feature set that MC cannot. We examine each walked step through the four cognitive-walkthrough questions [29], which probe the user’s cognition: Q1) *will the user try to achieve the right effect?* Q2) *Will the user notice that the correct action is available?* Q3) *will the user associate the correct action with the effect they want?* and Q4) *once the correct action is performed, will the user see that progress is being made?*

The walkthrough predicts whether an untrained user can determine what to do at each step. It cannot measure workload, task time, error rate, trust calibration, or psychological safety, and we make no such claim. Instead, it identifies specific interaction barriers and points to directions for future study.

The technique is designed for expert analysis of a modeled user’s decision-making, so it suits preliminary exploration but does not replace a more comprehensive study (discussed in future work). The authors conducted the walkthrough and

scored against our own interface rather than in its favor when encountering uncertainty. We distinguish learned barriers (cases where the operator may make a mistake that experience can prevent) from persistent ones (cases where the interface never supplies the needed information or operation). Beyond the four standard questions, we also track, per step, whether completion requires the science user to request, question, or correct the expert operator, which we report in Section V-B. We used no human participants for this preliminary analysis; we leave that for future work.

Assumed User Profile. The assumed user of the AUV is the typical AUV Sentry case: a sea-going investigator who has requested AUV deployments across the broad spectrum of use-cases Sentry supports, from custom sensor deployments to lawnmower surveys to adaptive sampling. Although this user knows what data they want and where they want it, and works comfortably with coordinates and charts, they lack the technical expertise to run a mission or understand the physical constraints that may make certain paths infeasible. They do not know the custom scripting interface for Sentry or how it orders zones and decomposes coverage, so they need an expert operator as an intermediary. They state their goals in their own terms: for example, “*perform a multibeam survey in this region for 5 hours, then transit to these coordinates for a photo survey close enough to see individual chimneys*”. Their position in the shipboard hierarchy relative to the expert operator and other investigators with competing priorities varies wildly, as does the presence of a working relationship. However, the investigator always specifies objectives, while the operator always retains a safety veto. Although we focus on Sentry, this relationship parallels many other deployment scenarios in which an operator deploys an autonomous system on behalf of a client, employer, or organization.

Baseline: Current Expert-Mediated Workflow. Sentry’s current workflow proceeds by waypoints provided to MC, generated by custom scripts that a system expert hand-modifies per mission based on the science user’s description and feedback. At sea, this cycle delays missions and introduces interpretation and miscommunication risks. Per the program’s internal procedural checklist, the workflow has roughly 27 atomic steps, from map preparation to scripting and verification. Every step is the expert operator’s; the client specifies objectives, confirms the operator’s output after generation, and may modify those objectives throughout the planning iteration cycle until the hard launch deadline. Coverage adequacy, altitude and depth, endurance, and which sensors are appropriate are determined by the operator and not independently verifiable. MC ingests a linear command sequence: it provides arbitrary behaviors, from surveys to idle patterns and more, but only as an expert-authored sequence with no reordering or reactivity.

A. Scenarios

The visual editor allows the science user (the client) to compose a mission and visually verify it meets their needs, after which the expert operator confirms safety for deployment.

Scenario 1 (the simplest case) shows barriers in each. In the visual interface, configuring parameters fails Q3 because altitude, swath, and pattern are vehicle terms, but we simplify with presets. In MC, specifying the survey fails Q1 when adding the survey block because this block is specified as a path rather than an intent, fails Q2 because text-based interfaces require that the user know what commands are available, and retains the Q3 barrier. Both are effective at executing the tasks once they are provided and therefore pass Q4.

Scenario 2 (multi-zone survey) inherits the above but adds ordering as another variable. In the visual planner, the optimization ignores user-specified order unless the user explicitly enforces it, and so we judge against ourselves for Q1.

Scenario 3 (exclusion zones) is a capability difference: the block interface represents a keep-out as a no-entry zone the planner routes around and the geofence enforces, verifiable before commitment, whereas MC has no keep-out mechanism and avoidance can only be approximated by not drawing tracklines through the region.

Scenario 4 (mid-mission re-tasking) is also a capability difference: a new plan can be constructed from the current operating state and communicated acoustically, as demonstrated at sea in prior work [1], while MC cannot be re-tasked in the water, so a change means aborting, recovering, and re-running the whole checklist.

Scenario 5 (non-survey goal delegation) covers a goal expressible in both architectures but beyond the standard survey case. The plan block delegates decomposition and ordering to the planner (the highest autonomy in our mapping), whereas MC requires the expert to hand-author a fixed linear sequence and manually decompose the goal.

B. Summary and Realizations

We compare the workflows structurally rather than by step count, which would compare unlike units; see Table II for detail. The key observation from this walkthrough is that success never requires the user to engage a fluent, precondition, or other symbolic planning parameter. In contrast with text-based methods, which require memorizing and authoring them directly, the planning model allows them to be derived or easily inserted with slots or presets. Further, this level of access remains accessible to the science user or other non-technical client, and with the built-in safety constraints producing known-safe behavior, the operator can shift into a safety approver role. Rather than the deployment depending on the operator’s ability to interpret needs outside their expertise, and the client to interpret information outside of theirs, the judgments are brought to the appropriate domain expert. We are unable to confirm that trust is better calibrated in this approach, but we do see a route towards it where the baseline does not.

VI. DISCUSSION AND FUTURE WORK

Why operator and client-facing autonomy is practical.

The central features we present here exist only because a reasoning and planning system runs onboard, providing a feature set that behavior trees, finite state machines, and linear

TABLE II
STRUCTURAL CONTRAST OF THE TWO WORKFLOWS ACROSS THE WALKED AND CAPABILITY-CONTRAST SCENARIOS (QUALITATIVE; SEE TEXT).

Dimension	Visual interface	Baseline (textual, e.g., MC)
Who constructs	client or operator	operator only
Client’s role	composes, verifies	states goals, reviews
Goal stated as	intent	scripted path
Zone ordering	planner-optimized	fixed, authored
Keep-out zones	geofence	none
In-water re-task	yes (inspector)	no (abort+rerun)
Symbolic authoring	none (survey/goto)	manual script
Client verifies by	direct inspection	asking the operator
Transparency	SAT-1/2/3	static text only

sequence executors like MC cannot provide. Delegating intent requires a system capable of interpreting goal-level mission specifications; feasibility feedback requires a model (which we provide with the task and motion planner); re-tasking requires a system capable of autonomous planning. By shifting the core domain construction problem to an engineering problem, which can be handled by a planning-domain expert rather than a problem-domain expert, tasking becomes a tractable subset of planning operators that the problem-domain expert can compose over. These realizations remain general because they are rooted in core features of planning and reasoning systems, not this particular implementation or vehicle.

What is often assumed to be a problem of interpretability is, in reality, a problem of abstraction. Previous objections to using such systems have cited domain authoring as a barrier; by relocating that work to the planning-domain expert (the system builder), who can author the planning operators once, the end-user gets a succinct set of tooling for mission composition. This tooling is accessible to expert operators, non-expert users, and clients.

Handling changing outcomes. Outcomes during execution may change intentionally (operators send a mission update) or unintentionally (sudden failure mode forces plan adaptation). The built-in performance assessment and re-tasking tooling allows this to be modeled, including reasoning over fault hypotheses to select a recovery [31], but its behavior is not well exposed to the casual operator. How self-assessments and their impact on future missions can be represented, and the impact that this has on team dynamics, requires additional research.

Towards user studies and deployments. Merely relying on sensible HRI principles is not enough to confirm reduction of cognitive load or appropriate calibration of trust dynamics, which can be empirically measured (and lacks comprehensive exploration in this area). Team-level dynamics and willingness to express risk assessments are now open for exploration: is an operator more willing to discuss risk factors accurately, or a client to revise objectives, when directly in the loop? Such a study would measure (among other factors) NASA-TLX [32], time to mission completion, and error rate. At-sea data (e.g., replan events and intervention frequency) would provide concrete data supporting the predictions of Section V.

REFERENCES

- [1] C. Thierauf, "Deployment and development of a cognitive teleoreactive framework for deep sea autonomy," in *OCEANS 2025 – Great Lakes*, 2025, pp. 1–6, arXiv: 2510.10716.
- [2] C. Thierauf, T. Law, T. M. Frasca, and M. Scheutz, "Toward competent robot apprentices: Enabling proactive troubleshooting in collaborative robots," *Machines*, vol. 12, no. 1, p. 73, 2024.
- [3] C. Thierauf, R. Thielstrom, B. Oosterveld, W. Becker, and M. Scheutz, "'do this instead' – robots that adequately respond to corrected instructions," *ACM Transactions on Human-Robot Interaction*, 2023.
- [4] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human Factors*, vol. 46, no. 1, pp. 50–80, 2004.
- [5] R. Parasuraman and V. Riley, "Humans and automation: Use, misuse, disuse, abuse," *Human Factors*, vol. 39, no. 2, pp. 230–253, 1997, <https://web.mit.edu/16.459/www/parasuraman.pdf>.
- [6] T. Law and M. Scheutz, "Trust: Recent concepts and evaluations in human-robot interaction," in *Trust in Human-Robot Interaction*, C. S. Nam and J. B. Lyons, Eds. Academic Press, 2021, ch. 2, pp. 27–57.
- [7] A. Edmondson, "Psychological safety and learning behavior in work teams," *Administrative Science Quarterly*, vol. 44, no. 2, pp. 350–383, 1999.
- [8] J. M. Beer, A. D. Fisk, and W. A. Rogers, "Toward a framework for levels of robot autonomy in human-robot interaction," *Journal of Human-Robot Interaction*, vol. 3, no. 2, pp. 74–99, 2014, https://scholarcommons.sc.edu/cgi/viewcontent.cgi?article=1127&context=csce_facpub.
- [9] T. B. Sheridan and W. L. Verplank, "Human and computer control of undersea teleoperators," MIT Man-Machine Systems Laboratory, Tech. Rep., 1978.
- [10] J. Y. C. Chen, K. Procci, M. Boyce, J. Wright, A. Garcia, and M. J. Barnes, "Situation awareness-based agent transparency," U.S. Army Research Laboratory, Tech. Rep. ARL-TR-6905, 2014, <https://apps.dtic.mil/sti/pdfs/ADA600351.pdf>.
- [11] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010.
- [12] C. L. Kaiser, D. R. Yoerger, J. C. Kinsey, S. Kelley, A. Billings, J. Fujii, S. Suman, M. Jakuba, Z. Berkowitz, C. R. German *et al.*, "The design and 200 day per year operation of the autonomous underwater vehicle sentry," in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. IEEE, 2016, pp. 251–260.
- [13] P. S. Dias, S. L. Fraga, R. Gomes, G. M. Gonçalves, F. L. Pereira, J. Pinto, and J. B. d. Sousa, "Neptus – a framework to support multiple vehicle operation," in *OCEANS 2005 – Europe*, 2005.
- [14] C. Ramirez-Atencia and D. Camacho, "Extending QGroundControl for automated mission planning of UAVs," *Sensors*, vol. 18, no. 7, p. 2339, 2018, <https://www.mdpi.com/1424-8220/18/7/2339>.
- [15] A. Atyabi, S. MahmoudZadeh, and S. Nefti-Meziani, "Current advancements on autonomous mission planning and management systems: An AUV and UAV perspective," *Annual Reviews in Control*, vol. 46, pp. 196–215, 2018, <https://arxiv.org/abs/2007.05179>.
- [16] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen, "T-rex: A model-based architecture for auv control," in *3rd Workshop on Planning and Plan Execution for Real-World Systems*, vol. 2007, 2007.
- [17] G. Ho, N. J. Pavlovic, and R. Arrabito, "Human factors issues with operating unmanned underwater vehicles," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 55, no. 1, pp. 429–433, 2011, <https://apps.dtic.mil/sti/pdfs/ADA555588.pdf>.
- [18] M. J. Islam, M. Ho, and J. Sattar, "Understanding human motion and gestures for underwater human-robot collaboration," *Journal of Field Robotics*, vol. 36, no. 5, pp. 851–873, 2019.
- [19] N. Mišković, M. Bibuli, A. Birk, M. Caccia, M. Egi, K. Grammer, A. Marroni, J. Neasham, A. Pascoal, and A. Vasiljević, "Caddy—cognitive autonomous diving buddy: Two years of underwater human-robot interaction," *Marine Technology Society Journal*, vol. 50, no. 4, pp. 54–66, 2016.
- [20] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, and D. Franklin, "Evaluating CoBlox: A comparative study of robotics programming environments for adult novices," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, 2018, pp. 1–12.
- [21] D. Weintrop, D. C. Shepherd, P. Francis, and D. Franklin, "Blockly goes to work: Block-based programming for industrial robots," in *IEEE Blocks and Beyond Workshop (B&B)*, 2017, pp. 29–36.
- [22] E. Tilley and J. Gray, "Dronely: A visual block programming language for the control of drones," in *Proceedings of the 2017 ACM Southeast Conference (ACMSE '17)*. ACM, 2017, pp. 208–211.
- [23] S. Dragule, E. Bainomugisha, P. Pelliccione, and T. Berger, "Effects of specifying robotic missions in behavior trees and state machines," *Journal of Computer Languages*, vol. 85, p. 101330, 2025.
- [24] J. Y. C. Chen and M. J. Barnes, "Situation awareness-based agent transparency and human-autonomy teaming effectiveness," *Theoretical Issues in Ergonomics Science*, vol. 19, no. 3, pp. 259–282, 2018.
- [25] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 32–64, 1995.
- [26] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on Systems, Man, and Cybernetics – Part A*, vol. 30, no. 3, pp. 286–297, 2000.
- [27] C. Thierauf and M. Scheutz, "What matters for designing reliable systems: A new taxonomy of failure cause-condition-effect relationships based on auv missions," in *2026 IEEE Engineering Reliable Autonomous Systems (ERAS)*. IEEE, 2026, pp. 144–150.
- [28] C. Thierauf, "Interpret, innovate, and interact: Resilient robot strategies in human domains," PhD thesis, Tufts University, 2025.
- [29] C. Wharton, J. Rieman, C. Lewis, and P. Polson, "The cognitive walkthrough method: A practitioner's guide," in *Usability Inspection Methods*, J. Nielsen and R. L. Mack, Eds. John Wiley & Sons, 1994, pp. 105–140, <https://www.colorado.edu/ics/sites/default/files/attached-files/93-07.pdf>.
- [30] P. G. Polson, C. Lewis, J. Rieman, and C. Wharton, "Cognitive walkthroughs: A method for theory-based evaluation of user interfaces," *International Journal of Man-Machine Studies*, vol. 36, no. 5, pp. 741–773, 1992, http://sonify.psych.gatech.edu/~ben/references/polson_cognitive_walkthroughs_a_method_for_theory-based_evaluation_of_user_interfaces.pdf.
- [31] C. Thierauf and M. Scheutz, "Self-debugging robots: Fault recovery through reasoning and planning," in *2024 Resilience Week (RWS)*. IEEE, 2024, pp. 1–10.
- [32] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (task load index): Results of empirical and theoretical research," in *Human Mental Workload*, ser. Advances in Psychology, P. A. Hancock and N. Meshkati, Eds. North-Holland, 1988, vol. 52, pp. 139–183.