MISSION ASSURANCE FOR AUTONOMOUS UNDERWATER VEHICLES

by
Karl A. Siil

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Engineering

Baltimore, Maryland
December 2021

© 2021 Karl A. Siil
All rights reserved

Abstract

The ubiquity of autonomous vehicles (AVs) is all but inevitable, and AVs have made fantastic leaps in their capabilities, partly thanks to advances in artificial intelligence and machine learning (AI/ML). With these great capabilities should come great assurance that AVs will behave safely and achieve their operational goals, or mission, despite foreseen and unforeseen circumstances. AV software is highly complex, increasing the likelihood of faults. AI/ML decision making is poorly understood. And, all computer-based systems are vulnerable to malicious software and other cybersecurity threats. Eliminating or mitigating any one of these is an open research problem. AVs must handle all three, without the benefit of a human operator. This dissertation investigates several aspects of AV mission assurance, and offers solutions for test and evaluation starting early in the development cycle, a use case with which to experiment, and a methodology for iteratively improving assurance as more is learned about a mission and its specific risks.

This dissertation focuses on autonomous underwater vehicles (AUVs). Each chapter explores particular aspects of AUV mission assurance and presents approaches to address them. We discuss the risks specific to AUV safety and mission assurance. We introduce the Digital Environment for Simulated Cyber Resilience Engineering, Test and Experimentation (DESCRETE) testbed that enables cost-effective AUV simulation, particularly with respect to system-level faults and attacks. We present the mission-assured AUV (MAAUV) use case, which we used to gather data on DESCRETE to improve the testbed and better understand mission assurance. We propose an iterative mission-assurance refinement analysis (IMARA) methodology for understanding system-failure impacts to mission. Applying IMARA to the MAAUV, we provide a guide for AUV and mission designers to best use limited assurance improvement and mitigation resources. Combining all these provides a comprehensive set of tools to improve AUV assurance.

Primary Reader and Advisor: Aviel Rubin

Secondary Reader(s): Matthew Elder, Anton Dahbura, Matthew Green, and Lanier Watkins

Acknowledgments

I would like to thank those people whose mentorship and support guided me down this long and winding road. I am extremely grateful for having Professor Aviel Rubin as my advisor, for his guidance as I pursued the Johns Hopkins University (JHU) Doctor of Engineering (D.Eng.) degree, and for his mentorship as I investigated autonomous vehicles, the computers that control them, and the risks those computers introduce. The decades of knowledge and experience that Avi brought were an invaluable resource for exploring concerns that not that long ago only applied to information systems, and now apply to cyber-physical systems that can do much more harm. Avi asked the hard questions that kept me on a path towards success, while allowing me the flexibility to forge that path based on my research and professional instincts.

I am also very grateful to Dr. Matthew Elder and Professor Anton Dahbura for serving on my Supervisory Committee and coming on this journey. They were the day-to-day tactical guidance that complemented Avi's strategic direction. Matt and Tony provided access to their extensive research experience and knowledge of multiple domains related to the system- and mission-level problems I was exploring, in particular, the emergent system-of-systems properties that lead to unexpected behaviors and outcomes.

I would also like to express my gratitude to Dr. Lanier Watkins and Professor Matthew Greene for serving on my Proposal Presentation and Examination board, and continuing to provide support and guidance throughout the remainder of the program. In particular, I want to thank Lanier for his guidance on conducting professional scientific research and writing high-quality research papers.

This dissertation, or even my pursuit of a doctoral degree, would not have happened without the JHU and Johns Hopkins University Applied Physics Laboratory (JHU/APL) partnership that established the D.Eng. program. I am honored to have been chosen for one of the first cohorts, and thankful to those who had the inspiration and provided the resources to create this higher-education opportunity. I would also

like to thank Mia Brooms and the rest of the faculty, staff and administrators in the JHU Whiting School of Engineering for their support and tireless effort in getting the fledgling program off the ground.

I am indebted to Brendan McNelly and his JHU/APL team for inspiring the MAAUV scenario; and, special thanks go to Stephen Giguere, Shaku Harshavardhana, David Sames, Katherine Watko, and JR Parsons for supporting my pursuit of this work in addition to my JHU/APL duties. I am also forever grateful to Jeffrey Chavis of JHU/APL for his guidance, having navigated the D.Eng. program in its first cohort.

Versions of the chapters in this dissertation have appeared in various publications or been submitted to various venues. I would like to take this opportunity to thank my co-authors on each publication: Avi Rubin [60] [61] [62], Matt Elder [60] [61] [62], Tony Dahbura [60] [61] [62], Lanier Watkins [60] [61] [62], and Matt Greene [60] [61] [62]. These publications and submissions may have been reproduced in whole or in part in this dissertation. I would also like to thank all the JHU/APL reviewers for taking the time to read this dissertation, as well as abstracts, papers, slide decks, and proposals: Wendy Blum, James Curbo, Peter Dinsmore, Matt Elder, Dave Sames, Don Vislay, Meghan Warner, and Cynthia Widick.

I would like to give special thanks to my wife Diane, and my daughters, Caroline and Allison, whose endless support allowed me to focus on my studies. Go Birds!

Lastly, I would like to thank my parents, Aksel and Tea Epp Siil, and grandparents, Peeter and Lydia Siirmets, who raised me and, each in their own way, gave me the gifts of integrity, honesty, tenacity, and curiosity. Above all, they motivated me to challenge myself and always strive to improve. You've all been gone a long time, but I remember those formative moments as if they were yesterday.

.

Dedication

This dissertation is dedicated to my wife Diane and my two daughters, Caroline and Allison, for the love, support, and sacrifices that allowed me to pursue and realize this long-sought goal. I would also like to dedicate this work to my parents, Aksel and Tea Epp Siil, and grandparents, Peeter and Lydia Siirmets, who taught me the value of hard work and the joy of something well-earned. I would not be where I am without their commitment to providing me the best education possible and preparing me for any challenge.

Table of Contents

List of Tables

List of Figures

1 INTRODUCTION

This dissertation investigates improving safety and mission assurance in autonomous vehicles (AVs), particularly those that operate with people on board or in the vicinity. Because of the human element, such an AV cannot simply be sacrificed if it suffers a system failure or comes under cyber-attack, like a robotic drone in a remote area can be, to prevent human injury or damage to more-valuable property.

Exploring a fictitious, but plausible, autonomous underwater vehicle (AUV) use case, we propose solutions to aid AUV designers, manufacturers and operators in expressing safety and mission objectives, determining which systems support those objectives, and experimenting on the safety/mission impacts of system failures in simulation before expensive AUV fabrication. We strive to create a formalized mission-assurance discipline for AUVs, built on the methodology and technologies proposed in this work, and enable the extension of that discipline to AVs in general as other tools are developed for the air, land, and space domains.

1.1 Autonomous Vehicle Concepts and Definition

An AUV is one of many kinds of AVs, and all AVs are composed of one or more cyber-physical systems (CPSs). A CPS is the tight conjoining of and coordination between computational and physical resources. In layman's terms, a CPS can be viewed as one or more computers connected to sensors and actuators (motors, servos, etc.) so that the system can interact with the physical world. The National Science Foundation envisions that the CPS of tomorrow will far exceed those of today in terms of adaptability, autonomy, efficiency, functionality, reliability, safety, and usability [19].

An *autonomous* cyber-physical system (ACPS) learns and adapts to dynamic environments and evolves as the environment around it changes. In comparison, an *automated* system operates within well-defined parameters and is very restricted in the tasks it can perform [45]. A car's (non-adaptive) cruise control or emergency airbag restraints are examples of automated systems. A self-driving car, on the other hand, is an ACPS.

Referring to Figure 1-1, abstractly an ACPS consists of an autonomy engine (AE), which is some combination of processors, software and possibly external communications mechanisms, e.g., to a cloud-based compute capability. The AE is connected to sensors and actuators that inform it about its environment and allow it to take actions in that environment, respectively. This dissertation does not investigate the internal operation of the AE, only the AE's role in ACPS operation and the consequences of the AE being faulty or compromised, e.g., by cyber-attack.

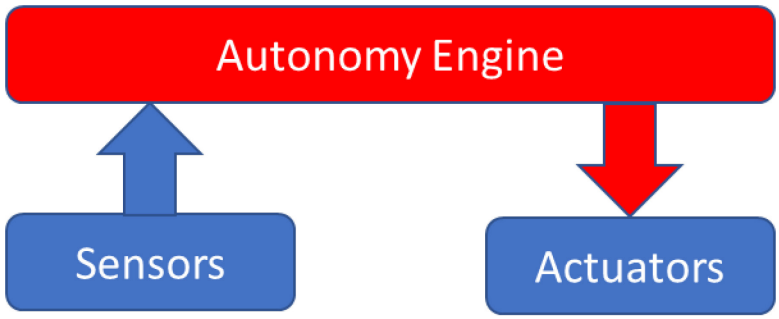


Figure 1-1 Abstracted autonomous cyber-physical system (ACPS)

While the abstraction in Figure 1-1 can describe any ACPS, this dissertation focuses on AUVs, which come in many shapes and sizes, with examples shown in Figure 1-2.

A complex vehicle like an AUV is built with a mission in mind. The mission might be to survey cables or pipelines, or it may be something for the military. The mission of the AUV in this dissertation is tourism. Touring passengers rely on underwater vehicles to take them to see interesting sights, like shipwrecks, sea life and geological formations. Chapter 2 describes the tourism scenario in detail.

Figure 1-2 Example AUVs from ECA Group [32]

1.2 Safety and Assurance Concerns of Autonomous Vehicles

While an AV is performing its mission, software flaws in the AE, or cyber-attacks against it, may cause unexpected, possibly dangerous, behavior that can lead to loss of the AV or damage to property. If humans are onboard or in the vicinity, the possibility of injury or death becomes a significant concern. Manufacturers of AVs that may interact with humans must convince operators, passengers, and the public at large that their products, which are highly reliant on complex software to operate, are safe despite

possibly flawed software or cyber-attack. And, while the AUV tourism scenario in this dissertation is currently fictitious, real-world concerns are driving AV manufacturers to make safety a top priority [25].

Many AV capabilities, e.g., following roads, recognizing and obeying signage, avoiding pedestrians and other vehicles, come from what we collectively call artificial intelligence and machine learning (AI/ML). At present, AI/ML characteristics are often a black box where even the developers who wrote the software may not know the reasoning behind every decision it makes [14] [58]. This raises the concern that, though an AV may have behaved well until now, there is no guarantee it won't exhibit unexpected behavior in the future. Standards [34] are emerging to address AI/ML transparency, but we consider them too immature at this time to provide adequate assurances.

In addition to potential AI/ML-related and other software flaws, there is a realistic possibility of the AE being compromised by direct cyber-attack or malicious software inserted during its development lifecycle. Recent supply-chain attacks [59] demonstrate that attackers are able and willing to infiltrate organizations that develop mission-critical software. Targeting a high-profile self-driving car company, or the manufacturer of AVs used by an adversary's critical infrastructure or military, is entirely plausible.

Taking the above concerns to the extreme, we assume an attacker has unfettered access to affect the AE in any way they desire. This could be via witting or unwitting insiders, supply chain compromise, or exploitation of vulnerabilities in deployed systems. Whatever the case, this dissertation places no trust in the AE, instead investigating what is required to remediate or mitigate the effects of a defective or malicious one. Worst case assumptions are made for all potential actions of a compromised AE.

We assume AE technology and development practices will not be worthy of sufficient trust well into the foreseeable future, requiring a means to monitor AE behavior and intervene with corrective or preventative actions should it misbehave. To support this need, we believe that predictable high-assurance systems can be built, with their development facilities, supply chains, etc., protected from attack, but not to the extent of fabricating an entire AV or something as complex as an AE. Instead, we

believe that, much like the highly resilient "black box" recorders on airliners, adding a high-assurance resilience module (RM) would greatly increase AV safety and mission quality in foreseen and unforeseen circumstances. An RM, shown in Figure 1-3 added to the abstracted ACPS, monitors the AE's behavior and takes corrective action, overriding the AE, if safety or mission objectives are put at risk.

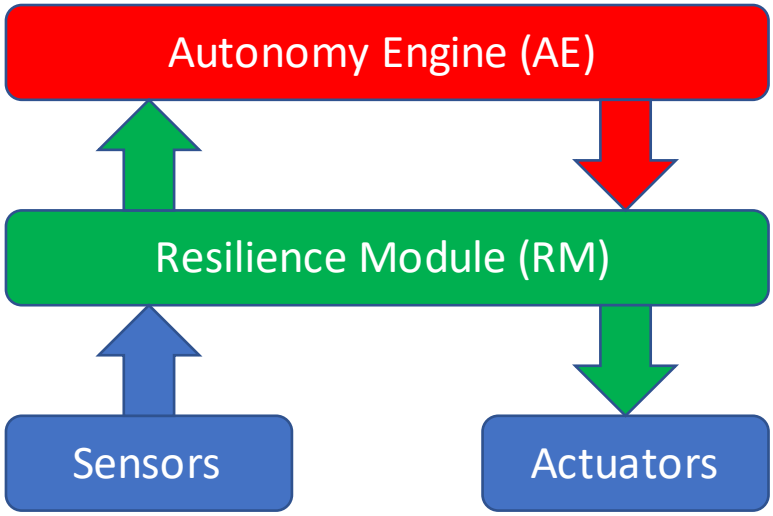


Figure 1-3 Resilience module between the autonomy engine and sensor/actuators

The RM would presumably be built using formal verification [24], source-code analysis [65], and other techniques that improve software assurance. We suspect, given the state of the art for such techniques, the RM will be less capable than its AI/ML-enabled AE counterpart. However, the RM should at least assure safety and that the most-important mission objectives (MOs) are met. This requires determining what those MOs are and what AV systems are required to meet them.

1.3 Vision and Approach

The issues identified and described above cover multiple technological areas and phases of the AUV lifecycle, requiring a holistic approach for addressing them that begins in the MO-definition phase and ends in day-to-day operations. Our vision is to explore these issues from the mission and system

perspectives, relating MOs to the systems that support them in operating environments where multiple vessels interact, sometimes at close range. This dissertation explores each issue and introduces technologies and a methodology that, in combination, address the holistic AV mission-assurance challenge. We capture the primary thrusts of our approach in [60] [61] [62], which are summarized in the subsections below.

1.3.1 Mission Assurance for Autonomous Underwater Vehicles

As AVs continue to evolve, they will interact ever more with humans as passengers, or as incidental bystanders in areas of AV operations. This requires assurances that the AVs will behave safely. Moreover, for people to depend on AVs for transportation, delivery of goods, etc., they must achieve their mission objectives with high likelihood in both foreseen and unforeseen situations.

AI/ML-based AEs are poorly understood complex systems that behave well most of the time, fail disastrously on occasion, and offer insufficient guarantees that they won't fail when they are needed most. In addition, AEs are susceptible to cyber-attack like any other computer, both in the traditional cybersecurity sense of software vulnerabilities, flawed architectures, etc., and against their sophisticated AI/ML algorithms. To mitigate AE risks until these technologies are more trustworthy, high-assurance systems must be incorporated into AVs to monitor them and act if situations arise that put safety or the mission at risk.

We detail a fictitious, but plausible, scenario where AUVs are being incorporated into a mission currently performed by human-piloted submersibles. The scenario frames the mission-assurance investigation of this dissertation. We introduce a testbed for experimentation with the scenario, propose an approach for remediating or mitigating AE vulnerabilities, and present mission-assured AUV (MAAUV) functionality, implemented as an additional system, to improve our AUV's mission.

1.3.2 Cost-Effective Mission Assurance Engineering Through Simulation

Developing an AUV to operate in an environment that is only partially known, maximizing the number of potential unknowns addressed while keeping the development effort cost-effective, is a significant challenge. Valuable knowledge is accrued with operational experience and factoring lessons learned from unexpected situations into maturing the AUV. However, this approach is expensive and time-consuming with actual vehicles, assuming they are available for testing, and raises the risk of injury or property damage. Moreover, prototypes of new vehicles would need to be built before any operational experience could be gained, consuming resources with no immediate return on investment when a project is just getting off the ground. We introduce DESCRETE (Digital Environment for Simulated Cyber Resilience Engineering, Test and Experimentation), a simulated testbed for AUV mission-assurance experimentation, present MAAUV experimentation results, and demonstrate the value of the testbed in collecting the results at significantly less cost.

1.3.3 A Methodology for Iterative Mission-Assurance Refinement Analysis

Mission assurance is an abstract concept not always well-understood in the context of system failures that affect it. Many methodologies examine safety and security at a system level, but only a handful consider operator requirements or mission. Identifying mission-critical systems would allow AUV designers and manufacturers to focus limited resources, as well as work with AUV operators to refine and prioritize mission objectives. We propose an iterative mission-assurance refinement analysis (IMARA) methodology to capture mission objectives, tie them to systems, characterize the impacts of system failures, and weigh remediation and mitigation options. We apply the methodology to our MAAUV use case, and propose an approach for automating the methodology using popular model-based system-engineering tools.

1.4 Outline of this Work

Each chapter in this dissertation explores an aspect of our MAAUV problem, and our proposed solutions, tools for experimentation, or experimental results.

- Chapter 2 introduces our MAAUV scenario, DESCRETE testbed, AUV-candidate submersibles, required AUV capabilities, and a mission-assurance scoring system.
- Chapter 3 introduces the RM concept and applies it to the MAAUV scenario.
- Chapter 4 presents and analyzes our experimental results using the DESCRETE testbed.
- Chapter 5 introduces the IMARA methodology.
- Chapter 6 presents a worked example of the IMARA methodology on the MAAUV use case.
- Chapter 7 summarizes the preceding topics investigated in this dissertation.
- Chapter 8 lists the references cited in the research and performance of work described herein.

2 MISSION ASSURANCE FOR AUTONOMOUS UNDERWATER VEHICLES

2.1 Introduction

The ubiquity of passenger-carrying autonomous vehicles (AVs) is all but inevitable, and the assurance that they will behave in a safe manner with respect to their passengers, as well as bystanders incidentally exposed to them, is moving forward, albeit slowly. While this is good news, AV benefits aren't fully realized if the only option for dealing with a faulty one, or one compromised by malicious software, is to shut it down. This may be (somewhat) acceptable for the owner of an expendable low-cost drone, but not for passengers left stranded far from home because the car was hacked. They may be safe from dangerous behavior by the car, but they are at least inconvenienced and, depending on where they are stranded, possibly exposed to other risks. AVs, like most things built by humans, are built for a purpose; call it a mission. Being able to perform the mission, or at least part of it, when a fault appears or the AV undergoes cyber-attack should be a factor in determining the suitability of the vehicle for the mission.

Our contributions are as follows: (1) we introduce a well-defined autonomous underwater vehicle (AUV) scenario to frame our mission-assurance investigation; (2) we introduce a simulated testbed that enables us to measure improvements in safety and mission assurance using objective scoring criteria; and, (3) we introduce the AUV models we've simulated that are similar to modern AUVs, thus enabling our results to be applied readily to real-world scenarios.

This remainder of this chapter is organized as follows. Section 2.2 provides an overview of the scenario and mission. Section 2.3 introduces our simulated testbed for mission-assurance experimentation. Section 2.4 describes the testbed's features for safety and collision detection. Section 2.5 presents our AUV options, the ones we selected, and future candidates. Section 2.6 details the AUV's mission functionality. Section 2.7 details how missions are measured, or scored. Section 2.8 presents the known limitations of our approach. Section 2.9 summarizes related work, which is reviewed in more detail in later chapters. Section 2.10 summarizes the chapter's important points, and Section 2.11 presents future work.

2.2 AUV Tourism Mission Overview

Our AUV's mission involves a fictitious, but plausible, company that operates human-piloted submersibles to tour *points of interest* (POIs) in an operating area, a notional example of which is shown in Figure 2-1. In addition to POIs, the figure shows piers, as well as surface vessels and other submersibles.

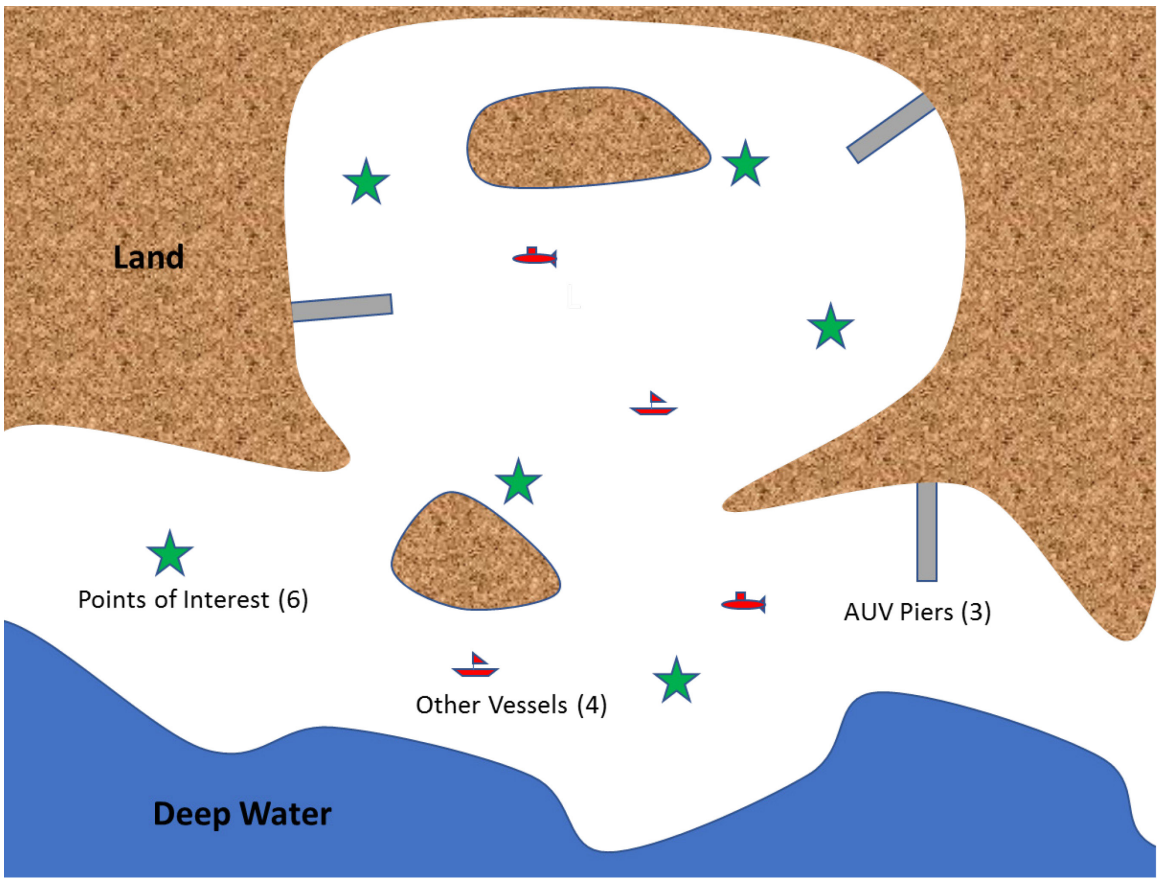


Figure 2-1 Notional AUV tourism scenario

The company would like to replace its human-piloted submersibles with AUVs, and this dissertation is framed as a study commissioned to better understand AUV operations, and their risks, with the goal of developing a mission-assured AUV (MAAUV) that is safe for its passengers, as well as trusted to complete high-quality tours in expected and unexpected situations.

The tour company's current submersibles are dispatched regularly from piers to visit as many POIs as possible within a given time period, called the *maximum-allowed tour duration* (TD-Max). TD-Max is based on passenger physiological limitations, as well as how long they are willing to spend on a tour. It is significantly less than vehicle endurance, i.e., fuel, battery, or oxygen capacity, which for that reason we do not investigate further in this dissertation.

There is also *minimum-required tour duration* (TD-Min) that, along with the number of POIs visited, represents passenger satisfaction. For example, passengers would feel short-changed if the tour visited all the POIs, but raced past not allowing time to admire or photograph them. We included TD-Min in our tour scoring in Section 2.7, but experimenting with it is planned for future work.

The tour company belongs to a local partnership with several, but not all, other such tour companies in the area. The partnership has invested in Harbor Control Services (HCS) to provide acoustic range-finding for all *partner submersibles* using a set of fixed stations, which allow them to determine their absolute and relative positions accurately. *Non-partner* submersibles nor surface vessels participate in the acoustic range-finding service and, therefore, non-partner submersibles can only be detected by sonar. Surface vessels must monitor HCS on marine radio to coordinate clearing a given area to allow a submersible to surface. HCS also informs submersibles when an area is clear.

The submersibles operate near their design limits in depth and against currents, avoiding deep water due to the risk of becoming incapacitated and descending below their maximum operating depth. The submersibles must also avoid fixed obstacles, each other, and surface traffic while on tours. The risks common to both the current human-piloted submersibles and the AUVs contending to replace them are:

- Running aground, bottoming out, hitting a pier, or straying into deep water
- Colliding with a surface vessel or another submersible
- Coming too close to, or colliding with, a POI

AUVs fit the abstract autonomous cyber-physical system (ACPS) model shown in Figure 1-1. They have a collection of processors and associated software that together implement an autonomy engine (AE). They have sensors to provide situational awareness to the AE, like position and proximity to obstacles. And, they have actuators with which to manipulate physical controls like propellers and rudders. Replacing the human-piloted submersibles with AUVs adds risks that would not exist with, or would be handled by, a human pilot. These additional risks stem from software flaws in, or cyber-attack against, the AE, and include:

- Not visiting some or all the POIs
- Staying out too long, or returning too quickly

Some of these risks can lead to danger, e.g., an AUV could navigate away from a pier and hover submerged indefinitely. Others are just bad for business. Yet another risk increased by replacing crew with autonomy is ride quality. A malfunctioning AE could pilot the vessel in ways to cause passenger discomfort, or even harm. Ride quality is out of scope for this dissertation.

Given the risks, the tour company must be assured that a level of safety and customer satisfaction can be achieved.

2.3 The DESCRETE Testbed

To experiment on mission assurance, we developed the Digital Environment for Simulated Cyber Resilience Engineering, Test and Experimentation (DESCRETE) testbed in which test environments can be configured and test events executed with simulated AUVs and other test articles. DESCRETE is built on the Gazebo robot simulation environment [73] and a Gazebo-based unmanned underwater vehicle (UUV) Simulator [33] that implements underwater physics and provides example UUVs. Both Gazebo and the UUV Simulator are open-source software.

DESCRETE contributes to the state of the art in AUV development and operation by making maritime cyber-resilience engineering more representative of the real world, without using actual vehicles that are

valuable and hard to come by. Using DESCRETE, we can simulate vehicles and environments at greatly reduced cost, putting no expensive systems at risk. We can test fault-free scenarios to evaluate algorithms, inject faults representative of, e.g., cyber effects, evaluate scenario plausibility and impact, and develop mitigations.

DESCRETE combines maritime simulation with fault injection in a unique testbed built to model and analyze resilient autonomy, especially cyber resilience. Its fault-injection framework can simulate complex cyber effects, while allowing users to choose vehicle fidelity as they see fit. By building simple models first, one can get started quickly and determine where to increase fidelity, given that cyber effects are often limited to one or a handful of systems, rather than having to build complicated high-fidelity vehicles from the start. Medium fidelity is good for system-level experimentation, i.e., what effect a fault can cause, not how it does it.

We built the environments for our AUV tourism experimentation using Gazebo and UUV-Simulator features to model the seafloor, POIs, piers, and AUVs. Every model has visual elements for rendering them in the testbed, examples of which are shown in Figure 2-2. The models also have collision and inertial elements to describe the geometry for collision checking and to apply physics models, respectively. The visual and collision elements can differ. For example, the axes in the figure can be seen by users, as well as any cameras on the models themselves. But, they cannot be collided with.

Using open-source texture images, we modeled a realistic seafloor. For fixed obstacles, we used simplified topologies to reduce testbed complexity. POIs are represented by hemispheres on the sea floor. Piers adjoined to land are represented by rectangular planks placed to provide enough room for an AUV to approach and dock while remaining clear of the seafloor. There are also open-water “piers,” represented by fixed floating disks, where surface vessels deploy and retrieve AUVs to eliminate the need for them to waste time and fuel traveling to the test area. Vehicles are modeled to be sufficiently similar

to their real or fictitious counterparts that physical forces they produce, like thrust, and forces that act on them, like drag and buoyancy, are of the required fidelity for the simulations.

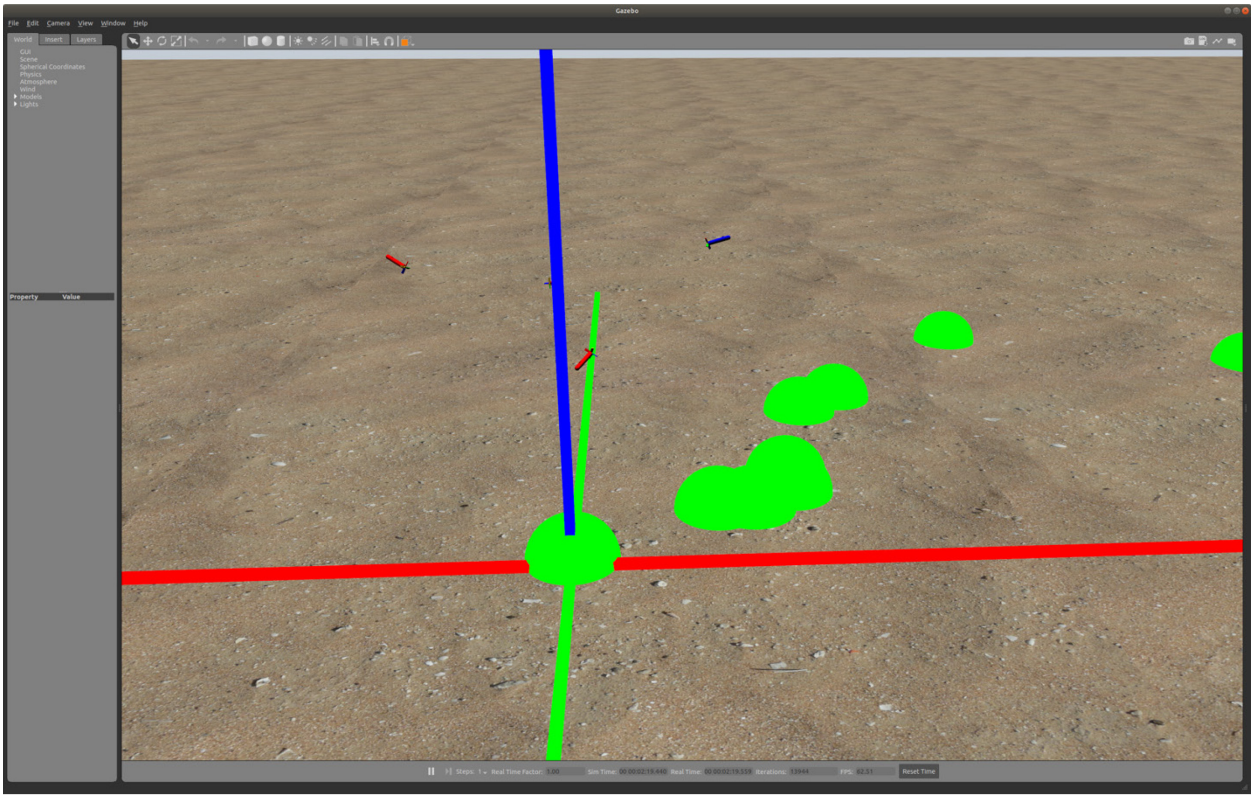


Figure 2-2 AUVs, in blue and red, performing tours amongst POIs, in green. The red, green, and blue lines are X, Y, and Z axes, respectively, for reference, and are transparent to the vehicles.

2.4 Separation and Boundaries

Each simulated vehicle in DESCRETE avoids collisions by maintaining *separation* from the seafloor, fixed obstacles, and other vehicles. Separation in DESCRETE is very similar to its use in the air-traffic control (ATC) system [2], where it is intended to keep aircraft far enough apart that unexpected maneuvers or loss of situational awareness by one aircraft doesn't immediately endanger other aircraft and gives everyone time to react. A *loss of separation* (LOS) occurs when the distance between a vehicle and some other object falls below the required minimum.

To support collision avoidance, as well as visiting POIs and docking with piers, one of DESCRETE's significant contributions is a set of boundaries, depicted in Figure 2-3. Everything in a test environment is surrounded by a set of concentric boundaries, similar to the boundaries used by Safeguard [23]. Crossing boundaries can change vehicle state, precipitate vehicle actions, or affect mission quality. The boundaries are described below. A complete description of DESCRETE's features is provided in Appendix A.

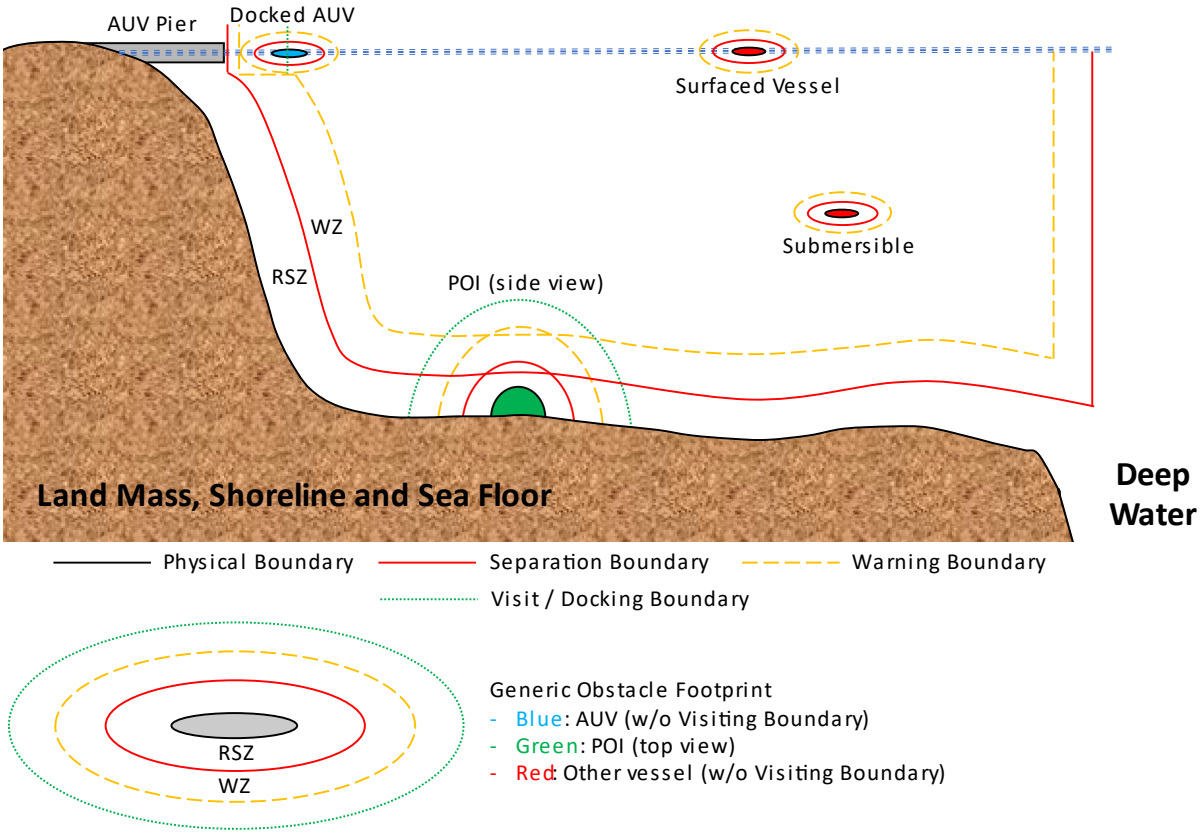


Figure 2-3 Fixed obstacle and vehicle boundaries (not to scale)

The DESCRETE boundaries and effects of crossing them, starting closest to a given object, are:

- **Physical Boundary (PB)**: This represents the physical object. Crossing it is a collision for DESCRETE, but not necessarily for Gazebo. Though entering deep water is technically not a collision, that case is handled the same way to simplify the testbed implementation.

- **Separation Boundary (SB)**: This represents the minimum distance that must be maintained from the object. Crossing it is LOS and enters an object's reduced-separation zone (RSZ).
- **Warning Boundary (WB)**: This is used by the resilience module (RM) described in Chapter 3 and represents the range at which the RM warns the AE of a potential LOS with the object. Crossing it enters an object's warning zone (WZ).
- **Visiting Boundary**: Only used for POIs, crossing it represents a POI being visited.
- **Docking Boundary**: Only used for piers, crossing it and reducing speed over ground to zero causes a vehicle to become docked.

2.5 AUV Selection

The first AUV we modeled was similar in passenger capacity, endurance, and performance to current human-piloted tourism submersibles (T-SUBs) [68]. An actual T-SUB and the MAAUV-01 simulation we built for experimentation are shown in Figure 2-4 and Figure 2-5, respectively. MAAUV-01 is 12 m long and carries 24 passengers as far down as 100 m at a speed of 1.5 m/s (3 knots).In our fictitious tour-company scenario, each MAAUV-01 would be a one-for-one replacement for a T-SUB until, if the plan is successful, all the human-piloted vessels are retired.

Challenges in constructing MAAUV-01 led us to re-think our vehicle choice, and we decided to experiment on a more typical AUV first. MAAUV-Lite, shown in Figure 2-6, is based on an existing UUV model [33] that we extended by adding a sail (the bump on top near the bow) and coloring the propeller blades to facilitate determining the vehicle's orientation and speed. A similar "partner" submersible model was built (see Section 2.2), identical in all aspects except that the hull color is red.

MAAUV-Lite's length is 2.45 m, about a fifth of MAAUV-01's. Its maximum speed and depth are the same as MAAUV-01's, and its maneuverability is proportional. With a smaller model, we were able to reduce the size of the experimentation environment, allowing us to have shorter, and therefore more, simulation runs. We used the MAAUV-Lite and related partner models for all of our experimentation.

Figure 2-4 A typical tourist submersible (T-SUB) [68]

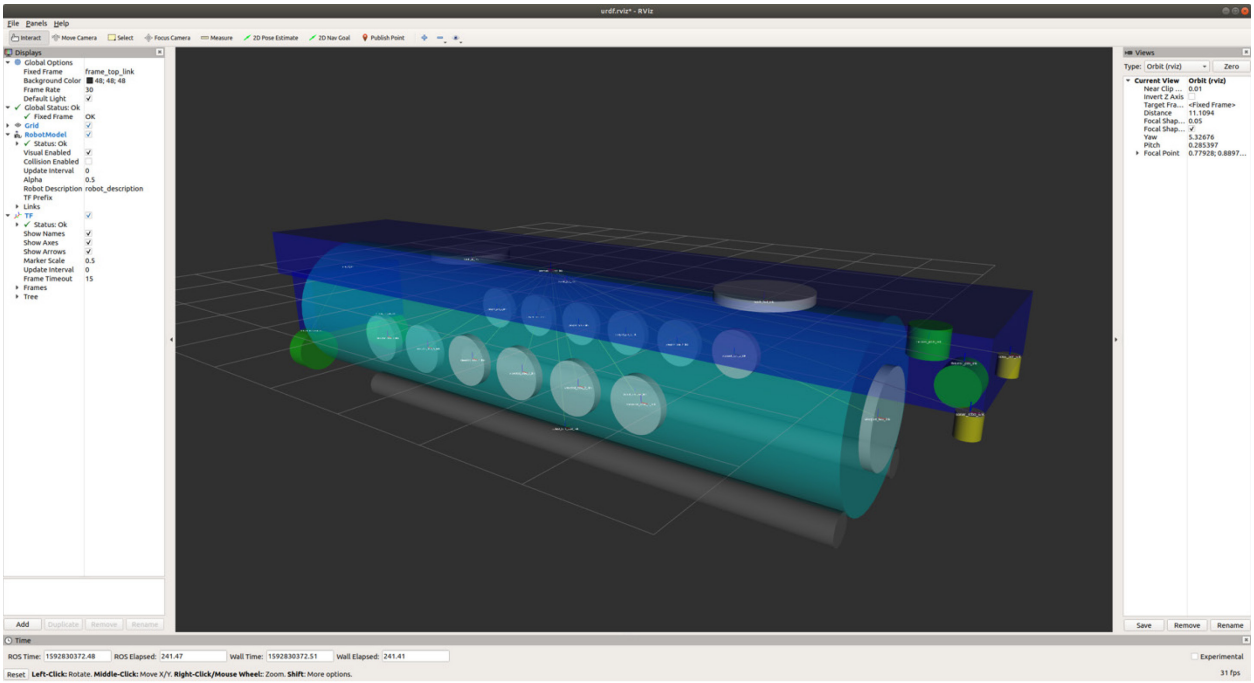


Figure 2-5 The MAAUV-01 simulation of a T-SUB

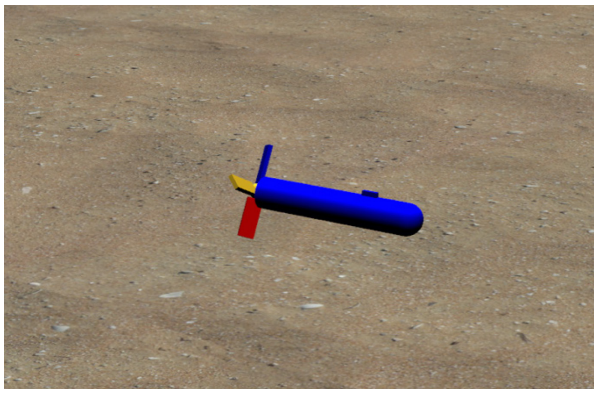


Figure 2-6 The MAAUV-Lite model for simulation experimentation

While MAAUV-Lite proved useful in our experimentation, it fell short of fully representing our goal to advance mission assurance of passenger-carrying AVs, specifically where the passengers are not necessarily trained operators of such vehicles. There are single-passenger submersibles roughly the size and shape of MAUUV-Lite, but the passenger requires scuba gear, which doesn't fit our tourism scenario where a layperson could be onboard, versus a professional diver.

We found a less-exact match, but still a suitable one, in the U-Boat Worx NEMO submersible [50], shown in Figure 2-7. At 2.8 m, the NEMO is less than 15% longer than the MAAUV-Lite, but more spherical than cylindrical. The NEMO has the same depth and speed limitations as the MAAUV-Lite, and can carry a pilot and passenger as comfortably as a car. An autonomous NEMO could replace the pilot with a second passenger. In addition to our tourism scenario, the NEMO can perform exploration, surveillance, inspection, even light manipulation with an optional robotic arm. Though limited in depth compared to "worker" submersibles like the DeepWorker 2000 [57] shown in Figure 2-8, which is also similar in size to the MAAUV-Lite, the NEMO still addresses a number of missions that could benefit from having a human observer or scientist on board, without requiring piloting skills.

Figure 2-7 The NEMO personal submersible [50]

Figure 2-8 The DeepWorker 2000 submersible [57]

2.6 AUV Mission Functionality

For each tour described in Section 2.2, the AUV performs the following functions. A tour starts with the AUV generating a *departure tour plan* (DTP) that describes the tour legs to traverse from the pier to the first POI, between the POIs, and returning to the pier. Tour legs are made up of one or more segments, with each segment having a course, depth and expected speed over ground.

A valid DTP must meet the following criteria:

- Visit all POIs
- An estimated time enroute (ETE) within the TD-Min/TD-Max limits
- Return passengers to the pier

When the DTP is approved for activation, it becomes the *active tour plan* and the AUV departs on the first leg of the tour. On reaching each POI, the AUV visits it and departs on the next tour leg. After visiting the last POI, the AUV returns to the pier from which it departed, or possibly another pier if, e.g., the ETE for returning to the departure pier exceeds TD-Max. On reaching a pier, the AUV docks, ending the tour.

Throughout the tour, the AUV assesses whether the active tour plan is still *achievable*, i.e., has an ETE no more than TD-Max. If the active tour plan becomes unachievable, the AUV generates an *enroute tour plan* (ETP), eliminating or re-ordering POI visits so as not to exceed what remains of TD-Max. If all remaining POIs visits must be eliminated, the only valid ETP is one that returns the AUV to a pier. In that case, the ETE is allowed to exceed TD-Max, though a pier with a lesser travel time is preferred.

As with the DTP, once a valid ETP is approved, it becomes the active tour plan and the AUV begins executing it. The cycle of assessing achievability and generating additional ETPs, if necessary, continues until all POIs, if any, on the latest active tour plan are reached and the AUV has returned to the pier. At that point, the tour is ended and scored using the equations in Section 2.7.

2.7 Tour Scoring

A score based on the DTP requirements in Section 2.6, and any LOS, can be computed using the following equations to determine how well the AUV did on a given tour:

$$Tour\ Score = \left\lfloor 100 \times \sum_{i=1}^{N} \frac{P_{A_i}}{\sum_{i=1}^{N} P_{1_i}} \cdot \left(\frac{T_1}{\max(T_A, T_1)}\right)^{M_1} \cdot \left(\frac{\min(T_A, T_0)}{T_0}\right)^{M_0} \cdot R_A \cdot LOS \right\rfloor$$

$$LOS = \begin{cases} \prod_{j=1}^{L} \frac{\min(S_{A_j}, S_{0_j})}{S_{0_j}}, & L > 0 \\ 1, & L = 0 \end{cases}$$

Scoring constants (see below):

M_0 = *Weighting exponent for not reaching TD-Min*

M_1 = *Weighting exponent for exceeding TD-Max*

Test Environment Variables:

N = *Number of POIs in the test environment*

P_{1_i} = *Level of Interest (LOI) of POI (i)*

S_{0_j} = *Minimum required separation from obstacle (j)*

T_0 = *Minimum-required tour duration (TD-Min)*

T_1 = *Maximum-allowed tour duration (TD-Max)*

Test Event Variables

L = *Number of losses of separation (LOS)*

P_{A_i} = $\begin{cases} \text{LOI of POI (i)}, & if\ POI\ (i)\ was\ visited \\ 0, & otherwise \end{cases}$

R_A = $\begin{cases} 1, & AUV\ returned\ passengers\ to\ a\ pier \\ 0, & otherwise \end{cases}$

S_{A_j} = *Minimum actual separation from obstacle (j)*

T_A = *Actual tour duration*

These equations have the following properties:

- Visiting all the POIs within the tour-duration limits with no LOS yields a perfect score of 100.
- Any collision, i.e., $S_{A_j}=0$, yields a score of zero, as does not returning passengers to a pier.
- M_0 equals 0 in this dissertation, because we did not experiment with TD-Min.
- M_1 equals 2, in this dissertation, though no actual tour durations exceeded TD-Max.

Also, if there is no LOS and a tour's duration is within the limits, the score is based entirely on how many POIs were visited, and how interesting they were compared to all the POIs in the environment. Chapter 4 presents the results of a variety of test tours run in DESCRETE, including their scores. An analysis of how the scores were affected by sub-optimal tours is also presented.

2.8 Known Limitations

Sensors, actuators, and other components are subject to wear and tear, accidental breakages, and other natural phenomena that could cause them to fail. To manage the scope of our investigation, we assume classical resilience engineering, along with proper inspection and maintenance procedures, are used to eliminate such failures for these components. Also, the solutions proposed in the remainder of this dissertation rely on trusted communications with trusted service providers, but do not account for accuracy or availability issues with them. These are all topics for future research (see Section 2.11).

2.9 Related Work

We published our AUV tourism scenario and approach for developing a MAAUV in [61], the results of our initial MAAUV experimentation in [60], and seek to publish our iterative mission-assurance refinement analysis (IMARA) methodology [62], which is described in Chapter 5 and applied in Chapter 6. Others have addressed autonomous-vehicle assurance. Some of their vehicles do not carry passengers, however, so they focus on providing protection against damage from the vehicle, but not damage to the vehicle itself. Several approaches consider the vehicle expendable, which the MAAUV in our tourism scenario is not.

Approaches to runtime safety include self-adaptive systems [4] [10] [27] [30] [42] [54], dynamic assurance cases [4], as well as state-machine based [39] and rule-based [75] approaches. Several methodologies can be applied to designing a MAAUV, including STAMP and STPA [43], STPA-Sec [76], and STPA-SafeSec [29]. Some have been applied to autonomous maritime systems [22] [77]. There are also many co-engineering methods for cyber-physical system (CPS) safety and cybersecurity [13] [28] [40], as well as to elicit mission objectives (MOs) [12] [21]. These approaches and methodologies are discussed more in subsequent chapters. See Section 3.7, Section 4.7, and Section 5.4.

2.10 Conclusion

This chapter introduced a well-defined AUV tourism mission to frame our MAAUV investigation. It also introduced our DESCRETE simulated testbed, which enables us to modify our AUV models with relative ease and measure improvements in safety and mission assurance using objective scoring criteria. We chose models similar to modern AUVs so that our results could be applied readily, and presented candidates for more-representative models in future work. Our use-case mission and tools have positioned us well for investigating AUV mission assurance in the chapters that follow.

2.11 Future Work

Our AUV tourism scenario has defined a complex problem with many facets, several of which we have not investigated in this dissertation. These are summarized below.

We assume an attacker can affect some aspects of the MAAUV's operating environment and generate false sensor inputs or deny real ones, e.g., HCS signals. This is discussed further in Section 4.9.

Based on reporting requirements for instrument flight rules (IFR) flight [41], we identified additional tour-plan parameters to explore that provide more granularity for how well a tour is being executed. Instead of just a single tour-plan ETE, tour-leg or even tour-segment ETEs would enable detection of anomalous behaviors, as well as measurement of a MAAUV's performance more uniformly throughout

the tour. Constraints on speeds and POI-visit loitering durations would further enable detection and measurement.

While the NEMO mentioned in Section 2.5 represents an exciting evolution in personal submersibles, it is not economical for middle-class AUV tourism. As of November 2021, a refitted (used) 24-passenger T-SUB, like the one in Figure 2-4, can be purchased for $1.25 M [69]. A brand-new two-passenger NEMO that's comparable (sonar is extra) costs €1.01 M ($1.15 M) [7] [50]. Prices listed on web sites for expensive items like submersibles are not necessarily accurate; but, the discrepancy would have to be an order of magnitude just for the NEMO's purchase price to be competitive. There are also operating costs to consider, which we suspect are significantly greater for a dozen NEMOs than one T-SUB to carry the same number of passengers, assuming demand was high enough for the T-SUB to tour fully loaded. Based on these observations, a practical MAAUV investigation needs to continue developing the MAAUV-01.

Expanding on the preceding, a real-world engineering solution factors in cost along with functionality and assurance requirements. An organization's financial resources are limited, and high assurance may not be economical. Cost is ignored in this dissertation. Instead, we assume that the cost of a high-assurance solution is distributed over many units; or, the assurance cost is acceptable due to, e.g., the importance of the mission or government regulations requiring the high assurance. Cost modeling, or measuring the real cost of developing high-assurance systems for our AUV tourism mission would complete the picture of how palatable a MAAUV would be to tour operators and AUV manufacturers.

3 A RESILIENCE MODULE FOR IMPROVED MISSION ASSURANCE

3.1 Introduction

The abstract autonomous cyber-physical system (ACPS) shown in Figure 1-1 contains an autonomy engine (AE) meant to process sensor inputs and command actuators to perform a mission. Based on our arguments in Section 1.2, we consider the AE insufficiently trustworthy to ensure the safety, never mind mission success, of an autonomous underwater vehicle (AUV) like the one described in Section 2.2. Our approach to improving safety and mission assurance adds a resilience module (RM) that monitors the AE's behavior and takes corrective action, overriding the AE, if safety or mission objectives are put at risk.

Figure 1-3 shows an RM in series between a potentially faulty or compromised AE and the AUV's sensors and actuators. A parallel architecture can also be used for the sensors, where the AE and RM share direct inputs, or have separate sensors each. For actuators, the RM requires ultimate control. This can be achieved by having the AE and RM in series, as shown in the figure, allowing the RM to inspect and possibly block commands before they reach the actuators. Or, the AE could be given direct access that the RM can cut off, if necessary, and then command the actuators using parallel connections.

A disadvantage of the parallel architecture for AE/RM actuator connections is that a "bad" command may have already been received and acted on before the RM takes over. This may not be important for actuators that operate slowly and can be reversed, like a ship's steering gear. It may be undesirable, however, for quick-acting irreversible actuators, like an airbag safety system. We use the in-series architecture shown in Figure 1-3 for our discussions throughout the rest of this dissertation.

Our contributions are as follows: (1) we introduce a general RM concept for ACPSs; (2) we introduce specialized functionality to improve the safety and mission assurance of our mission-assured AUV (MAAUV); and, (3) we propose an RM with that functionality for our MAAUV-Lite model, which we experiment on in Chapter 4.

The remainder of the chapter is organized as follows. Section 3.2 describes the RM's internal structure and Section 3.3 its interfaces. Section 3.4 describes the RM functionality we designed to execute the tourism mission in Section 2.6, and develop a specialized RM for our MAAUV. Section 3.5 discusses the potential impact of the RM concept being adopted by industry. Section 3.6 addresses known limitations of our approach and implementation. Section 3.8 summarizes our conclusions, and Section 3.9 presents future work.

3.2 Resilience Module Internal Structure

The RM's internal structure is based on the five functions of the NIST Cybersecurity Framework: Identify, Protect, Detect, Respond, and Recover (IPDRR) [52]. The identify function is not implemented explicitly, but represented by the RM design. Referring to Section 2.4 and Section 2.6, the tour-plan generation, validation, and approval process provides protect functions. Detect functions for separation and tour-plan achievability continuously monitor the AUV on its tour. Respond and recover functions are the actions the RM takes if the AE behaves unexpectedly, separation is lost, etc.

How the IPDRR functions are incorporated into the RM is shown in Figure 3-1. Their exact functionality is described in Section 3.4.

In addition to the IPDRR functions, the RM stores IPDRR and AUV safe operating envelope (SOE) information in a database. The SOE part of the database contains parameters and constraints the AUV must adhere to for safe operation, e.g., maximum speed and depth. The IPDRR part contains information required for those functions, including pre-generated tour plans in some of the RMs described below.

The RM also implements trusted external communications, called Harbor Control Services (HCS), which are described in Section 2.2. These communications are protected, e.g., via strong cryptographic authentication and message integrity mechanisms, and provide situational awareness information, as well as coordination with shore facilities and other vessels.

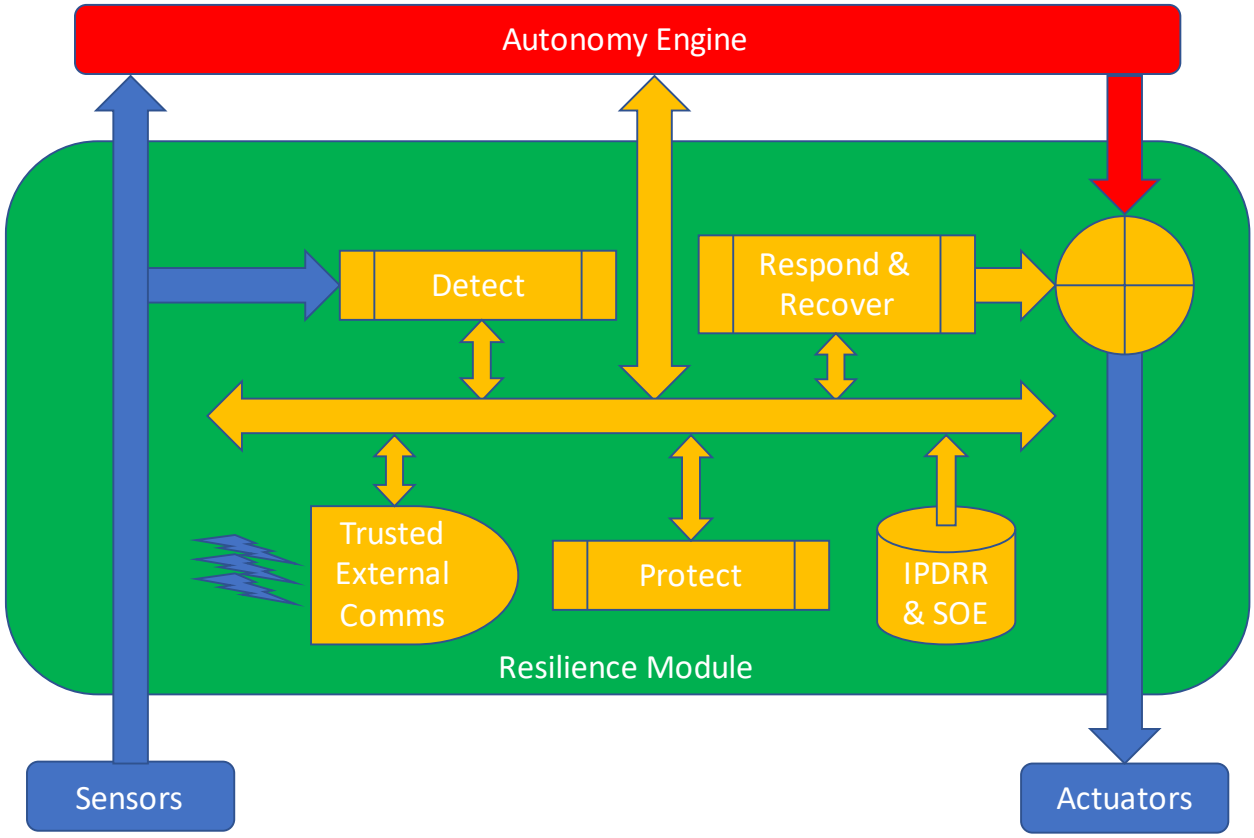


Figure 3-1 Resilience module internal structure

3.3 Resilience Module Interfaces

In our approach, the RM shares sensor inputs with the AE, and therefore must support the interfaces the sensors use to send data. Similarly, the RM can send commands to the AUV's actuators and must support their interfaces for receiving, and perhaps acknowledging, commands. Moreover, the RM must be able to interpret commands the AE sends through it to the actuators, so as to detect undesirable AE behavior and prevent those commands from passing through. The RM also has a two-way interface to the AE, i.e., the AE knows the RM is there and must interact with it to operate the AUV. Section 3.4 describes how the AE and RM interact via this interface, which must be well-defined and resistant to exploitation by a potentially malicious AE.

3.4 Resilience Module Functionality

A logical high-level view of the RM functions is shown in Figure 3-2, and the functions are described in the subsections that follow.

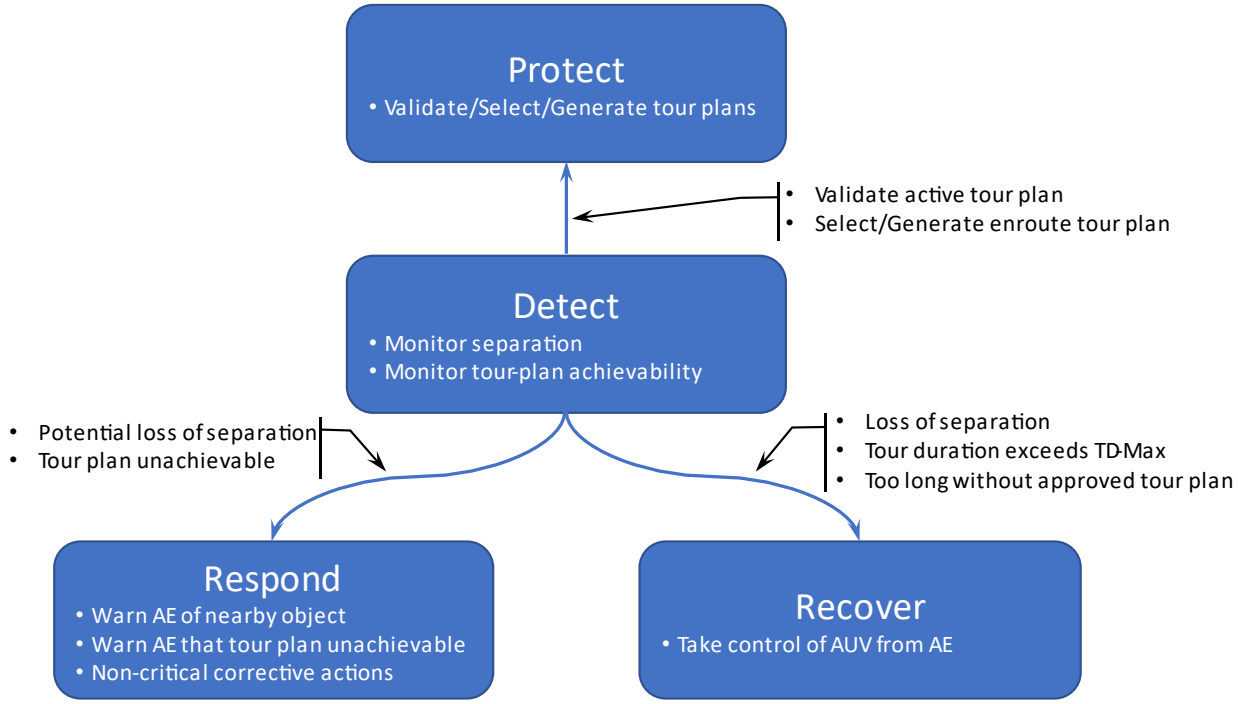


Figure 3-2 Resilience module functionality: blue lines show RM functions invoking other functions for the reasons described in the callouts.

The specific functions described in each subsection are a result of the iterative approach we used in our research, development, and experimentation. Not all of the functions were implemented. For example, if a function that was more effective at providing safety or mission assurance could be implemented with fewer resources than a less-effective one, the less-effective one was skipped. The functions that were implemented are identified in each subsection. Our experimentation on those functions is presented in Chapter 4.

3.4.1 Protect Functions: Tour Plan Validation, Selection, and Generation

Protect functions support the validation, selection, and generation of departure tour plans (DTPs) and enroute tour plans (ETPs), described in Section 2.6. The RM must approve tour plans, which allows the AE to activate and execute them. Without an active tour plan, the RM does not allow AE commands to reach the AUV's actuators.

3.4.1.1 Protect Function #1 (PR-1): Rudimentary Tour-Plan Validation

Protect function #1 (PR-1) performs rudimentary tour-plan validation. A valid DTP must visit all points of interest (POIs) with an estimated time enroute (ETE) between the maximum-allowed and minimum-required tour durations, TD-Max and TD-Min, respectively. A valid ETP need not visit all (or any) remaining POIs, but must have an ETE within the tour-duration limits, unless no remaining POIs are visited, in which case, the ETE may exceed TD-Max.

We implemented PR-1 and it was effective for DTPs. For ETPs, because PR-1 has no requirement to visit POIs, it does little to prevent an AE from submitting very poor plans (see Section 3.6).

3.4.1.2 Protect Function #2 (PR-2): Pre-Generated Tour Plan Selection

Protect function #2 (PR-2) takes a different approach to tour-plan quality. Instead of validating AE-generated tour plans, it uses a pre-generated tour-plan library. Either the AE voluntarily selects a tour plan from the RM's library, or the RM alerts the AE that it must select one before a recover function is invoked.

We did not implement PR-2, because the more-effective protect function described in the following subsection could be implemented more readily. We retain the PR-2 description below because it presents a novel approach to the tour-plan generation problem, similar to how standard arrival procedures are used in the air-traffic control (ATC) system [3].

PR-2 employs a tour-plan library generated by the AUV manufacturer or operator. For a given operating area, the tour-plan library is expected to contain a small number of DTPs, as few as one. Benefits of generating multiple DTPs include enabling the use of different piers, allowing operators to distribute

visits so a given POI is not overwhelmed, and adding variety for repeat-business passengers. A much larger set of ETPs is also generated. Each pre-generated ETP provides a path for completing the given tour, based on the initial conditions of remaining unvisited POIs and current AUV position.

We assume that increasing the potential initial conditions covered by pre-generated ETPs also increases the expected tour score (see Section 2.7). We estimate the required number of ETPs by first computing the number of possible tours of one or more POIs. For n POIs, this is:

$$\#\ of\ possible\ POI\ tours = \sum_{k=1}^{n} P(n,k)$$

With five POIs, 325 different tours are possible. With ten POIs, there are almost ten-million different tours. And, with twenty POIs, the number of different tours exceeds 6.6×10^{18}. While the numbers become daunting quickly, for a given operating area many tour plans may be similar and could be collapsed together. Also, in some (perhaps many) cases, the visit order may not matter.

For each possible ETP, we determine the AUV positions for which it would be the optimal choice, leveraging that many positions are very much alike. For example, all positions within some distance of a given point might have the same optimal ETP. Adding *feeder segments* from such *join points* to pre-generated tour legs gives the AUV more options to safely use a pre-generated ETP. See Figure 3-3 for a notional example.

Clearly, the number of pre-generated tour plans required for PR-2 is highly dependent on the operating area's geography and POI placement. There may be commonalities that can be applied generally, e.g., an algorithm for reducing tour permutations, or the number of join points and feeder segments and how to place them, which can lead to manageably sized tour-plan libraries. At worst, the PR-2 approach shows promise for small numbers of POIs, say between five and ten.

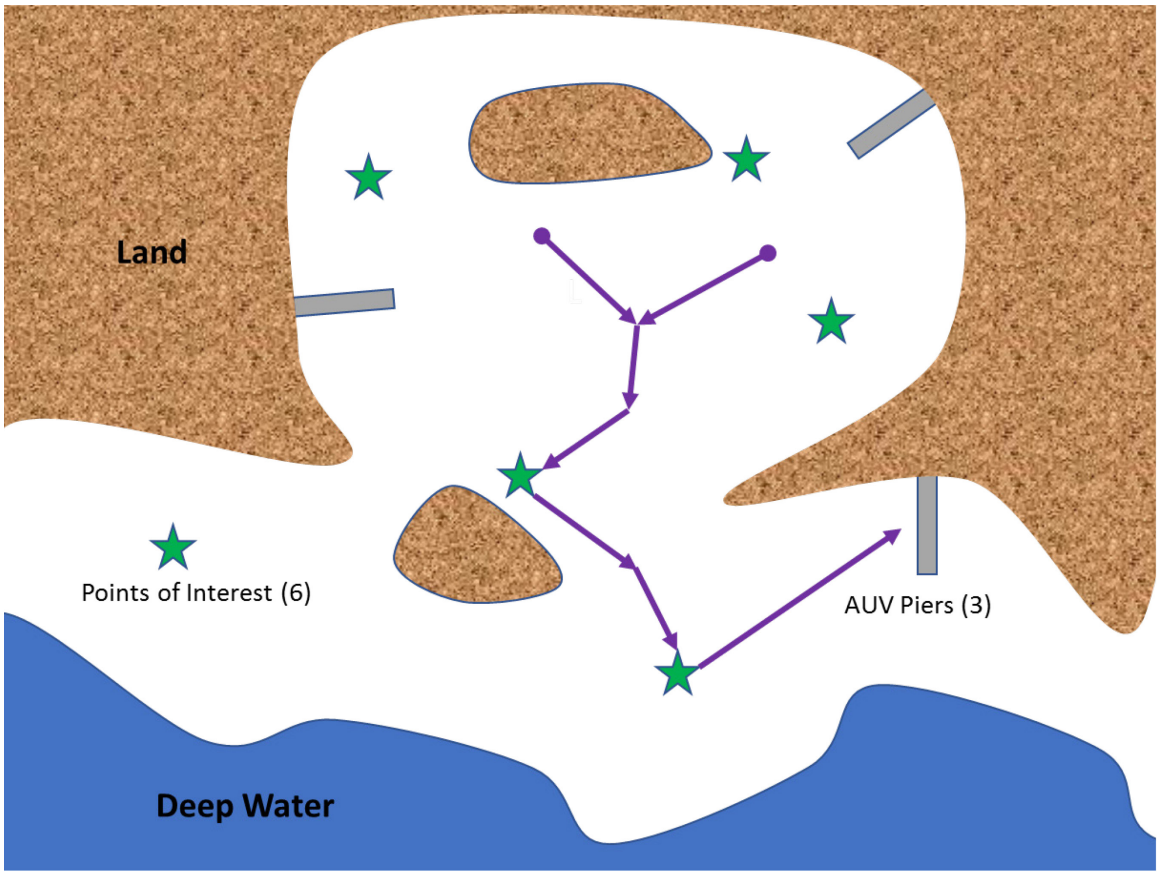


Figure 3-3 Notional enroute tour plan with two join points and their feeder segments

Tour-plan library generation is a trusted function. The plans are assumed to be the best possible, unaltered by malicious entities, and delivered in a trusted manner to the AUV. For example, libraries may be digitally signed by the manufacturer/operator to prove authenticity and the absence of tampering.

3.4.1.3 Protect Function #3 (PR-3): Full Tour-Plan Generation

Protect function #3 (PR-3) incorporates the AE tour-plan generation function into the RM. The challenge comes in providing the required assurance. Because we deferred developing an actual high-assurance RM (see Section 3.6), only simulating one, we opted for implementing PR-3 by reusing our AE's tour-plan generation functions. These functions employ straightforward algorithms like random POI

ordering, shortest-path, or traversing the POIs in numerical order. We are confident the state of the art in high-assurance system development can achieve any of these.

In addition, and related to, the RM assurance challenges, we acknowledge in Section 1.2 that an AE based on artificial intelligence and machine learning (AI/ML) is likely to have superior capabilities to what's possible with current high-assurance software development techniques. AI/ML can add nuances that are not as straightforward to implement in state-/rule-based systems. For example, perhaps based on social-media feedback and natural language processing, an AI/ML system could learn not only preferred POIs, but also the best direction/depth from which to approach them, how long to loiter, etc. In other words, the system could learn everything a human pilot who listens to their passengers would. Such functionality may be possible without AI/ML, but as features are added the complexity likely increases quickly to exceed what can be formally verified, for example, and therefore we don't include it in our RM.

3.4.2 Detect Functions: Tour Monitoring

Detect functions continuously monitor the AUV's state and invoke other RM functions if conditions warrant. The RM monitors separation from the seafloor, fixed obstacles, and other vehicles, using digital charts and HCS. Much like Safeguard [23], the RM does this by comparing the AUV's position and depth to the object boundaries described in Section 2.4. The RM also monitors the achievability of the active tour plan, using PR-1. If separation or tour-plan achievability are at risk, the RM invokes protect, respond or recover functions, as necessary. We implemented both the separation and tour-plan achievability detect functions.

3.4.3 Respond Functions: Warnings and Non-Critical Corrective Actions

Respond functions warn the AE that corrections to its behavior are required soon, or they correct the behavior without permanently overriding the AE like recover functions do. Examples include warning the AE that a collision risk has arisen and, if necessary, performing collision avoidance if it appears the AE is not going to in time. Another example is increasing speed if tour-plan achievability comes into question.

We only implemented warning respond functions because our original boundary-based approach (see Section 2.4) did not provide a clear place for invoking corrective actions without irrevocably taking control from the AE. During experimentation, we determined that adding a response boundary (RB) between the warning boundary (WB) and separation boundary (SB) provided such a place (see Section 4.6). For example, in collision avoidance our current RM invokes a warning respond function when an object's WB is crossed. If we add an RB, and if it is also crossed, an improved RM would invoke a corrective respond function to avoid collision before the SB is crossed, mandating a recover function. When the potential loss of separation (LOS) is resolved, the RM returns control to the AE. Experimenting with RBs and corrective respond functions is part of our future work.

We expect an AE might ascribe corrective respond-function induced changes in position, depth, heading, speed, etc. to variations in currents. A fault-free uncompromised AE should handle these without issue. Of course, ideally such an AE also would not require respond functions. A faulty or compromised AE might, e.g., continue to get into potential LOS situations, or command incorrect speeds. In those cases, the RM continues to respond as long as the situation does not devolve to requiring a recover function.

3.4.4 Recover Functions: Permanent Override of AE

Recover functions are for when the RM no longer trusts the AE, and takes control of the AUV for the remainder of the tour. They are invoked on LOS, if the elapsed tour time exceeds TD-Max, or if it takes too long to generate an achievable ETP, e.g., because the AE refuses to do so. As recover functions are implemented, they are added to the set of existing ones. Having a variety of recovery options allows the RM to maximize safety and mission assurance based on the situation at hand and what other systems have failed. We explore this in Chapter 6.

3.4.4.1 Recover Function #1 (RC-1): Stop and Surface

Recover function #1 (RC-1) is a simple safety override, like those in Safeguard [23]. Though all our development was performed on simulations, we implemented RC-1 to represent a mechanical system for

maximum assurance. On invocation, the RM shuts off propulsion and drops ballast to surface the AUV, coordinating with HCS. This cuts the tour short, failing the mission, but achieves the safety goals.

3.4.4.2 Recover Function #2 & #3 (RC-2 & RC-3): Return to Pier

Recover function #2 (RC-2) is a return-to-pier function similar to Safe2Ditch [44] in ICAROUS [18]. The RM surfaces the AUV in coordination with HCS, and navigates back to the pier from which it departed. The mission may be a partial success, if any POIs were visited.

Recover function #3 (RC-3) has the RM select the "best" pier for the AUV. This can be based on a number of factors, including pier proximity, separation concerns enroute, how much time is left before reaching TD-Max, and the attractiveness of returning to the departure pier, e.g., it's where the passengers' cars are. The mission assurance of this recover function is somewhat better than RC-2, because it may reduce collision risks by returning the AUV to a closer pier through open water. We implemented RC-2, but not RC-3, because we considered the difference minimal.

3.4.4.3 Recover Function #4 & #5 (RC-4 & RC-5): Complete Pre-Generated Tour Plan

Recover function #4 (RC-4) improves mission assurance by having the RM take over and continue the DTP tour. RC-4 requires PR-2, but only uses the DTPs in the tour-plan library. Instead of the AE generating a DTP for RM approval, the AE selects a pre-generated DTP from those offered by the RM. If the AE exhibits safety or mission-threatening behavior during the tour, the RM takes over. RC-4 only works as long as the DTP is achievable. If achievability is lost, the RM must use one of the tour-ending recover functions above.

Recover function #5 (RC-5) improves on RC-4 by using the full PR-2 tour-plan library, allowing the RM to continue a tour after the DTP, or a subsequent ETP, becomes unachievable. In operating areas with many POIs, the tour-plan libraries may not be comprehensive, so RC-5 may be limited in how it completes a tour after achievability has been lost.

We did not implement RC-4 or RC-5, because of their dependence on PR-2, which we also didn't implement for the reasons given in Section 3.4.1.2.

3.4.4.4 Recover Function #6 (RC-6): Full Tour-Plan Execution

Recover iteration #6 (RC-6) requires PR-3 and, using it, fully incorporates the AE's tour planning and execution functions in the RM. We implemented RC-6 for the same reasons as PR-3 (see Section 3.4.1.3). Like PR-3, the challenge to implementing RC-6 in the RM comes from needing to provide the required assurance, as well as losing any presumably superior capabilities of an AI/ML-based AE.

3.5 Discussion

In combination with the iterative mission-assurance refinement analysis (IMARA) methodology described in Chapter 5, and applied to our AUV scenario in Chapter 6, we were able to develop objectives for acceptable, if not always optimal, missions. We then captured the IMARA-informed functionality needs of those mission objectives (MOs) in the design of the RM's IPDRR functions. None of the functions were extraordinarily complex, instilling confidence that they could be implemented in a high-assurance RM. Moreover, given the similarities in the MOs of many different autonomous-vehicle (AV) missions (see Section 6.7), only a handful of generalized RMs might be required to provide assurance-improvement options for all of them.

As ACPS in general, and AVs in particular, are integrated into day-to-day life and interact with people more frequently, we imagine an RM industry arising, much like the many safety/security industries of today, e.g., the aforementioned aircraft black boxes, fire detection and prevention systems, burglar alarms, and a host of others. Standards could be developed such that any manufacturer's RM would work in a compliant AV, much like the trusted platform module (TPM) standard [36] enabled motherboard manufacturers to provide a common socket that supports any compliant TPM [71]. Standardization further led to the incorporation of TPM technology in Intel [51] and AMD [5] CPUs since 2013, and it being required for Windows 11 [47]. A TPM trustworthily implements a small set of generic functions that secure a wide variety of applications. Generalized RM functionality, even if it is specific to a domain or type of

mission (an airliner's black box is different than a cruise ship's), would provide manufacturers a straightforward and standard way to add mission assurance.

3.6 Known Limitations

Without at least the protect functionality of PR-2, an AE can generate and have approved valid, but extremely poor quality, ETPs. For example, because PR-1 only validates that an ETP's ETE falls within the tour-duration limits, the AUV could meander all over the operating area and return to the pier without visiting any more POIs, as long as it did so before exceeding TD-Max. The only comfort is that such sub-optimal planning is reflected in tour scoring.

We didn't develop the high-assurance RM software that we deem necessary for AV operations with or near people, but plan to in future work. Instead, we focused on maximizing our testbed's capabilities to measure assurance. This yielded many valuable results in instrumenting collision-avoidance algorithms, for example, and collecting detailed metrics on AUV behavior (see Chapter 4). We constrained our RM software's functionality to what's possible with formal verification (FV) [24]. Any flaws found during experimentation were fixed and simulations repeated. This did not guarantee the software fault-free or correct for future unexpected situations, but gave our results the same credibility as if it was. In addition, just as we would on a real AUV, we had the (simulated) mechanical RC-1 as the ultimate failsafe, in case FV didn't catch everything, or a situation arises that our design didn't anticipate. Focusing on the limits of high-assurance software informed how much mission assurance such an RM could provide, and what was beyond it. See the notional example in Section 3.4.1.3.

3.7 Related Work

Developing high-quality complex software continues to be a significant challenge and much is written about the problem in academia and the technical press [8] [9] [31] [46] [55] [70]. To address the problem a different way, the concept of incorporating simpler more reliable safety systems has arisen over the years and shows promise. However, as discussed below, current safety systems are limited to protecting

the public, operator, or vehicle. Some [66] [75] are implemented with limited or no assurance, particularly against malicious actors. Very few discuss protecting the mission that the vehicle needs to perform.

Xiao et al. [75] developed a rule-based safety kernel for unmanned systems. While a separate processor is used for controlling the vehicle's motors and reading its sensors, the safety kernel is not implemented on this processor. Instead, it is a user-level process implemented on a "main control processor" that contains all the other application software and connects via Wi-Fi to an external personal computer (PC) to process image data for navigation. Malware introduced into this main control processor presumably could circumvent the safety functions, making this approach unacceptable for the safety of human life.

Stevens and Atkins [66] propose a geofence system independent of the autopilot for redundancy. They, however, make no mention of the assurance requirements for such a system. Therefore, one can assume that it is as vulnerable to malware as any other part of the vehicle.

Safeguard [23] is a totally independent onboard unmanned aerial system (UAS) geofencing system, including independent sensors, with no inputs from the vehicle and only two discrete outputs – a warning that a geofence violation is imminent, and a kill signal if the warning does not result in corrective action. Safeguard only protects others against damage by the vehicle, not damage to the vehicle. Its only recourse for a UAS about to violate the geofence is to cut power to the motors. It does not assure the vehicle's safety, which is understandable because Safeguard was built under the assumption that hull loss is acceptable for small inexpensive UASs. The AUVs in this dissertation cannot assume that, because there are passengers on board. Also, Safeguard gets around the need to protect bystanders by forbidding any in the vehicle's operating area. An AUV operating amongst other vehicles with crews or passengers cannot assume that, either.

ICAROUS [18] has improved mission-assurance functionality, complementing Safeguard's geofencing by adding NASA's DAIDALUS [48] detect and avoid capabilities against fixed obstacles and other vehicles

that are communicating with the UAS (or an ATC-like system that the UAS is monitoring), as well as the ability to determine a conflict-free "return to mission" path when the obstacle has been avoided. ICAROUS is being integrated with Safe2Ditch [44] [56], a computer-vision-based landing site selection system being developed at NASA Langley, which should reduce risk if the UAS decides it needs to land. If Safe2Ditch is invoked, the mission is over. Moreover, in the event of an impending geofence violation, Safeguard still takes the drastic action described above. Like Safeguard, ICAROUS mitigates the risks to bystanders by forbidding any in the UAS's operating area. And, because ICAROUS's vehicle avoidance is limited to other aircraft announcing their position, velocity, heading, etc., it is insufficient in an environment where all participants are not necessarily communicating with each other, or known. In the end ICAROUS is a safety system. It does not take over the mission if other processors onboard become erratic or unresponsive.

3.8 Conclusions

This chapter introduced a general RM concept for ACPSs and described specialized functionality we designed leveraging our IMARA methodology (see Chapter 5) to improve the safety and mission assurance of our MAAUV. We developed an RM with that functionality for the MAAUV-Lite model from Section 2.5 and experimented with it in a number of test environments. The details of that experimentation are presented in Chapter 4. We see this IMARA-based design process as a template for designing mission-specific RMs, and possibly for maturing RM design and development to produce a small set of general-purpose RMs for a variety of similar missions.

3.9 Future Work

Having gained experience with a simulated high-assurance RM, we would like to develop a real one. Though our work is entirely in simulated environments at present, we believe the same high-assurance software for a physical RM would work unchanged in Gazebo [73]. Maturing an RM this way would reinforce our cost-effectiveness arguments for simulated environments presented in the next chapter.

With the addition of the RB based on our experimentation in Chapter 4, we now have a place to invoke corrective respond functions, and would like to develop and experiment on some for the MAAUV mission.

The need to investigate the impacts of sensor and HCS accuracy/availability losses discussed in Section 4.9, and determine how to remediate or mitigate them to maintain safety and mission assurance, applies directly to the RM. What's required for trusted HCS communications must also be investigated. There are robust cryptographic approaches for authenticating communicants and assuring message integrity, for example, that would not necessarily remediate all attacks, but would make them detectable, and the impact of detectable HCS failures is explored in Chapter 6.

4 COST-EFFECTIVE MISSION ASSURANCE ENGINEERING

4.1 Introduction

Safety and mission assurance in an autonomous vehicle (AV) must address multiple areas at the component, system, and system of systems levels. Our approach is two-fold: a methodology for iterative mission-assurance refinement analysis (IMARA) (see Chapter 5), and a means to engineer, test, and experiment on AVs without prohibitive hardware costs, or even the need to build an actual AV until algorithm designs are more solid, i.e., simulation.

We tackle the challenges of developing an autonomous underwater vehicle (AUV) to operate in an environment that is only partially known, maximizing the number of potential unknowns addressed while keeping the development effort cost-effective. For example, we can envision numerous possible collision scenarios, but history has shown that every possibility is rarely anticipated. The best way to increase the required knowledge is by accruing operating experience, assuming that unexpected situations will be encountered and factored into maturing the AUV. However, this is an expensive approach with actual vehicles, and the risk of injury or property damage comes with it.

In this chapter, we employ a simulated mission-assured AUV (MAAUV) equipped with a resilience module (RM) from Chapter 3 to perform the mission described in Chapter 2, i.e., transport passengers amongst points of interest (POIs), avoid loss of separation (LOS) from other objects, adhere to a maximum-allowed tour duration (TD-Max), and return the passengers safely to the pier at the end. We call these missions “tours.” Using our Digital Environment for Simulated Cyber Resilience Engineering, Test and Experimentation (DESCRETE) testbed, described in Section 2.3, we tested a MAAUV collision-avoidance algorithm by varying certain parameters and measuring the number and characteristics of “recovered” test tours, i.e., those that suffered LOS.

Our contributions are as follows: (1) we introduce an open-source simulation testbed for cost-effective mission-assurance engineering without the need for high-fidelity modeling up front; (2) we

experiment on and analyze our MAAUV use case with the testbed; and, (3) we propose refinements to the RM's role in mission-assurance, based on our experimental results.

The remainder of the chapter is organized as follows. Section 4.2 describes the functionality of the test AUVs. Section 4.3 describes our experimental setup and procedure. Section 4.4 and Section 4.5 detail the results of our first and second sets of experiments, respectively, and Section 4.6 discusses our observations. Section 4.7 reviews related work in the literature. Section 4.8 presents our conclusions, and Section 4.9 describes future research.

4.2 AUV Functionality

Each test AUV is 2.45 m long, with a turn radius (TR) of 6 m at its maximum speed of 1.5 m/s (3 knots). All the AUVs contain an autonomy engine (AE) to control touring and collision avoidance. Based on our arguments in Section 1.2, we assume the AE is overly vulnerable to software flaws and cyber-attack, and should not be trusted alone to protect human passengers. Therefore, the MAAUV also contains a trusted RM in case the AE's behavior puts safety or mission at risk. The RM monitors all sensors and moderates all commands to actuators, overriding the AE if necessary with recover functions. The RM and its recover functions are described in detail in Chapter 3.

In each experiment below, if the RM had to act, it first completed any collision avoidance, and then invoked one of the following recover functions:

- **RC-1**: Stop and surface immediately
- **RC-2**: Abandon the tour and return to the pier
- **RC-6**: Continue the tour

Our collision-avoidance solution employs the following boundaries, described in detail in Section 2.4:

- **Warning Boundary** (WB): Crossing this causes the RM to issue a warning to the AE.
- **Separation Boundary** (SB): Crossing this is LOS, RM invokes the stop/surface function.
- **Physical Boundary** (PB): Crossing this is a collision.

In all the experiments, the AEs were configured to take immediate collision-avoidance action on a WB crossing. For the MAAUV, if LOS occurred, the RM took over and invoked the recover function chosen for the given experiment.

In addition to handling LOS, the RM attempts to keep the tour duration below TD-Max. If the tour's estimated time enroute (ETE) exceeds TD-Max, the AE performs tour-plan regeneration, which removes unvisited POIs from the plan until the ETE is no more than TD-Max. If there are no more POIs to remove, the AE generates a plan to return to the pier. If the elapsed tour duration exceeds TD-Max, which did not occur in any of the experiments, the RM also invokes a recover function.

4.2.1 The Collision-Avoidance Algorithm

Our current collision-avoidance algorithm, common to all the AUVs, is a simple one that uses the maritime collision regulations (COLREGS) [49], and is invoked when an AUV crosses another object's WB. Using the relative bearings, headings, and speeds of the closest conflict and itself, 100 times per second each AUV executes the expected avoidance maneuver based on the COLREGS. All AUVs implicitly assume the others also perform the maneuvers expected of them. If multiple objects are within the WB, an AUV maneuvers to avoid the closest one. This, of course, could steer it directly toward the next closest object. When that object becomes the closer one, however, the AUV maneuvers to avoid it.

Though far from optimal, our algorithm serves to improve the testbed by raising collision risks frequently enough for us to evaluate data collection and analysis techniques improvements without waiting for a rare LOS in a "good" algorithm. As the testbed matures, our focus will shift to exercising and analyzing better conflict resolution algorithms that, e.g., factor in other AUVs before they enter the WB, and account for multiple conflicting AUVs rather than just the closest one. Existing algorithms will be sampled from the literature, as well as novel approaches explored.

4.2.2 Balancing Safety and Mission

In the real world, risks are often taken and some safety is traded to improve the chances of getting the mission accomplished. Depending on the mission's importance, safety may be sacrificed to the point where vehicle damage, or human injury/death can occur, e.g., military operations.

For the MAAUV, the overlaps between safety and mission are LOS and returning passengers to a pier. There is no wiggle room in the latter; either the passengers are returned to a pier or they are not. For LOS, we can make safety/mission tradeoffs by adjusting the boundaries described above. These boundaries have minimums based on AUV performance, i.e., speed, turn radius, stopping distance, dive rate, etc. They do not have safety-based maximums, however. Those are driven by mission requirements.

For example, we presumed going in that increasing the WB size adds safety to tours. However, this would also reduce the number of AUVs that can operate together effectively, which reduces the number of tours per day (or the number of tour companies allowed to operate in the area), directly impacting the tour companies' bottom lines. As seen below, increasing WB size also increased some tour durations to the point that POIs had to be skipped.

4.3 Experimental Evaluation

Our experimentation explored collision-avoidance effectiveness as a function of WB size, as well as the impact on tour duration as WB size increased and AUVs were more likely to require avoiding each other. We varied all AUV WB and SB sizes as functions of multiples of TR, which makes the results more generalizable than if absolute distances, e.g., meters, were used.

4.3.1 Experimental Setup

Experiments were conducted in DESCRETE with a MAAUV-Lite vehicle and three partner AUVs (PS#2 through PS#4), defined in Section 2.2. The MAAUV-Lite model, named for its simplified construction, is shown in Figure 2-6. The other AUV models are the same, except their hulls are red in color. An example of a MAAUV-Lite and partner submersibles interacting in DESCRETE's computer-generated imagery (CGI)

is shown in Figure 2-2. To convey our results better, they are presented using charts like Figure 4-1. This chart does not depict the image in Figure 2-2.

The chart in Figure 4-1 is read as follows:

- Green circles represent POIs.
- Red Xs represent partner submersibles, and the blue cross represents the MAAUV.
- Blue lines represent the tour path each AUV followed (see below).

We generated 100 test environments, each with a flat seafloor 80 m deep, calm water, no currents, and ten POIs laid out pseudo-randomly in a 200 m by 200 m test range centered at (0,0). Each POI was represented by a 10 m wide hemisphere centered on the seafloor, i.e., the POI extended 5 m from the seabed. POI boundaries were held constant across all test tours, with PB at 5.0 m, SB at 12.0 m, and WB at 30.0 m. The charts for all the layouts are provided in Appendix B.

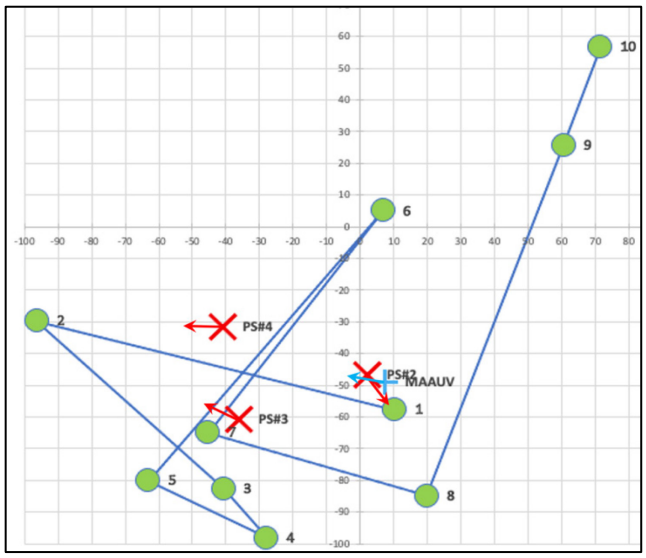


Figure 4-1 A simplified view of a test tour in progress in the DESCRETE testbed.

Each AUV had a dedicated floating pier at (±155,±155), outside Figure 4-1. The MAAUV pier was at (155,155). Each partner's pier was in the quadrant corresponding to the AUV's number, e.g., PS#2's pier

was at (-155,155). The AUVs all shared position, heading, and speed information via the Harbor Control Services (HCS) described in Section 2.2, allowing each to know where it and all the other partners were.

4.3.2 Experimental Procedure

The following procedure was performed twice, first with an initial set of boundary values, Test001, and then with updated values, Test002, based on the Test001 results:

1. Perform all 100 test tours five times with RC-1, varying WB and SB sizes based on 1xTR to 5xTR
2. Repeat any tours that had LOS, first with RC-2, and then with RC-6

For each test tour, the MAAUV was given a TD-Max of one hour to tour a path of POIs in order from POI #1 to POI #10. Visiting a POI required an AUV to get within 40 m of it. Touring the POIs in order prevented random or optimized tour generation from affecting the results, and also increased opportunities for conflicts between AUVs. The average tour ETE with no conflicts was 1,262 s, making an hour almost three times the ideal. It was hypothesized that, even with conflicts, time would not be a factor. The other AUVs used the same POI paths, but had no time limit.

4.4 Test001 Results

All 100 test tours were first run five times with RC-1 as the recover function, varying the AUV WB size from 1xTR to 5xTR (Test001a to Test001e, respectively), and AUV SB size was set to 0.4xWB to provide 60% of the separation on initial conflict as “maneuvering room.” These values are shown in the first three rows of Table 4-1.

Test001 used a PB size of 1.225 m, half the length of the MAAUV-Lite. It was later discovered that the PB size was determined incorrectly. Fortunately, this did not affect the results, because no collisions occurred. This is discussed in Section 4.5.

Table 4-1 Test001 MAAUV-Lite Results (PB at 1.225 m)

Test:	001a (RC1)	001a (RC2)	001a (RC6)	001b	001c	001d	001e
Test Variable Value (AUV WB in #TRs):	1	1	1	2	3	4	5
AUV WB (m):	6.00	6.00	6.00	12.00	18.00	24.00	30.00
AUV SB = 0.4xWB (m):	2.40	2.40	2.40	4.80	7.20	9.60	12.00
# Recovered Tours (of 100):	6	6	6	0	0	0	0
# Tour Plan Regenerations:	0	0	0	0	0	0	1
Average Estimated Tour Duration (s):	1262	1262	1262	1262	1262	1262	1262
Average Actual Tour Duration (s):	858	862	874	940	1092	1269	1534
Time en route to POI (%):	80.6%	80.3%	80.5%	77.0%	71.5%	66.1%	61.5%
Time en route to pier (%):	17.0%	17.9%	17.7%	16.5%	14.2%	12.2%	10.2%
Time avoiding conflict (%):	1.7%	1.8%	1.8%	6.5%	14.3%	21.6%	28.2%
Time recovering (%):	0.6%	1.0%	2.5%	0.0%	0.0%	0.0%	0.0%
Closest Approach to another AUV (m):	1.99	1.83	1.83	4.91	9.33	13.15	14.70
Closest Approach to AUV (%WB):	33.2%	30.5%	30.5%	40.9%	51.8%	54.8%	49.0%
Closest Approach to AUV (%SB):	82.9%	76.3%	76.3%	102.3%	129.6%	137.0%	122.5%

The Test001 results are summarized in Table 4-1 and show the vast majority of the test tours ended successfully. Six tours were recovered, all with a WB size of 1xTR, and are discussed further in Section 4.4.1. There was one tour-plan regeneration, triggered by the tour ETE exceeding TD-Max. The tour-plan regeneration occurred on one of the same tours that was recovered with a lesser WB size, i.e., tour #69 (see Section 4.4.2.4).

The six recovered Test001 tours were run again with RC-2 and RC-6, with additional details provided in Table 4-2 and Table 4-3, respectively.

Table 4-2 Test001a RC-2 Recovered-Tour Details

Tour #	#POIs Visited	Min. AUV Separation (m)	Score (of 100)
54	8	2.18	72
79	8	2.08	69
93	9	1.83	68
18	5	2.40 (just barely LOS)	49
40	4	2.36	39
69	1	2.34	9

Table 4-3 Test001a RC-6 Recovered-Tour Details

Tour #	#POIs Visited	Min. AUV Separation (m)	Score (of 100)
18	10	2.40 (just barely LOS)	99
40	10	2.36	98
69	10	2.34	97
54	10	2.18	90
79	10	2.08	86
93	10	1.83	76

Using the scoring equations defined in Section 2.7, the six recovered tours all scored zero with RC-1, because the MAAUV didn't return to the pier. With RC-2, these tours all scored higher (see Table 4-2), but the scores were negatively impacted by the unvisited POIs, as well as the LOS. With RC-6, the scores improved further (see Table 4-3), because with all the POIs visited, only LOS reduced them.

Table 4-1 shows the minimum separation for Test001a with RC-2 and RC-6 was reduced to 1.83 m from 1.99 m with RC-1. This occurred on Tour #93 because the AUVs were still closing on each other when LOS occurred. With RC-2/RC-6 the MAAUV continued its collision-avoidance turn, rather than stopping as it did with RC-1, and closed more before the AUVs started separating. The minimum-separation values are the same for RC-2 and RC-6, because the RM resolved the LOS-inducing conflicts the same way in both cases. By pure chance, no additional LOS occurred after the initial instance in each tour. There were additional conflicts after the LOS in a few tours, but none led to another LOS.

4.4.1 Recovered Tours

The only recovered tours were with a WB size of 1xTR. Figure 4-2 and Figure 4-3, cropped from charts in Appendix B, show where recovery-causing conflicts started for each of the six recovered tours, and where LOS occurred.

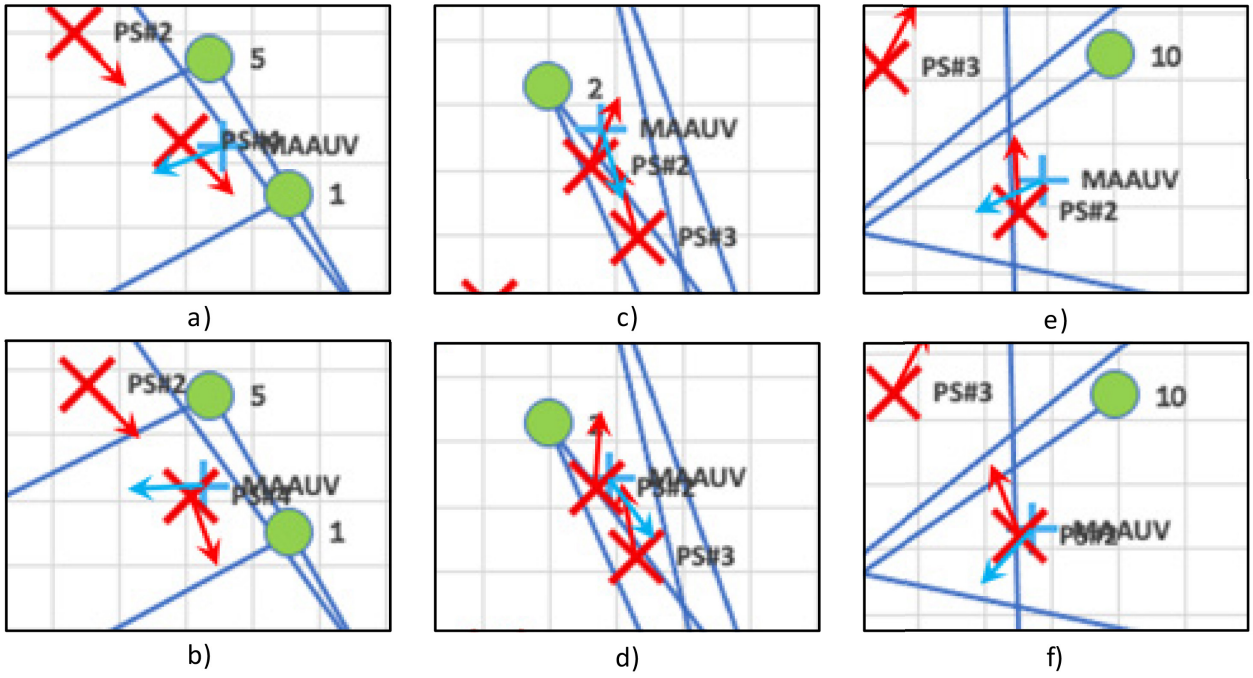


Figure 4-2 a) Start of conflict for tour #18, b) LOS for tour #18; c) Start of conflict for tour #40, d) LOS for tour #40; e) Start of conflict for tour #54, f) LOS for tour #54.

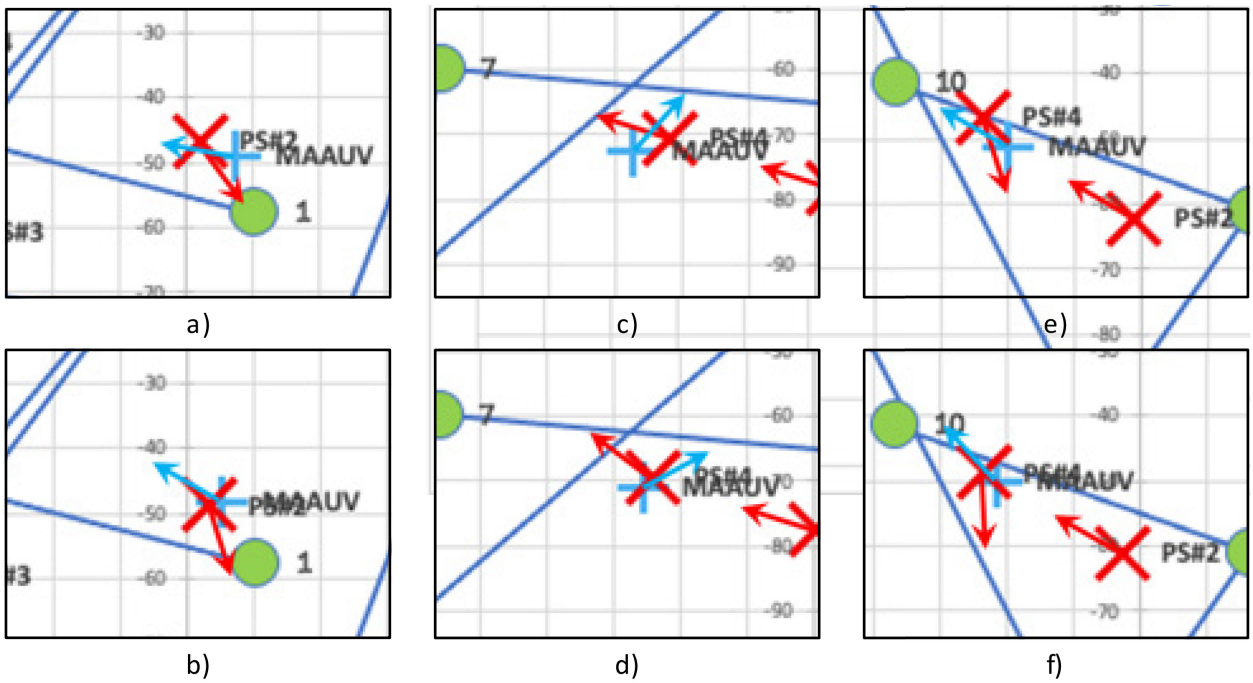


Figure 4-3 a) Start of conflict for tour #69, b) LOS for tour #69; c) Start of conflict for tour #79, d) LOS for tour #79; e) Start of conflict for tour #93, f) LOS for tour #93.

All six tours were recovered on the first or second conflict, suggesting that a 1xTR WB size is too small. In an ideal environment like our tests, with no system faults, the expectation should be no recovered tours, because faults or malicious attacks would only make the likelihood of needing recovery greater.

As Figure 4-2 and Figure 4-3 suggest (and the raw data confirm), there were no collisions. The closest another AUV got to the MAAUV was 1.99 m, or about a boat length. The closest that two partner AUVs got to each other was 1.79 m, due to their lack of recovery mechanisms. Had the RM not invoked recovery, the MAAUV LOS cases would also not have caused collisions. Still, the distances would be extremely unnerving to passengers, at the very least.

4.4.2 Conflicts without LOS

Even without LOS, AUVs often got closer to each other than expected, especially given the associated WB sizes.

4.4.2.1 WB at 12 m, SB at 4.8 m

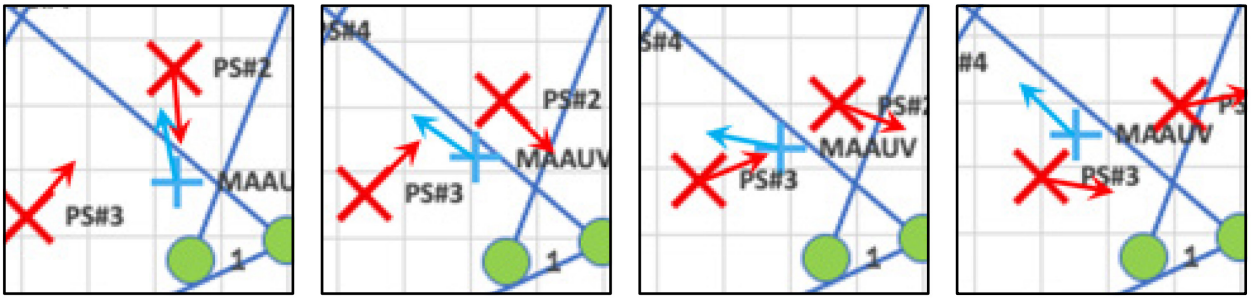
At a WB size of 2xTR, two partner AUVs closed to 4.22 m of each other, which caused no reaction beyond normal avoidance maneuvers, because of their lack of RMs. This was part of a four-way conflict between all AUVs where, by sheer luck, nothing got closer than 7.92 m to the MAAUV, so recovery was not triggered. A different conflict brought another AUV to 4.91 m of the MAAUV, barely avoiding recovery.

4.4.2.2 WB at 18 m, SB at 7.2 m

At a WB size of 3xTR, tour #7's separation fell to 9.33 m for the MAAUV. The specifics of this conflict represent a case seen many times in the test results – one that the data make clear must be factored into the design of the safety, avoidance, and touring algorithms. Figure 4-4 captures the conflict unfolding.

The conflict began with the MAAUV and PS#2 approaching each other head on (Figure 4-4(a)), arguably the worst case given the AUVs' relative speeds and how quickly separation could be lost. Each AUV turned to port, as expected for the avoidance algorithm. During these maneuvers, PS#3 and the MAAUV crossed each other's WBs (Figure 4-4(b)). PS#3 began avoidance maneuvers, turning to starboard,

again as expected. Because the avoidance algorithm only reacts to the closest conflict, the MAAUV continued to avoid PS#2, maintaining its turn to port, and towards PS#3!



a) b) c) d)

Figure 4-4 A multi-AUV conflict: a) MAAUV and PS#2 start to avoid each other; b) PS#3 starts to avoid MAAUV, but MAAUV is still avoiding the closer PS#2, turning towards PS#3; c) MAAUV starts to avoid the now closer PS#3; d) MAAUV and PS#3 at minimal separation, but not LOS.

Eventually, PS#3 became the closer conflict to the MAAUV (Figure 4-4(c)), which then reversed its turn and started turning to starboard to avoid PS#3. The minimal separation of 9.33 m occurred during this part of the avoidance maneuver (Figure 4-4(d)). After that, the AUVs all eventually resumed their tours, with one more conflict between the MAAUV and PS#2.

This example, and many others in the data, show that a lot can happen within the WB when multiple AUVs are in play.

4.4.2.3 WB at 24 m, SB at 9.6 m

At a WB size of 4xTR, the closest conflict was 13.15 m, i.e., barely half the WB size, and leaving less than a 40% margin on the SB. The situation was similar to the previous case where another AUV entered the scene and was not avoided until it drew closer than the first conflict. This reinforces the need to research a larger WB/SB-size ratio.

4.4.2.4 WB at 30 m, SB at 12 m

At a WB size of 5xTR, the closest conflict was 14.7 m. Again, this was due to a multi-AUV conflict, and is very close to the SB – closer than the 3xTR and 4xTR minimum separation distances as a percentage of SB (see Table 4-1). Given that avoidance started when the initial conflicting AUVs were 30 m apart, this begins to suggest that merely increasing the WB size or changing the WB/SB-size ratio are insufficient to guarantee separation.

At this WB size, there was also a tour-plan regeneration. Even with the close calls mentioned above, the MAAUV still maneuvered long enough to push the tour ETE past TD-Max, which had been generously set at three times the no-conflict ETE. Addressing the excessive maneuvering problem diametrically opposes the idea of increasing the WB/SB-size ratio. Assuming a minimal SB size of 2xTR, a WB size of 10xSB that might allow plenty of space to react to conflicts would be 120 m, which is four times the WB size that led to TD-Max overruns.

4.5 Test002 Updates and Results

Analysis of the Test001 results uncovered an error in how PB size was determined. DESCRETE (and Gazebo [73]) represent a vehicle's position as a point in space. For the AUVs in our experiments, that point is roughly at the geometric center of the vehicle. As mentioned earlier, the AUVs' longest dimension is 2.45 m, which led to Test001 using a PB size of half that, or 1.225 m, to indicate that anything at that distance or closer has collided with the vehicle. However, this did not factor in that the point representing the colliding vehicle is at *that vehicle's center*. Therefore, two identical vehicles that DESCRETE indicates are 2.45 m apart could actually have collided, e.g., head-on bow-to-bow.

Reviewing the raw data, the AUV positions and orientations in the Test001a tours that suffered LOS, along with their cylindrical shapes, avoided actual collisions, even though the SB size was less than the correct PB size. Had the AUVs been more spherical, like the U-Boat Worx NEMO [50] in Figure 2-7, collisions would have been much more likely to occur. The error was corrected in Test002, with PB size increased to 2.45 m.

Analysis of the Test001 results also suggested improvements for determining WB size. To the TR multiples, Test002 adds the PB size when determining WB and SB size, as shown in the first three rows of Table 4-4. This way, when a WB is crossed, the outer-most parts of the vehicles, rather than their centers, are at least the corresponding TR multiple apart.

All 100 tours were run again five times with RC-1, varying the WB size as described above.

Table 4-4 Test002 MAAUV-Lite Results (PB at 2.45 m)

Test:	002a (RC1)	002a (RC2)	002a (RC6)	002b	002c (RC1)	002c (RC2)	002c (RC6)	002d	002e
n = Test Variable Value:	1	1	1	2	3	3	3	4	5
AUV WB = n*TR + PB (m):	8.45	8.45	8.45	14.45	20.45	20.45	20.45	26.45	32.45
AUV SB = 0.4*n*TR + PB (m):	4.85	4.85	4.85	7.25	9.65	9.65	9.65	12.05	14.45
# Recovered Tours (of 100):	22	22	22	0	1	1	1	0	0
# Tour Plan Regenerations:	0	0	0	0	0	0	0	0	8
Average Estimated Tour Duration (s):	1262	1262	1262	1262	1262	1262	1262	1262	1262
Average Actual Tour Duration (s):	780	800	894	998	1135	1136	1142	1369	1767
Time en route to POI (%):	79.1%	77.2%	79.3%	74.5%	69.5%	69.5%	69.5%	64.3%	58.0%
Time en route to pier (%):	15.6%	19.7%	17.3%	15.5%	13.5%	13.6%	13.6%	11.4%	8.8%
Time avoiding conflict (%):	2.8%	3.1%	3.4%	9.9%	16.9%	16.9%	16.9%	24.3%	33.2%
Time recovering (%):	2.5%	4.9%	14.9%	0.0%	0.1%	0.1%	0.7%	0.0%	0.0%
Closest Approach to another AUV (m):	3.17	3.41	2.97	7.61	8.19	8.02	8.02	14.57	17.53
Closest Approach to AUV (%WB):	37.5%	40.4%	35.1%	52.7%	40.0%	39.2%	39.2%	55.1%	54.0%
Closest Approach to AUV (%SB):	65.4%	70.3%	61.2%	105.0%	84.9%	83.1%	83.1%	120.9%	121.3%

The increases in WB sizes for corresponding TR multiples from Table 4-1 to Table 4-4 might suggest that Test002 tours would have fewer conflicts lead to LOS. The Test002a WB size is more than 40% larger than its Test001a counterpart, but instead of the six recovered tours in Test001a, Test002a had 22. Moreover, only two of the six recovered tours from Test001a were also recovered in Test002a. The others were new. Test002c also had a recovered tour, even though Test001c had none and the former's WB size was more than 13% larger. These results suggest that the WB-size increase is not the only factor, which is discussed further in Section 4.6.

All the Test002 recovered tours were run again with RC-2 and RC-6, with increases in scores like those seen in the Test001 results. The Test002a scoring-related RC-2 details are provided in Table 4-5.

Unlike the Test001a results, in at least one tour the minimum separation was more with RC-2 than with RC-1. This highlights a dangerous collision-avoidance situation where an AUV is expecting to pass behind the moving MAAUV, and the MAAUV suffers LOS and stops. This could cause a collision, rather than prevent one, and further reinforces that RC-1 is an insufficient mission assurance capability.

Table 4-5 Test002a RC-2 Recovered-Tour Details

Tour #	#POIs Visited	Min. AUV Separation (m)	Score (of 100)
52	9	4.09	75
100	8	3.93	64
79	8	3.49	57
67	7	3.90	56
88	7	3.43	49
75	5	4.15	42
5	4	4.47	36
83	4	4.15	34
36	3	3.52	21
98	2	3.41	14
81	1	4.50	9
15	1	4.72	9
10	1	4.75	9
47	1	3.88	8
32	1	4.03	8
63	1	4.32	8
69	1	3.67	7
68	0	3.75	0
66	0	3.79	0
49	0	4.38	0
76	0	4.51	0
28	0	4.63	0

The Test002a RC-6 results are summarized Table 4-6. Many of the minimum separation values remained the same as with the RC-2 results. In three tours, however, additional LOS occurred after the one that caused the RM to take over. This is not surprising, because the RM uses the same collision-avoidance algorithm as the AE. In two of the multi-LOS tours, the later occurrences had less minimum separation, with one for Tour #47 coming within almost 0.5 m of potential collision.

The Test002c RC-2 and RC-6 results are summarized in Table 4-7. The one recovered tour had similar results to those in Test001, i.e., the MAAUV continued the turn and closed a little more, 8.02 m versus 8.19 m, before the AUVs started separating.

Table 4-6 Test002a RC-6 Recovered-Tour Details

Tour #	#POIs Visited	Min. AUV Separation(s) (m)	Score (of 100)
10	10	4.75	97
28	10	4.63	95
5	10	4.47	92
81	10	4.50	92
76	10	4.51	92
49	10	4.38	90
63	10	4.32	89
75	10	4.15	85
83	10	4.15	85
52	10	4.09	84
32	10	4.03	83
100	10	3.93	81
67	10	3.90	80
15	10	4.72, 3.99	80
66	10	3.79	78
68	10	3.75	77
69	10	3.67	75
79	10	3.49	71
98	10	3.41	70
88	10	3.43	70
36	10	3.52, 3.55, 4.80	52
47	10	3.88, 2.97, 4.84	48

Table 4-7 Test002c Recovered-Tour Details

Tour #15	#POIs Visited	Min. AUV Separation (m)	Score (of 100)
RC-2	1	8.02	8
RC-6	10	8.02	83

4.6 Discussion

Our results are consistent throughout the test tours and match expectations, mainly that RC-1 is useless for mission assurance. The other two recover functions, RC-2 and RC-6, improve tour quality by returning the passengers to the pier and adding POI visits, respectively. However, our current boundary-based approach is insufficient to invoke these functions. Adding a response boundary (RB) between the SB and WB would provide a trigger for the RM to take over before tour quality is affected. An RB would also give the AE a chance to respond to an RM warning when the WB is crossed. Until the RB is crossed, the AE is in charge. Ideally, it should be assessing avoidance on its own, reacting to potential conflicts long before they occur, e.g., by avoiding crowded areas and going to other POIs until the crowd breaks up. Even when inside a WB, the AE might maintain present course and depth because it expects the potential conflict to resolve itself and no reaction is the best one. This is explored further with an additional HCS capability in Chapter 6.

On the bright side, our results suggest that LOS-inducing conflicts occur rarely. Using a mediocre collision-avoidance algorithm while touring crowded POI layouts with small vehicle boundaries led to only six recovered tours in Test001. Test002 results show slightly different boundaries leading to LOS almost four-times more often using the same POI layouts. However, the bulk of the LOS occurred while using extremely tight boundaries. Ignoring the Test001a and Test002a results, though we saw many close calls in the other test tours and large amounts of WB margin lost before a collision was finally avoided, only one LOS occurred in 200 tours, or 0.5%. This makes one wonder how many accidents don't happen just because luck prevented them.

The significantly increased number of recovered tours in Test002a over Test001a suggests factors beyond WB size. Along with the predicted algorithmic deficiencies in resolving multi-vehicle conflicts, highlighted with the example in Section 4.4.2.2, adding PB size to both WB size and SB size changed their ratio. For example, in the Test002a tours, SB size was over 57% of WB size, while in the Test001a tours, it

was only 40%, and both boundary spheres were smaller. The reduced maneuvering room within the Test002a WB made it more likely for an AUV to cross the SB. More work is required regarding how to determine boundary sizes, and whether boundaries alone are the correct approach for RM triggers.

4.7 Related Work

Our MAAUV architecture shares key properties with ones found in work on self-adaptive systems (SASs) [4] [30] [42], where a managing system (or controller) can modify the behavior or configuration of a managed system in response to changes in the operating environment [54]. In the SAS context, our RM is the managing system and the MAAUV actuators comprise the managed one.

The SAS concept is illustrated well in [10] using a self-adaptive unmanned underwater vehicle (UUV) whose controller balances safety and energy efficiency by only using sensors that require more power when more efficient ones are unavailable. The controller also has a "failsafe" mode that stops the UUV if no valid configuration of sensors is available, or the controller takes more than a pre-programmed amount of time trying to determine a configuration. Once a valid configuration is determined, the mission resumes. This is very similar to our RM concept.

The dynamic assurance cases (DACs) in [4] use artificial intelligence and machine learning (AI/ML) for safety and mission assurance. Although we claim that AI/ML is too immature to police itself or another AI/ML system, DACs address the same things we do, i.e., mission assurance and risk for a defined application and operating environment, runtime monitoring of relevant variables, and reacting to situations not explicitly envisioned at design time.

A state-machine based approach to runtime safety is proposed in [39], with an illustrative example on vehicle platooning using cooperative or adaptive cruise control (CACC/ACC). The example state machine contains a "safe" state and a number of states where one or more constraints are violated and recovery actions are defined. Most of the states use CACC between all vehicles, except state S5, which captures the case where a safety constraint cannot be verified, e.g., due to a malfunctioning sensor. In this case, the

vehicle reverts to its self-contained ACC system, reducing mission efficiency, but retaining safety. This is analogous to our RM taking over when circumstances arise requiring corrective actions that are not being made by other onboard systems.

A flowchart for safe dynamic reconfiguration is presented in Fig. 2 of [27], with steps 5, 6, and 8 capturing our RM concept, i.e., constantly evaluating system and environmental variables, correcting if needed, or executing a safety function if conditions exceed tolerable bounds. However, these steps are explicitly out of scope in [27].

Several other methodologies can be applied to RM design. STAMP and STPA [43] take a system-theoretic view of safety analysis to address the emergent properties of complex modern systems. STPA-Sec [76] extends STPA to cover both unintentional accidents and intentional attacks against system security. STPA-SafeSec [29] extends STPA to address the safety and security of cyber-physical systems specifically. Our IMARA methodology (see Chapter 5) can augment any of these methodologies.

4.8 Conclusions

Our DESCRETE testbed demonstrated the ability to provide useful results for engineering, testing, and experimenting with mission assurance for AUVs. We chose a specific mission and AUV, as well as RMs implementing several recover functions responsible for assuring that safety is preserved and mission objectives are met. The results show that the primitive RC-1 recover function, despite performing fairly well in regards to safety, does nothing for mission-assurance. More-effective recover functions, like RC-2 and RC-6, can improve mission quality, provided they can be implemented with adequate assurances of their functionality and dependability.

The results also strongly suggest that with the current conflict-avoidance algorithm, no matter how large the WB size is, conflicts between more than two AUVs can whittle down separation and lead to LOS or collisions. A better algorithm is required. Even with a better algorithm, though, the current WB/SB-based design appears inadequate to provide both an RM enough time to act before mission quality is

affected, and a potentially malfunctioning AE a chance to correct the situation. The addition of an RB should help establish a balance.

The results and observations above help clarify the WB's purpose. It shouldn't invoke a signal for the AE to immediately start an avoidance maneuver. Instead, the WB should be a sort of "control-alt-delete" to poke an AE that may have experienced a fault, possibly putting it into a "safe" mode or reloading it from a trusted software image onboard. Whatever crossing the WB causes the AE to do, until LOS or perhaps the RB is crossed, the AE is in charge. It should be assessing avoidance on its own, ideally reacting to potential conflicts long before they occur, e.g., by avoiding crowded areas and going to other POIs until the crowd breaks up. Even when inside a WB, the AE might maintain present course and depth because it expects the potential conflict to resolve itself and no reaction is the best one.

4.9 Future Work

Continued research is required on warning and other triggers beyond simple vehicle boundaries, their ratios, etc. Predictive algorithms could use future AUV positions, as determined from HCS data and other sensors like onboard sonar, to detect and warn the AE of potential LOS conflicts long before they occur. Small corrections made early on are better than drastic maneuvers made at the last minute. Also, the current collision-avoidance algorithm does not utilize the AUVs' speed. In some cases, slowing down or stopping to let another AUV pass might be the best way to avoid a conflict.

Our experimentation was limited to partner submersibles that were all well aware of each other's positions, depths, headings, and speeds through HCS. The mission scenario in Section 2.2 includes non-partner submersibles that do not participate in HCS and require detection by sonar. Such vehicles should be included in future experimentation. Given the reliance on HCS, investigating the effects of limiting its availability/accuracy is also required.

All the sensor inputs in our MAAUV are currently considered 100% available and accurate, which takes a lot of the guesswork out of what the RM should do next. Real sensors fail and their readings have margins

of error, etc. While we see our research and sensor reliability as disjoint for now, incorporating multiple redundant sensors and failsafe states, like [10] does, would be valuable future work.

The mission scenario includes surface vessels and describes constraints on them, like coordinating via HCS when AUVs surface, but we did not experiment with them. Surface vessels can be readily simulated in DESCRETE, but we prioritized exploring interactions between the MAAUV and other AUVs. We would like to introduce surface vessels to explore their interactions with AUVs, especially related to recover functions like RC-1 (see Section 3.4.4), where the AUV has little control when or where it surfaces.

Finally, we cannot ignore the fidelity of the DESCRETE simulator, and the deficiencies of simulators, in general. All the AUVs move ideally through the water, with no localized disturbances from currents or eddies, variations in propeller rotation rates, control-surface flutter, etc. While the subtle and asymmetric forces produced by such things can increase or decrease the chances of a given collision, we cannot just average them out. The next step for DESCRETE is to model a real submersible, experiment on it, and compare the results with actual field-test data. The U-Boat Worx NEMO [50] mentioned in Section 4.5 is a possible candidate. As Figure 2-7 shows, the NEMO is roughly spherical. DESCRETE's simulation techniques and spherical collision-avoidance boundaries apply well here, requiring only minor modifications to the Test002 boundary sizes. Of course, a new model is also required. The NEMO is slightly larger than the MAAUV-Lite, 2.80 m versus 2.45 m, and has the same maximum speed and depth limits, 1.5 m/s and 100 m, respectively. With its multiple thrusters, the NEMO is more maneuverable. An initial model could simulate this with an extremely small turn radius, though pivoting in place would require additional development. A NEMO-Lite model could be constructed relatively quickly. Before that, however, we must determine the missions where an autonomous NEMO would be valuable.

5 A METHODOLOGY FOR ITERATIVE MISSION-ASSURANCE REFINEMENT ANALYSIS

5.1 Introduction

As autonomous vehicles (AVs) become ever-increasingly part of our daily lives, much research is going into making them safer, as well as more secure from cyber threats. Many methodologies address the safety/security of cyber-physical systems (CPSs) [40], of which AVs are one type. Safety and security by themselves are insufficient, however. Just as "the only truly secure system is one that is powered off…" [64], the only truly safe AV is one that never leaves the driveway. However, that's not what they're built for. Any AV, whether its domain is land, air, space, or on/under water, is built with a "mission" in mind. Existing and proposed missions include surveying pipelines, exploring places not readily accessible to humans, delivering packages, dusting crops, ferrying passengers, and many others. As such, in addition to safety, "mission assurance" must be addressed, i.e., getting the AV's job done despite foreseen and unforeseen circumstances.

Determining how an abstract concept like mission assurance applies to AV systems that implement capabilities to meet mission objectives is no small feat, given the complexity of AVs used for even modest missions. While some dependencies are obvious, e.g., requiring charged batteries or fuel in the tank for mobility, others are less straightforward, like the sensors, topographic/bathymetric databases, communications links, and time/position references for basic navigation, potential collision detection and avoidance. Identifying the system(s) that must be operational to meet each mission objective is an essential part of improving mission assurance.

Our iterative mission-assurance refinement analysis (IMARA) methodology evolved from research in trading mission quality for mission assurance in autonomous underwater vehicles (AUVs). Imara is also, coincidentally, a Swahili word for firm, stable, or strong [35]. To prevent adverse effects from unexpected behaviors, our mission-assured AUV (MAAUV) included a high-assurance resilience module (RM),

described in Chapter 3, between its autonomy engine (AE) and sensors/actuators (see Figure 1-3). Part of the RM's role is to prevent undesirable AE commands from reaching the actuators.

The RM is capable of taking over AUV operations and performing some mission-related functions, though perhaps not as many as the AE, or as well as the AE would perform them. This is where IMARA is used to determine which mission objectives the RM must meet, the AUV systems required to meet those objectives, and the assurances required for those systems.

Our contributions are as follows: (1) we propose a methodology for iterative mission-assurance refinement analysis to maximize confidence in the achievability of key mission objectives, and (2) in Chapter 6, we apply the methodology to our MAAUV use case.

This remainder of this chapter is organized as follows. Section 5.2 describes the methodology and its iterative approach. Section 5.3 discusses the benefits of IMARA and one possible approach to automating it. Section 5.4 addresses related work in the literature.

5.2 IMARA Methodology Description

Improving mission assurance requires examining not only the mission level, but also the system and component ones. The system level and below are represented by a number of methodologies, discussed in Section 5.4. IMARA focuses on the mission level, analyzing how mission objectives are met by capabilities implemented by systems. IMARA stops at the system boundary, reducing that portion of the analysis to whether a system is operational, or has failed. How or why the system failed is not in IMARA's scope, only the impact to the mission of its failure. This binary view does not mean a "system" must be everything that resides in one box. For example, a multi-function printer-copier-fax-scanner can be represented as four (or more) systems. Conversely, several physical systems could be analyzed as one logical one, e.g., global-positioning system (GPS) satellites, their ground stations, and a user's receiver.

IMARA is an iterative approach for improving mission assurance using an assurance-refinement (AR) loop, depicted graphically in Figure 5-1.

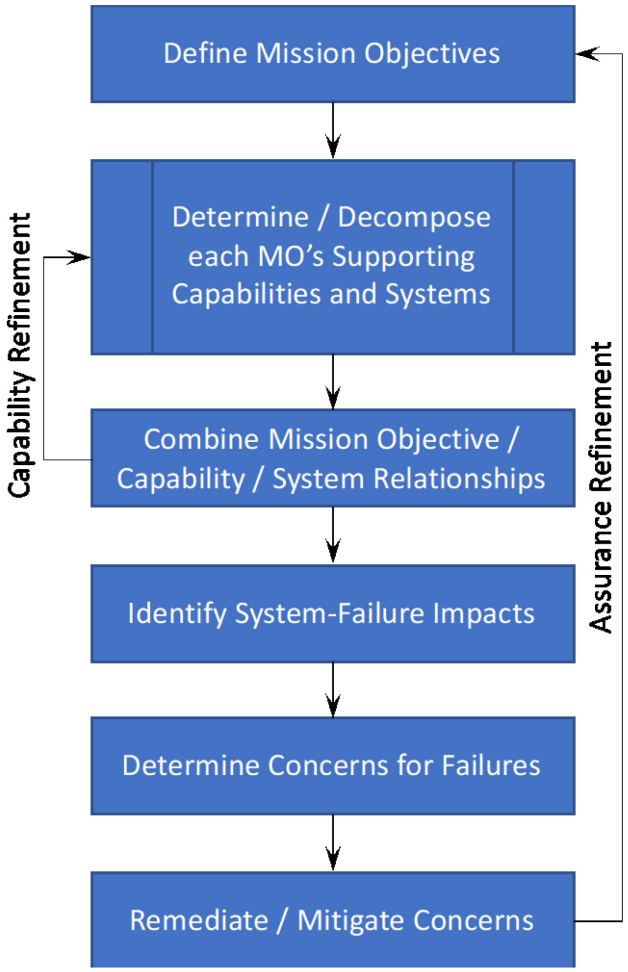


Figure 5-1 IMARA overview flow diagram

The steps of each iteration are described below using the following terminology:

- *Mission Objectives* (MOs) are a mission's goals, i.e., what must be accomplished. They can be *met*, *partially met*, or *unmet*.
- *Capabilities* are the logical functions used to meet MOs. They can be *available*, *partially available*, or *unavailable*.
- *Systems* are the physical implementations of the functions required to provide capabilities. They can be *operational*, *partially operational*, or *failed*.

What constitutes partially met MOs, partially available capabilities, and partially operational systems is likely specific to a given mission and the systems used to implement/accomplish it. These intermediate values should be defined as objectively as possible. For example, a partially operational system can be described by the operational/failed status of individual functions, or by the percentage of certain functions' availability, e.g., the amount of energy remaining in a battery. Systems (capabilities) can degrade, but remain operational (available). They fail (become unavailable) when the degradation exceeds a certain amount, which also should be well-defined. Lastly, depending on the nature of any degradations, and the architecture of the overall set of systems performing the mission, MOs might still be partially or fully met with failed systems or unavailable capabilities.

5.2.1 Define Mission Objectives

How well a problem is solved is closely related to how well it's described and understood. In software engineering, problems are described by requirements documents [37], use cases [6], user stories [16], etc. An IMARA analysis describes the problem it's trying to solve, i.e., mission assurance, using MOs. Well-defined MOs with quantified and testable criteria for being met lead to the best IMARA results.

Any prioritization of MOs should be captured to inform assurance-implementation decisions later on in the methodology, e.g., landing on the moon is important, but getting the crew home safely is far more so. In some cases, two MOs are only important if both can be met. Otherwise, a third MO may become more important. MOs should also capture mission constraints. For example, an MO to "navigate a route" suggests it could be met using a GPS system. However, if the route is underwater, GPS may not be available. In that case, the MO should be, "navigate a route in a GPS-denied environment."

5.2.2 Determine/Decompose Supporting Capabilities and Systems

MOs are met using capabilities implemented with systems. A capability or system can support multiple capabilities. Also, a capability can be supported by multiple capabilities, systems, or a combination of both.

In IMARA, a system is any implementation of capabilities, ranging from a simple magnetic compass to a complex nuclear power plant. IMARA's analysis is bounded at the system level, focusing on what capabilities systems provide, not how they provide them. Detailed system analysis is left to the many methodologies whose focus is there, e.g., STPA-SafeSec [29]. To facilitate the failure analysis below, systems with many functions should be abstracted to support different capabilities for each function. For example, a GPS receiver provides position information, and also provides accurate time. Having the receiver support "location" and "timing" capabilities allows for independent analysis of these functions failing, as well as a means for adding redundancy to one, but not necessarily the other.

For a "clean-sheet" design with no existing implementation, a top-down approach can iterate through the MOs and determine the capabilities required to meet each one, decomposing capabilities until systems to implement them can be determined. High-level capabilities might reflect MOs closely, e.g., a "navigation" capability to meet an MO for point-to-point travel on a map. As capabilities are decomposed, lower-level ones become less MO-focused and more-closely reflect systems that implement them. For example, a "current position" capability supporting navigation is specific enough for a GPS-based implementation, assuming no mission constraints against using GPS.

If an implementation already exists, and is being re-purposed for a new mission, a bottom-up approach might identify how its systems provide lower-level capabilities, combining those to form higher-level ones, and possibly adding or augmenting systems to close gaps. A hybrid approach is also possible.

When capturing MO, capability, and system relationships, capture direct and indirect ones, where the former are immediate supporters of an MO or higher-level capability, and the latter support the immediate ones. This serves as a sanity check that all capabilities/systems supporting a given MO or capability are accounted for. It also facilitates system-failure impact analysis in later steps of the methodology.

5.2.3 Combine Mission Objective/Capability/System Relationships

The systems tasked to meet a set of MOs are often bound to each other somehow physically to produce a system of systems, e.g., in the case of a vehicle, which IMARA calls a "platform." A nuclear power plant, oil refinery, or bakery is also a platform in IMARA, though the term is less intuitive in those cases. To incorporate this system-of-systems binding, once the support relationships are determined for each MO, they are combined to form a single platform-centric optimal set. Possible outcomes from this optimization include elimination of duplication, reduction in overall relationship complexity, or reorganization of capabilities and systems. Later, system-failure impact analysis may determine a certain amount of redundancy is required for assurance, for example, but at this stage of IMARA the focus is on optimization.

The loop consisting of this step and the prior one, shown in Figure 5-1, addresses the situation where supporting capabilities and systems determined for each individual MO are not necessarily the best ones for combining MOs. Capabilities in particular, because they represent what can be somewhat arbitrary decompositions from meeting MOs to determining systems for implementation, may require refinement to support multiple MOs better. For example, only part of a complex capability to meet one MO may be required by another. In that case, the capability could be further decomposed so that each MO is only dependent on the part that it needs. Conversely, if a set of capabilities is required by multiple MOs, a new capability supported by the original ones could be created and used instead. For example, most MOs for a vehicle require it to move. Rather than having, e.g., "propulsion" and "steering" capabilities directly supporting every MO, it might be easier to have them support a new "mobility" capability, which then supports the MOs. This reduces relationship complexity, facilitating the system-failure impact analysis that follows.

5.2.4 Identify System-Failure Impacts

Once the relationships between MOs, capabilities, and systems are determined, IMARA can analyze the impacts of systems failing. System failures can lead to capabilities becoming unavailable and MOs going unmet. However, not every system failure necessarily leads to unmet MOs. One reason for this is that systems are often built with everything needed to perform their functions, even if other systems on the platform also provide those functions. A simple example is the number of clocks in a typical modern kitchen. The failure of one system does not mean the cook does not know the time.

IMARA's system-failure impact analysis is comprehensive and covers all system-failure combinations, which, for N systems, leads to up to 2^N-1 failure cases. However, this number can typically be reduced significantly, e.g., because some systems are critical for meeting MOs and, if they fail, the status of the other systems is irrelevant. For example, a vehicle with an MO to transport cargo experiencing an engine failure leads to the same unmet MO outcome, collapsing roughly half of the cases. Such failure cases can be discovered early on and collapsed by first analyzing the impact of single systems failing, then pairs failing, etc. Analyzing the failure of all systems required for a given capability can also eliminate many cases at once. For example, if a motor, battery, and transmission are required for a vehicle, the operational status of other systems probably doesn't matter if these three have all failed.

5.2.5 Determine Concerns for Failures

The system-failure impact analysis produces a list of failure cases without addressing, e.g., the likelihood or the resultant state of the platform after a given failure has occurred. Mission operators and system designers, however, need to know how concerned they should be about each failure case. This step determines those concerns so that remediating and mitigating them in the next step can be prioritized.

The most likely failures are usually associated with complex systems like engines or computers and other electronics. As such, these kinds of systems get the most attention when limited resources are

available for improving mission assurance. Conversely, some failure cases are deemed all but impossible and cause only minimal concern. Still, unknown phenomena or system properties only discovered after several failures occurred in a brief timespan, or in a new field, have led to advances in fields such as aircraft design [17], so no failure case should ever be completely dismissed outright.

5.2.6 Remediate / Mitigate Concerns

IMARA defines remediation as reducing the likelihood of a failure case occurring, and mitigation as reducing its impact. Remediation might involve "hardening" a system, i.e., making it less likely to fail, perhaps through system or component redundancy/diversity. This most likely requires augmenting or adding systems. Eventually, further remediation is not possible, at which point, mitigations are added to achieve as many of the most-important MOs as possible. Both remediations and mitigations must consider what is practical for the platform and its mission, which is discussed further in the next section. Mitigations must also consider the state of the platform after a failure has occurred.

5.2.7 Execute Assurance-Refinement AR Loops

As shown in Figure 5-1, once all remediations and mitigations have been applied for an AR iteration, IMARA loops to the beginning. This continues until no further practical improvements to mission assurance can be identified. Practical improvements are those whose benefits outweigh their added costs, e.g., financially, as increases to weight, space, power, cooling, etc. beyond system capacity, or as unacceptable schedule delays.

To find practical improvements, the systems are examined first for where classical "reliability engineering" techniques such as redundancy, improved quality of materials, better software development practices, etc., can be applied. More expensive systems that are also more reliable might be substituted, as long as the benefits outweigh the costs. A key contribution of IMARA is clearly defining MOs and determining that they are met. If the MOs are defined correctly, there is no need to exceed them, which substituting a more-expensive system might do. Therefore, designers can also investigate more-reliable

less-capable systems, as long as they can still meet the MOs. Such “lesser” systems may not be as complicated, which may be part of why they are more reliable.

If no further system-level improvements are identified, the system architecture is examined. Sometimes capabilities or systems initially meant to support one MO or capability can also be used to improve assurance in others. Such synergies may not become evident until after, e.g., a few instances of the mission have been executed, and a “re-wiring” of the architecture could improve mission assurance without changing or replacing systems, or adding redundant ones.

Once the systems and architecture improvements have reached their practical limits, mission assurance might be improved further by adding capabilities, which probably means adding systems. Adding systems, like all assurance improvements, must be weighed against the costs mentioned above.

Once the system-centric approach has been exhausted, the remaining way to improve mission assurance is by refining MOs. MOs may contain desirable, but unnecessary, objectives that can be eliminated. At some point, however, refinement yields the minimum set of MOs that achieve an acceptable mission. At that point, only systems and architecture can contribute further improvements.

5.3 Discussion

IMARA addresses an important need in mission-assurance, i.e., tying mission failures to system failures. This has value in a top-down approach, whereby determining how MOs depend on systems identifies mission-critical ones. There is also value in a bottom-up approach, where systems with identified vulnerabilities can be assessed for how exploitation of those vulnerabilities affects MOs. Either way, IMARA bootstraps systems analysis methodologies, like STPA-SafeSec [29], so that limited analysis, remediation, and mitigation resources are allocated to the most important systems.

IMARA’s iterative process is readily suitable for automation. For example, a model-based systems engineering (MBSE) tool, like Cameo Systems Modeler [11], allows users to construct the MO/capability/system relationships discussed in Section 5.2.2 and Section 5.2.3. A plugin, in Cameo

parlance, could be developed to extract all possible system-failure impact cases (Section 5.2.4). To keep the number of failure cases manageable, the plugin could allow users to identify systems, or combinations of systems, whose failures overshadow the failures of other systems with respect to meeting MOs, allowing the collapsing of failure cases using don't-care values. Users could then be queried to capture concerns regarding the remaining failure cases, both for assurance analysis documentation purposes, and as the first step in selecting systems to analyze for assurance improvements.

IMARA can be used in situations involving circular capability/system dependencies, such as a vehicle with a power-generation system that requires cooling, and also provides power to the cooling system. How to approach such interdependent systems with IMARA is situation specific, and partly depends on how MOs are constructed. For the above example, instead of an MO that simply requires power to be available at all times, a more specific one might require that power remain available for a certain period after a cooling failure, allowing other mitigations to be performed during that time.

5.4 Related Work

The safety/security community has been aware for some time that component-level failure analyses are insufficient for modern systems of systems. Due to complex interdependencies and emergent properties, a systematic approach to safety and security analysis is required. STAMP and STPA [43] take a system-theoretic view of safety analysis. STPA-Sec [76] extends STPA to cover both unintentional accidents and intentional attacks against system security. STPA-SafeSec [29] extends STPA to address CPS safety and security specifically. Any of these methodologies can be informed by IMARA's mission-level analysis. Moreover, applications of these methodologies to autonomous maritime systems [22] [77] can be applied to the MAAUV use case.

There are many CPS safety/cybersecurity co-engineering methods. However, of the 68 reviewed in [40], only nine involve users—the people executing the mission—as stakeholders. CSRM [13] and Mission Aware [28] recognize that a CPS is part of a broader mission context. They are not reviewed in [40]. Both

are similar to IMARA in the activities performed, but differ because of their cybersecurity focus, as IMARA addresses all forms of system failures. While IMARA is more general in its failure coverage, it stops at the system boundary. CSRM and Mission Aware also analyze the systems, using STAMP/STPA-like approaches. CSRM specifically relies on SysML [20] models, which IMARA also endorses. A merging of all three methodologies would realize the benefits of all of them.

Several works cover IMARA's MO-definition step (Section 5.2.1), and their approaches can be used to implement it. The War Room [12] is an elicitation exercise with mission stakeholders that serves to gather mission goals, objectives, expectations, and procedures. It directly informs a modified STPA-Sec analysis, but is stated not to provide exhaustive potential-failure results. IMARA fills this gap by exhaustively iterating through all system failure cases, further informing methodologies like STPA-Sec regarding prioritization of the systems to analyze.

IMMS [21] uses the Event-B formal method [1] to identify and analyze mission requirements for squads of autonomous maritime missions. This approach is an improvement over classical requirements documents, for example, which are prone to different interpretations.

5.5 Conclusion

We have presented a methodology for iterative mission-assurance refinement analysis (IMARA) to maximize confidence in the achievability of MOs. IMARA ties MOs to the systems designed to meet them through the abstraction of possibly multiple levels of capabilities, ultimately provided by those systems to support the MOs. The MO/capability/system support relationships are used to comprehensively analyze the impact of system failures, determine failure-related concerns regarding meeting MOs, and develop remediations/mitigations to eliminate or reduce their impact. IMARA's mission-level results inform system- and component-level safety and security analysis methodologies, like STPA-SafeSec [29], allowing prioritization of which systems to analyze for maximum remediation and mitigation benefits, or to allocate limited resources.

6 APPLYING IMARA TO THE MAAUV

6.1 Introduction

This chapter applies the iterative mission-assurance refinement analysis (IMARA) methodology to our autonomous underwater vehicle (AUV) use case from Chapter 2. The resultant mission-assured AUV (MAAUV) represents several classes of autonomous vehicles (AVs) and their missions. The MAAUV performs underwater tours, transporting passengers to points of interest (POIs) within given tour-duration limits, in the presence of other submersibles on similar tours. Though passenger AUVs are not yet as prevalent as other AVs, e.g., self-driving cars, a MAAUV does carry passengers. Because of this, MAAUV designers must prioritize vehicle safety, unlike cases where a purely robotic vehicle may be sacrificed to prevent harm to people or damage to more-valuable property.

The example that follows is meant to showcase the use of the IMARA methodology, and is not a comprehensive design exercise. Significant required systems are deliberately left out, most notably life support for passengers. And, while we include batteries as an energy source, they are discussed only in the context of propulsion, not powering electronics, fans, pumps, etc.

The remainder of this chapter is organized as follows. Section 6.2 through Section 6.6 apply the IMARA steps from Section 5.2 to the MAAUV use case. Section 6.7 discusses the applicability of this chapter's analysis to other domains, vehicles, and missions.

6.2 Define Mission Objectives

Referring to Figure 5-1, defining mission objectives (MOs) is the first IMARA step. MOs can be derived from concepts of operations (CONOPS) [38], use cases [6], discussions with mission and systems specialists, etc. For our MAAUV case, there are four MOs:

- MO-1: Maintain Separation
  - All AUVs must keep minimum distances from each other, POIs, and the seabed.
  - Failing to keep these minimum distances constitutes loss of separation (LOS).

- MO-2: Return Passengers to Pier
  - On tour completion, the MAAUV's passengers must be returned to the pier.
  - Passengers also must be returned if a tour is cut short (see MO-3).
- MO-3: Complete Tour Within Maximum-Allowed Tour Duration (TD-Max)
  - Passengers do not want to tour longer than a certain amount of time.
  - The MO also factors in the amount of available battery power in the MAAUV.
- MO-4: Maximize POI Visits
  - The MAAUV is rewarded for visiting the most POIs possible.
  - POI attractiveness can be weighted with level-of-interest (LOI) values.

The MAAUV use case evaluates the quality of a tour by scoring, i.e., representing the value of the above MOs using the equations defined in Section 2.7. Using tour scores to measure tour quality is only meaningful, however, within a given mission-operator's priorities. For example, only a mission operator can determine whether visiting all the POIs in a given scenario with a 20% LOS is better or worse than visiting 80% of the POIs with no LOS, because both outcomes yield the same score.

6.3 Assurance-Refinement Iteration #1 (AR-1)

6.3.1 AR-1 Supporting Capabilities and Systems

The IMARA methodology supports a top-down clean-sheet design from MOs to systems, a bottom-up approach fitting an existing set of systems to MOs, or a hybrid approach. The AUV already exists as a simulation, so Step #2 of the analysis first identifies the current systems in Table 6-1.

Reference data is provided by S-01, S-02, and S-09. S-01 and S-02 provide time and own-position data, respectively. A highly-accurate clock in S-02 is also a time reference. S-09 provides bathymetry data for the seabed, including any land masses that extend above the water. S-09 also provides the locations of all POIs and piers.

Table 6-1 MAAUV Systems Entering AR-1

ID	System
S-01	Onboard Clock
S-02	Inertial Navigation System (INS)
S-03	Harbor Control – Navigation (HC-N)
S-04	Propeller
S-05	Main Drive
S-06	Batteries
S-07	Rudder
S-08	Diving Planes
S-09	Electronic Chart Database
S-10	Autonomy Engine
S-11	Sonar

Sensors, S-03 and S-11, provide information about the outside world. S-03 is the receiver for an acoustic communications network that provides position information for all AUVs, to all AUVs. The infrastructure providing the information received by S-03 uses a collection of transducers to precisely locate the AUVs, which also makes S-03 an independent own-position reference from S-02. S-11 provides direct information about objects around the AUV. S-11 has enough detection range for avoiding LOS, but not nearly as much as the “infinite” range of S-03.

Actuators, S-04 through S-08, provide mobility, i.e., propulsion, steering, and diving. While the interrelationships in a propulsion/maneuvering system can be very complex, the IMARA analysis simplifies this, requiring all five systems to be operational for mobility to be available.

The autonomy engine (AE), S-10, commands actuators using reference data and sensor inputs to meet MOs. The analysis assumes a general-purpose computer, possibly “ruggedized” for the environment, running a combination of open-source, commercial, and custom software. At least some artificial intelligence and machine-learning (AI/ML) is assumed, partly because this is a common approach for implementing systems of this type, and this dissertation investigates assurance concerns in AI/ML.

Using a hybrid approach, Step #2 of the analysis continues, defining the capabilities supporting the MOs in Section 6.2, decomposing them until they can be mapped to systems, and listing them in Table 6-2.

MO-1 is supported by a high-level conflict avoidance capability (C-01), which is supported by high-level conflict detection (C-04) to see potential LOS, and mobility (C-05) to avoid it. C-04 is decomposed to direct (C-08) and indirect (C-07) capabilities. C-08 is provided by sonar (S-11), and C-07 is provided primarily by HC-N (S-03) via two vehicle position capabilities (C-09 and C-11), and terrain & landmark information (C-10) using electronic charts (S-09).

Table 6-2 MAAUV Capabilities Entering AR-1

ID	Capability
C-01	Conflict Avoidance
C-02	Touring
C-03	Timing
C-04	Conflict Detection
C-05	Mobility
C-06	Navigation
C-07	Indirect Conflict Detection
C-08	Direct Conflict Detection
C-09	Own Position and Orientation
C-10	Terrain and Landmark Information
C-11	Vehicle Positions and Orientations

MO-2 through MO-4 are supported by a high-level touring capability (C-02), which is supported by navigation (C-06) and mobility (C-05). C-06 is supported by many of the same capabilities as C-07. MO-3 is also directly supported by timing (C-03), to compare estimated and actual-elapsed tour durations to TD-Max. C-03 supports many other capabilities, directly or indirectly.

6.3.2 AR-1 Combined Mission Objective/Capability/System Relationships

Step #3 of the IMARA methodology combines the capability/system relationships for each MO to produce both graphical (Figure 6-1) and tabular (Table 6-3) representations. The "OR" symbols in Figure 6-1 indicate that the capability on the left-hand side can be wholly supported by any of the

capabilities or systems on the right-hand side. In Table 6-3, a ‘D’ indicates direct support of a capability, while an ‘I’ indicates indirect support.

6.3.3 AR-1 System-Failure Impacts and Concerns

Step #4 of the IMARA methodology analyzes the impact of all possible system-failure combinations. With 11 systems, this suggests there are 2047 failure cases to analyze. However, because many of the systems are single points of failure, the number of cases collapses quickly, as shown in Table 6-4. Concerns related to the remaining failure cases, and remediation/mitigation options, are shown in Table 6-5.

The case numbers in the left-most column of Table 6-4 are computed by treating each failed system as a bit in an eleven-bit value, read left to right in the “System Failures” columns. An ‘O’ in those columns indicates an operational system, and an ‘F’ indicates a failed one. For the “Capability Availability” columns, an ‘A’ indicates an available capability, and a ‘U’ indicates an unavailable one. For the “Mission Objectives Met” columns, an ‘M’ indicates a met MO, and a ‘U’ indicates an unmet one. A ‘P’ indicates partially available or partially met, depending on which column it appears. Cells containing an ‘X’ indicate a “don’t-care” value, meaning that the failure case is the same, regardless of the system’s, capability’s, or MO’s individual status.

The analysis determined that S-04 through S-11 are mission-critical. If any of these systems fails, that failure propagates up supporting dependencies, making mission-critical capabilities unavailable, causing one or more MOs to become unmet. If both S-02 and S-03 fail, a mission-critical capability (C-09) becomes unavailable. If S-03 fails, but S-02 remains operational, in theory MO-2 through MO-4 could be met. However, without indirect vehicle conflict detection, the MAAUV would only have sonar to detect conflicts directly.

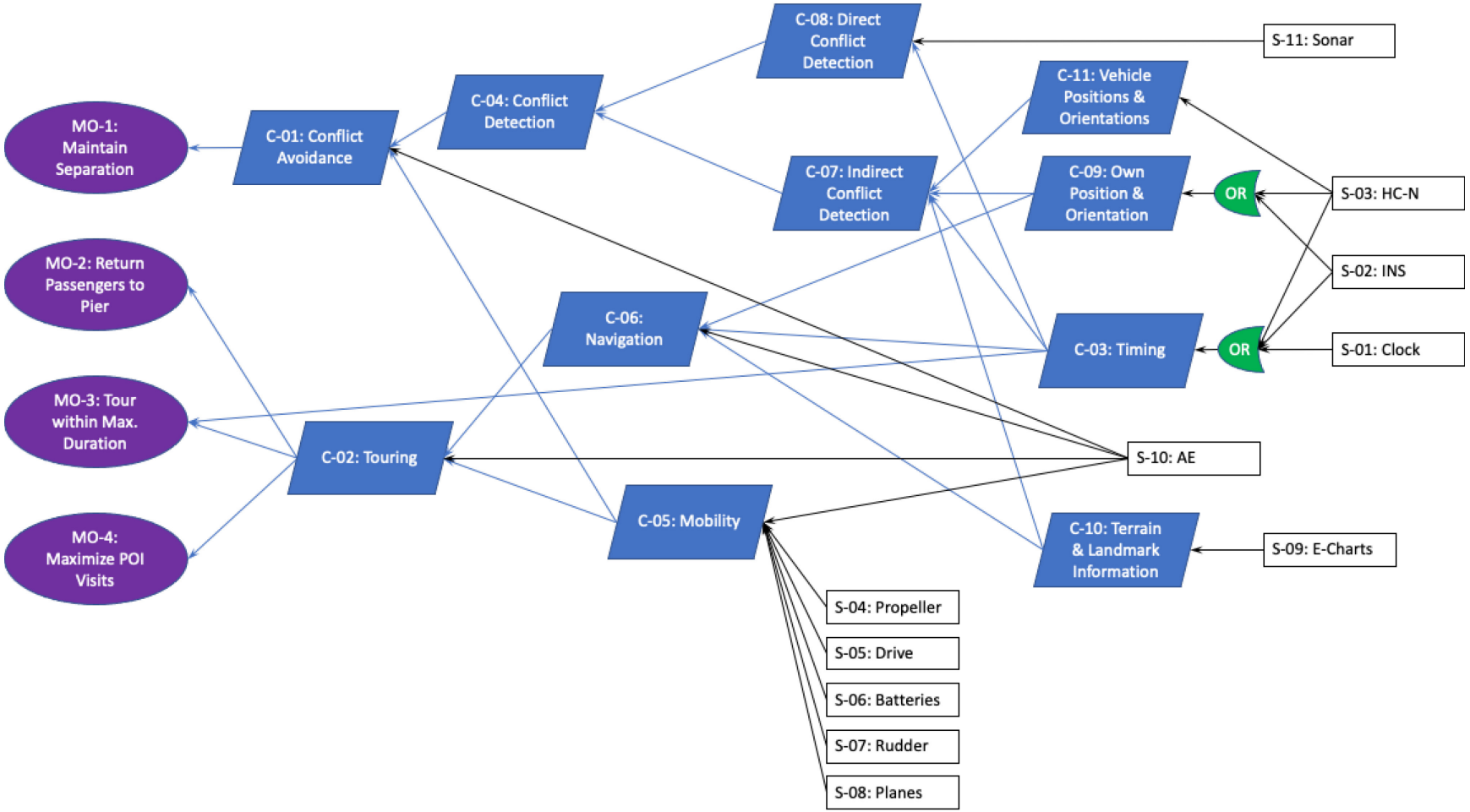


Figure 6-1 MAAUV AR-1 combined MO, capability, and system relationships

Table 6-3 MAAUV AR-1 Combined MO, Capability, and System Relationships

	Supporting Capabilities										
MO/C	**C-01**	**C-02**	**C-03**	**C-04**	**C-05**	**C-06**	**C-07**	**C-08**	**C-09**	**C-10**	**C-11**
MO-1	D		I	I	I		I	I	I	I	I
MO-2		D	I		I	I			I	I	
MO-3		D	D		I	I			I	I	
MO-4		D	I		I	I			I	I	
C-01			I	D	D		I	I	I	I	I
C-02			I		D	D			I	I	
C-03											
C-04			I				D	D	I	I	I
C-05											
C-06			D						D	D	
C-07			D						D	D	D
C-08			D								
C-09											
C-10											
C-11											

	Supporting Systems										
Capability	**S-01**	**S-02**	**S-03**	**S-04**	**S-05**	**S-06**	**S-07**	**S-08**	**S-09**	**S-10**	**S-11**
C-01	I	I	I	I	I	I	I	I	I	D	I
C-02	I	I	I	I	I	I	I	I	I	D	
C-03	D	D	D								
C-04	I	I	I						I		I
C-05				D	D	D	D	D		D	
C-06	I	I	I						I	D	
C-07	I	I	I						I		
C-08	I	I	I								D
C-09		D	D								
C-10									D		
C-11			D								

As shown, the analysis can produce a failure case #0, i.e., no failed systems. Normally, the "all's well" case is not analyzed, but cases with only don't-care values collapse to case #0, indicating that, as long as all the other systems are operational, these don't-care ones are non-critical. The AR-1 analysis determined that S-01 and S-02 are non-critical, as long as S-03 is operational. In fact, S-01 appears unnecessary based on the analysis. Therefore, if there was ever a power, space or cooling concern with systems on the MAAUV, S-01 could probably be removed.

6.3.4 AR-1 Remediation & Mitigation

Table 6-5 shows the remediation and mitigation for the failure cases from the MAAUV AR-1 analysis. The gold highlighting shows areas where remediation or mitigation can be applied to the related systems. The first AR-loop iteration highlights all failure cases, while subsequent iterations only highlight those where system or architectural changes were introduced by the prior iteration. This, though not strictly required for the methodology, is a useful tool to eliminate revisiting systems that have already been remediated/mitigated to the maximum extent practical.

Table 6-4 MAAUV AR-1 System-Failure Impacts

	System Failures											Capability Availability											Mission Objectives Met			
Case #:	S-01	S-02	S-03	S-04	S-05	S-06	S-07	S-08	S-09	S-10	S-11	C-01	C-02	C-03	C-04	C-05	C-06	C-07	C-08	C-09	C-10	C-11	MO-1	MO-2	MO-3	MO-4
0	X	X	O	O	O	O	O	O	O	O	O	A	A	A	A	A	A	A	A	A	A	A	M	M	M	M
4	X	O	F	O	O	O	O	O	O	O	O	P	A	A	P	A	A	P	A	A	A	U	P	M	M	M
6	X	F	F	X	X	X	X	X	X	X	X	X	U	X	X	X	U	U	X	U	X	U	X	U	U	U
8	X	X	X	F	X	X	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
16	X	X	X	X	F	X	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
32	X	X	X	X	X	F	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
64	X	X	X	X	X	X	F	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
128	X	X	X	X	X	X	X	F	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
256	X	X	X	X	X	X	X	X	F	X	X	X	U	X	X	X	U	X	X	X	U	X	X	U	U	U
512	X	X	X	X	X	X	X	X	X	F	X	U	U	X	X	U	U	X	X	X	X	X	U	U	U	U
1024	X	X	X	X	X	X	X	X	X	X	F	U	X	X	U	X	X	X	U	X	X	X	U	X	X	X

Table 6-5 MAAUV AR-1 Concerns, Remediation & Mitigation

Case #:	Concern	Remediation; Mitigation
0	NONE: Non-critical system failures	N/A; N/A
4	No indirect vehicle conflict detection	Reliability Eng.; RC-2
6	Mission-critical capability (C-09) unavailable	Reliability Eng.; RC-1
8	Mission-critical system failure	Reliability Eng.; RC-1
16	Mission-critical system failure	Reliability Eng.; RC-1
32	Mission-critical system failure	Reliability Eng.; RC-1
64	Mission-critical system failure	Reliability Eng.; RC-1
128	Mission-critical system failure	Reliability Eng.; RC-1
256	Mission-critical system failure	Reliability Eng.; RC-1
512	Mission-critical system failure	RM (RC-6, RC-2); RC-1
1024	Mission-critical system failure	Reliability Eng.; RC-1

Failure case #512 covers the failure of the AE (S-10). In addition to hardware failures, this case also covers software faults and cyber-attacks against the AE. While reliability engineering (see below) can remediate some AE failures to a certain degree, the MAAUV use case claims that remediating AI/ML-related failures is beyond the state of the art, necessitating another approach. Our approach is the resilience module (RM) described in Chapter 3, i.e., a high-assurance system that can sufficiently meet the MOs, provide passenger safety, and significantly mitigate the risk of AI/ML-related failure. A potential RM implementation uses a separate computer running rule-/state-based software developed using formal methods, and additional digital, analog, and mechanical sensors/actuators.

The RM implements recover functions used for mitigation in Table 6-5, and defined as:

- RC-1: Stop and surface immediately
- RC-2: Abandon the tour and return to the pier
- RC-6: Continue the tour

As the numbering suggests, there are other RCs that are not relevant to this analysis. The RM implements RC-2 and RC-6 using trusted hardware and software. RC-1 is separate from the RM computer, and primarily analog/mechanical to mitigate software-fault and cybersecurity risks. Together, these recover functions represent the mitigations for when the remediated systems still fail. The choice of which recover function to invoke depends on factors external to this analysis, including the capabilities of a given RM, mission-operator risk tolerance, the type of AE failure, the number of other AUVs in the tour area, etc. A separate “panic button” might also be included to manually invoke a recover function for safety-related events the RM cannot detect, e.g., passenger health emergencies or injuries.

As Table 6-5 shows, most system failures cause the MAAUV to abandon the mission in favor of passenger safety. The exception is failure case #4, where the MAAUV returns to the pier, though with an increased risk of conflict from another vehicle. Capturing and mitigating this and other recovery-related risks requires broader definitions than just what the MAAUV does. For example, RC-1 and RC-2 would

likely also include, e.g., an emergency beacon to alert others that the AUV is impaired so that the area where it surfaces can be cleared of conflicts, or a clear path back to the pier can be provided. The beacon would also alert rescue teams to pick up the passengers in the case of RC-1, or be prepared to do so if the RC-2 situation degrades.

The remaining failure cases in Table 6-5 use "reliability engineering" as shorthand for remediation that employs techniques such as system redundancy, improved quality of materials, better software development practices, etc. For example, a better propeller might be possible with stronger materials. Adding a protective cowling could also help. Both of these options likely involve increased vehicle weight or cost, which must be factored against any benefits.

At some point, additional remediations become impractical, and the likelihood of failure must be weighed against the mission holistically. This could lead to a wholesale AUV re-design, a change in MOs, or a number of other outcomes all outside the scope of this example. We assume the MOs must remain as they are, and that systems are remediated as much as practical in each iteration of the AR loop.

6.4 Assurance-Refinement Iteration #2 (AR-2)

6.4.1 AR-2 Supporting Capabilities and Systems

The AR-2 systems are mostly the same as those in AR-1 (see Table 6-6), except for the addition of the RM. To simplify the analysis, S-10 is updated to be an AE/RM combination, thereby increasing its assured operation. With this approach, the AR-2 capabilities remain the same as those in AR-1 (see Table 6-2).

6.4.2 AR-2 Combined Mission Objective/Capability/System Relationships

Given that the AR-2 capabilities and systems are nearly identical to those in AR-1, with the exception of adding the RM to S-10, the combined relationship diagram (see Figure 6-2) is almost identical, except for the RM addition. The tabular representation remains exactly the same as Table 6-3.

Table 6-6 MAAUV Systems Entering AR-2

ID	System
S-01	Onboard Clock
S-02	Inertial Navigation System (INS)
S-03	Harbor Control – Navigation (HC-N)
S-04	Propeller
S-05	Main Drive
S-06	Batteries
S-07	Rudder
S-08	Diving Planes
S-09	Electronic Chart Database
S-10	Autonomy Engine / Resilience Module
S-11	Sonar

6.4.3 AR-2 System-Failure Impacts, Concerns, and Remediation & Mitigation

Because the RM was incorporated into S-10, the AR-2 system-failure impact analysis looks superficially the same (see Table 6-4). Invisible at this level of the analysis, the AE/RM combination has far higher assurance than the AE does alone.

Table 6-7 shows the remediation and mitigation options for AR-2. Because the RM has been incorporated into S-10, the only remaining remediation for case #512 is to make a better RM (or AE). Also, because of the assumption that the other cases were remediated as much as practical after AR-1, the gold highlighting has been removed.

Once S-10 has been remediated as much as practical, nothing further can be done to improve the assurance of individual MAAUV systems. Analyzing the architecture, however, other improvements can be found. At present, navigation (C-06) does not factor in the locations of other vehicles. Adding indirect conflict detection (C-07) as a supporting capability allows navigation to optimize POI visits by deconflicting with the present positions of those vehicles. This directly improves how well MO-4 is met, and indirectly improves meeting MO-1 – conflicts that never happen don't need to be avoided. This is analyzed in the next AR iteration.

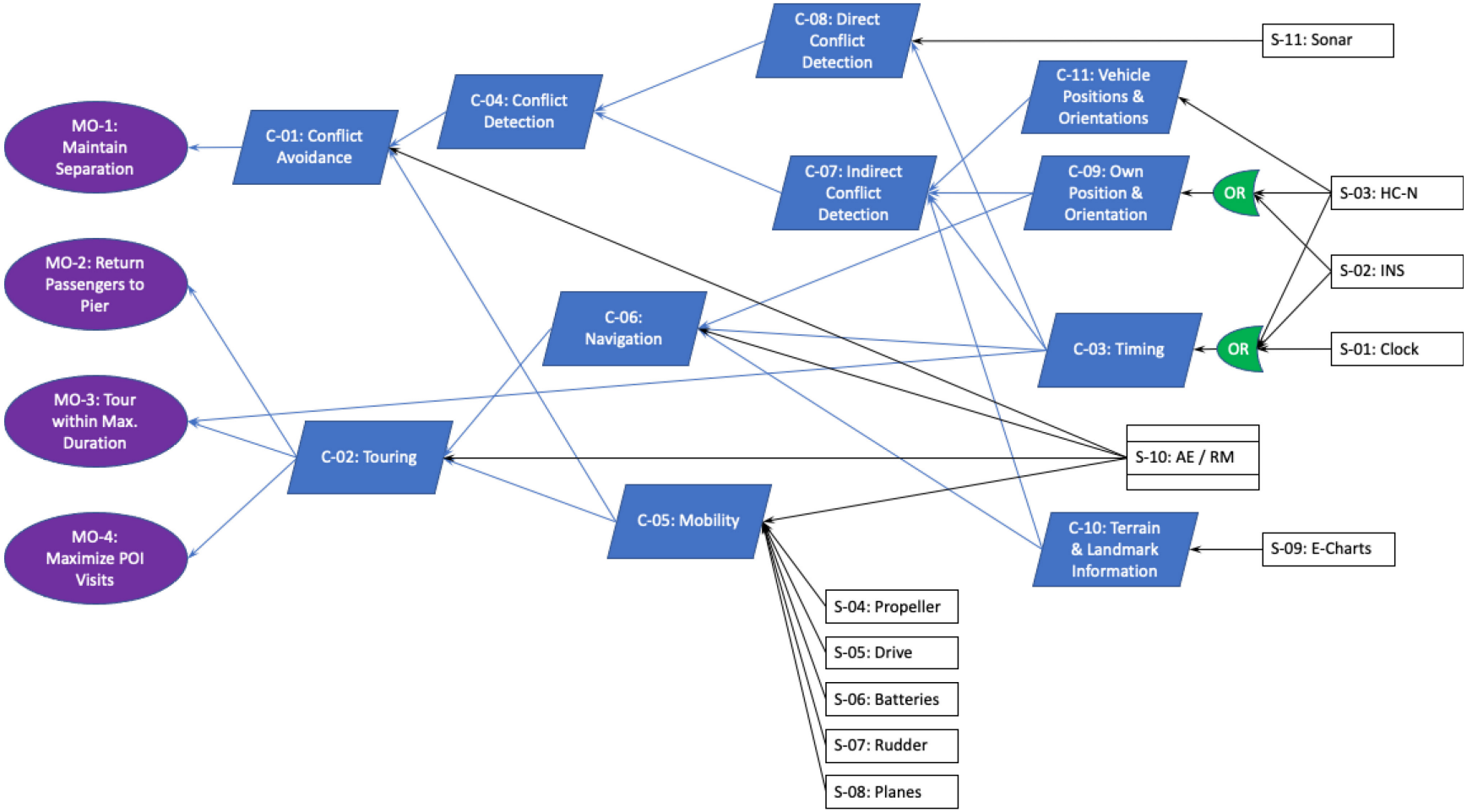


Figure 6-2 MAAUV AR-2 combined MO, capability, and system relationships

Table 6-7 MAAUV AR-2 Concerns, Remediation & Mitigation

Case #:	Concern	Remediation; Mitigation
0	NONE: Non-critical system failures	N/A; N/A
4	No indirect vehicle conflict detection	Reliability Eng.; RC-2
6	Mission-critical capability (C-09) unavailable	Reliability Eng.; RC-1
8	Mission-critical system failure	Reliability Eng.; RC-1
16	Mission-critical system failure	Reliability Eng.; RC-1
32	Mission-critical system failure	Reliability Eng.; RC-1
64	Mission-critical system failure	Reliability Eng.; RC-1
128	Mission-critical system failure	Reliability Eng.; RC-1
256	Mission-critical system failure	Reliability Eng.; RC-1
512	Mission-critical system failure	Reliability Eng.; RC-1
1024	Mission-critical system failure	Reliability Eng.; RC-1

6.5 Assurance-Refinement Iteration #3 (AR-3)

6.5.1 AR-3 Supporting Capabilities and Systems

The architectural change to have C-07 support C-06 adds no capabilities or systems in AR-3, leaving the supporting ones those in Table 6-2 and Table 6-6, respectively, except S-10 isn't new anymore. The capability/system relationships do change, however, and are discussed in the next section.

6.5.2 AR-3 Combined Mission Objective/Capability/System Relationships

For AR-3, navigation (C-06) adds indirect conflict detection (C-07) as a supporting capability. The results of this addition are shown by the new arrow from C-07 to C-06 in Figure 6-3 and the direct and indirect relationships added in Table 6-8, highlighted in gold. Note that which systems support each capability has not changed, because C-06 and C-07 already rely on most of the same underlying systems. What has changed is that (other) vehicle positions and orientations (C-11) now indirectly supports C-06, allowing navigation to plan around potential conflicts. This requires changes to the navigation capability to use the new data, which are implemented in the AE/RM (S-10). This leads to opportunities to further improve the assurances in S-10.

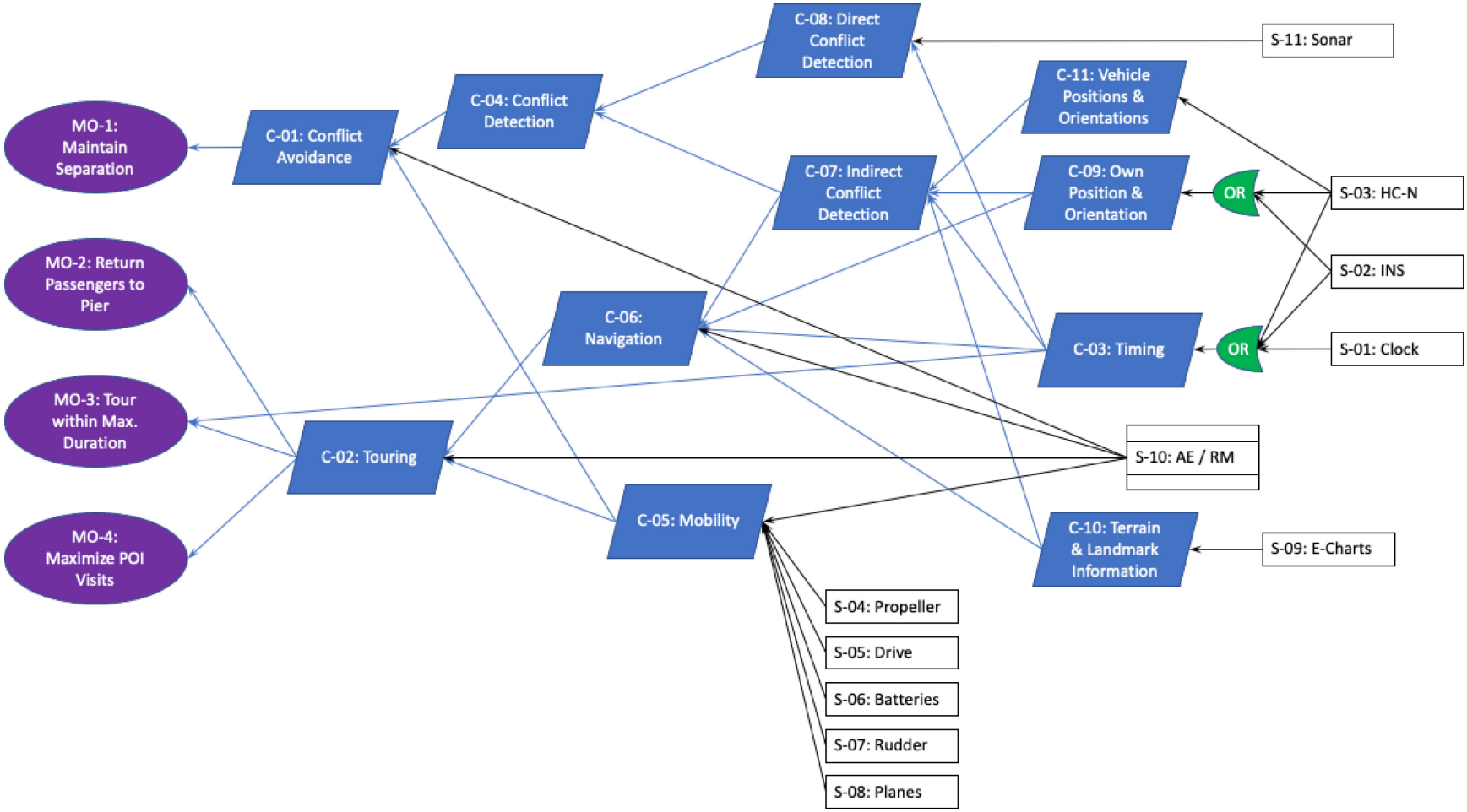


Figure 6-3 MAAUV AR-3 combined MO, capability, and system relationships

Table 6-8 MAAUV AR-3 Combined MO, Capability, and System Relationships

	Supporting Capabilities										
MO/C	**C-01**	**C-02**	**C-03**	**C-04**	**C-05**	**C-06**	**C-07**	**C-08**	**C-09**	**C-10**	**C-11**
MO-1	D		I	I	I		I	I	I	I	I
MO-2		D	I		I	I	I		I	I	I
MO-3		D	D		I	I	I		I	I	I
MO-4		D	I		I	I	I		I	I	I
C-01			I	D	D		I	I	I	I	I
C-02			I		D	D	I		I	I	I
C-03											
C-04			I				D	D	I	I	I
C-05											
C-06			D				D		D	D	I
C-07			D						D	D	D
C-08			D								
C-09											
C-10											
C-11											

	Supporting Systems										
Capability	**S-01**	**S-02**	**S-03**	**S-04**	**S-05**	**S-06**	**S-07**	**S-08**	**S-09**	**S-10**	**S-11**
C-01	I	I	I	I	I	I	I	I	I	D	I
C-02	I	I	I	I	I	I	I	I	I	D	
C-03	D	D	D								
C-04	I	I	I						I		I
C-05				D	D	D	D	D		D	
C-06	I	I	I						I	D	
C-07	I	I	I						I		
C-08	I	I	I								D
C-09		D	D								
C-10									D		
C-11			D								

6.5.3 AR-3 System-Failure Impacts, Concerns, Remediation & Mitigation

Table 6-9 shows the results of the AR-3 system-failure impact analysis, with the system failures leading to the failure cases the same as the previous AR iterations. Table 6-10 shows the concerns remain the same as well, except for failure case #4 (see below), and the opportunities for remediation and mitigation remain those from Table 6-7, with failure case #512 remaining highlighted because of the changes in the AE/RM to incorporate C-11 data in C-06.

The architecture changes analyzed in AR-3 do not improve how well MOs are met in the event of system failures. However, as case #0 indicates with 'M+' in Table 6-9, they do improve mission quality when all critical systems are operational. Due to navigation (C-06) incorporating other vehicle position data (C-11), tours with fewer potential conflicts can be generated, improving how many POIs can be visited (MO-4), as well as assurances that tours complete within TD-Max (MO-3).

For case #4, the cells with 'P*' in Table 6-9 indicate that, while those capabilities are only partially available in the AR-3 iteration, this is because what is lost was also new to AR-3. With the failure of S-03, navigation (C-06), and consequently touring (C-02), revert back to their pre-AR-3 capability levels, making case #4 no worse than previous AR iterations. Table 6-10 also shows a "sub-optimal touring" concern, which is purely academic because the mitigation for an S-03 failure is RC-2, i.e., returning to the pier.

Table 6-9 MAAUV AR-3 System-Failure Impacts

	System Failures											Capability Availability											Mission Objectives Met			
Case #:	S-01	S-02	S-03	S-04	S-05	S-06	S-07	S-08	S-09	S-10	S-11	C-01	C-02	C-03	C-04	C-05	C-06	C-07	C-08	C-09	C-10	C-11	MO-1	MO-2	MO-3	MO-4
0	X	X	O	O	O	O	O	O	O	O	O	A	A	A	A	A	A	A	A	A	A	A	M	M	M+	M+
4	X	O	F	O	O	O	O	O	O	O	O	P	P*	A	P	A	P*	P	A	A	A	U	P	M	M	M
6	X	F	F	X	X	X	X	X	X	X	X	X	U	X	X	X	U	U	X	U	X	U	X	U	U	U
8	X	X	X	F	X	X	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
16	X	X	X	X	F	X	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
32	X	X	X	X	X	F	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
64	X	X	X	X	X	X	F	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
128	X	X	X	X	X	X	X	F	X	X	X	U	U	X	X	U	X	X	X	X	X	X	U	U	U	U
256	X	X	X	X	X	X	X	X	F	X	X	X	U	X	X	X	U	X	X	X	U	X	X	U	U	U
512	X	X	X	X	X	X	X	X	X	F	X	U	U	X	X	U	U	X	X	X	X	X	U	U	U	U
1024	X	X	X	X	X	X	X	X	X	X	F	U	X	X	U	X	X	X	U	X	X	X	U	X	X	X

Table 6-10 MAAUV AR-3 Concerns, Remediation & Mitigation

Case #:	Concern	Remediation; Mitigation
0	NONE: Non-critical system failures	N/A; N/A
4	No indirect vehicle conflict detection; sub-optimal touring	Reliability Eng.; RC-2
6	Mission-critical capability (C-09) unavailable	Reliability Eng.; RC-1
8	Mission-critical system failure	Reliability Eng.; RC-1
16	Mission-critical system failure	Reliability Eng.; RC-1
32	Mission-critical system failure	Reliability Eng.; RC-1
64	Mission-critical system failure	Reliability Eng.; RC-1
128	Mission-critical system failure	Reliability Eng.; RC-1
256	Mission-critical system failure	Reliability Eng.; RC-1
512	Mission-critical system failure	Reliability Eng.; RC-1
1024	Mission-critical system failure	Reliability Eng.; RC-1

Having exhausted all the assurance improvement opportunities practical for existing MAAUV systems and the architecture, the only way to improve mission quality/assurances further is to add systems. This is discussed in the next AR iteration.

6.6 Assurance-Refinement Iteration #4 (AR-4)

6.6.1 AR-4 Supporting Capabilities and Systems

With no remaining practical remediations or mitigations with the current set of systems and existing architecture, AR-4 adds a Harbor Control – Touring (HC-T) system, and associated conflict prevention capability, as shown in Table 6-11 and Table 6-12, respectively.

HC-T (S-12) provides a sort of air traffic control (ATC) service for the MAAUV use case, extending AR-3's use of other-vehicle data in tour planning. HC-T is not dependent on HC-N (S-03), because the systems represent individual receivers for each service, not the services provided by the receivers. HC-T uses its knowledge of the positions of all AUVs to generate optimal tour plans for them, deconflicting at a cross-AUV level. As long as all the AUVs follow the tour plans HC-T provides, the absence of conflicts is virtually guaranteed, leading to the conflict prevention capability (C-12) shown in Table 6-12.

Table 6-11 MAAUV Systems Entering AR-4

ID	System
S-01	Onboard Clock
S-02	Inertial Navigation System (INS)
S-03	Harbor Control – Navigation (HC-N)
S-04	Propeller
S-05	Main Drive
S-06	Batteries
S-07	Rudder
S-08	Diving Planes
S-09	Electronic Chart Database
S-10	Autonomy Engine / Resilience Module
S-11	Sonar
S-12	Harbor Control – Touring (HC-T)

Table 6-12 MAAUV Capabilities Entering AR-4

ID	Capability
C-01	Conflict Avoidance
C-02	Touring
C-03	Timing
C-04	Conflict Detection
C-05	Mobility
C-06	Navigation
C-07	Indirect Conflict Detection
C-08	Direct Conflict Detection
C-09	Own Position and Orientation
C-10	Terrain and Landmark Information
C-11	Vehicle Positions and Orientations
C-12	Conflict Prevention

6.6.2 AR-4 Combined Mission Objective/Capability/System Relationships

The AR-4 combined MO/capability/system relationships are shown in Figure 6-4, with the addition of S-12, C-12, and an arrow to C-06, showing that HC-T supports navigation. The added direct and indirect support dependencies are shown in Table 6-13.

6.6.3 AR-4 System-Failure Impacts, Concerns, Remediation & Mitigation

The AR-4 system-failure impact analysis results, shown in Table 6-14, add two failure cases associated with the new HC-T system. Table 6-15 shows changed concern and elimination of mitigation in failure case #4, and that the mitigations for the two new cases are superior to just failing the mission outright.

With the addition of HC-T, Table 6-14 shows that failure case #0 now exceeds prior levels of maintaining separation (MO-1), while retaining the improved touring from AR-3 changes. The use of 'P*' is removed from failure case #4, because the failure of just S-03 no longer impacts navigation (C-06) or touring (C-02). Moreover, because C-12 all but guarantees a lack of conflicts, failure case #4 no longer needs mitigation, allowing a tour to continue (see Table 6-15).

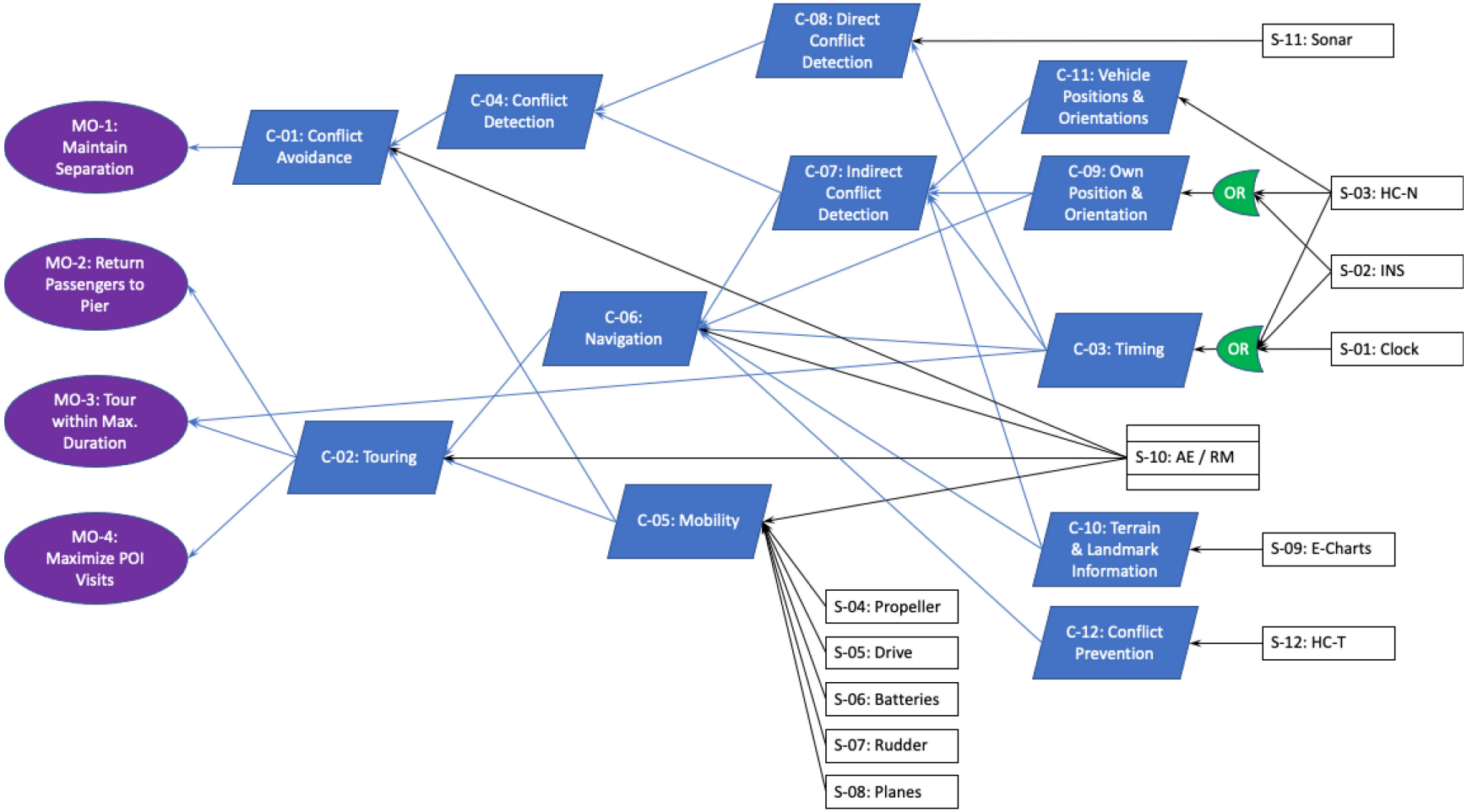


Figure 6-4 MAAUV AR-4 combined MO, capability, and system relationships

Table 6-13 MAAUV AR-4 Combined MO, Capability, and System Relationships

	Supporting Capabilities											
MO/C	C-01	C-02	C-03	C-04	C-05	C-06	C-07	C-08	C-09	C-10	C-11	C-12
MO-1	D		I	I	I		I	I	I	I	I	
MO-2		D	I		I	I	I		I	I	I	I
MO-3		D	D		I	I	I		I	I	I	I
MO-4		D	I		I	I	I		I	I	I	I
C-01			I	D	D		I	I	I	I	I	
C-02			I		D	D	I		I	I	I	I
C-03												
C-04			I				D	D	I	I	I	
C-05												
C-06			D				D		D	D	I	D
C-07			D						D	D	D	
C-08			D									
C-09												
C-10												
C-11												
C-12												

	Supporting Systems											
Capability	S-01	S-02	S-03	S-04	S-05	S-06	S-07	S-08	S-09	S-10	S-11	S-12
C-01	I	I	I	I	I	I	I	I	I	D	I	
C-02	I	I	I	I	I	I	I	I	I	D		I
C-03	D	D	D									
C-04	I	I	I						I		I	
C-05				D	D	D	D	D		D		
C-06	I	I	I						I	D		I
C-07	I	I	I						I			
C-08	I	I	I								D	
C-09		D	D									
C-10									D			
C-11			D									
C-12												D

The new cases cover the failure of S-12 and S-03. If both systems fail (case #2052), the situation reverts to failure case #4 in AR-3 (see Table 6-10), i.e., the MAAUV executes RC-2 and returns to the pier. If only S-12 fails (case #2048), the situation reverts to failure case #0 in AR-3. Though this is a failure case in AR-4, because of the lack of an S-12 in AR-3, it's the case where all non-critical systems are operational in the latter iteration. As such, a tour can continue.

The highlighted cells in Table 6-15 show where reliability engineering can be applied to improve assurance before the next AR iteration. This example analysis ends after four iterations, but could continue until no further practical system and architecture changes are identified.

Table 6-14 MAAUV AR-4 System-Failure Impacts

	System Failures												Capability Availability												Mission Objectives Met			
Case #:	S-01	S-02	S-03	S-04	S-05	S-06	S-07	S-08	S-09	S-10	S-11	S-12	C-01	C-02	C-03	C-04	C-05	C-06	C-07	C-08	C-09	C-10	C-11	C-12	MO-1	MO-2	MO-3	MO-4
0	X	X	O	O	O	O	O	O	O	O	O	O	A	A	A	A	A	A	A	A	A	A	A	A	M+	M	M+	M+
4	X	O	F	O	O	O	O	O	O	O	O	O	P	A	A	P	A	A	P	A	A	A	U	A	P	M	M+	M+
6	X	F	F	X	X	X	X	X	X	X	X	X	X	U	X	X	X	U	U	X	U	X	U	X	X	U	U	U
8	X	X	X	F	X	X	X	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	X	U	U	U	U
16	X	X	X	X	F	X	X	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	X	U	U	U	U
32	X	X	X	X	X	F	X	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	X	U	U	U	U
64	X	X	X	X	X	X	F	X	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	X	U	U	U	U
128	X	X	X	X	X	X	X	F	X	X	X	X	U	U	X	X	U	X	X	X	X	X	X	X	U	U	U	U
256	X	X	X	X	X	X	X	X	F	X	X	X	X	U	X	X	X	U	X	X	X	U	X	X	X	U	U	U
512	X	X	X	X	X	X	X	X	X	F	X	X	U	U	X	X	U	U	X	X	X	X	X	X	U	U	U	U
1024	X	X	X	X	X	X	X	X	X	X	F	X	U	X	X	U	X	X	X	U	X	X	X	X	U	X	X	X
2048	X	X	O	O	O	O	O	O	O	O	O	F	A	P*	A	A	A	P*	A	A	A	A	A	U	M	M	M+	M+
2052	X	O	F	O	O	O	O	O	O	O	O	F	P	P*	A	P	A	P*	P	A	A	A	U	U	P	M	M	M

Table 6-15 MAAUV AR-4 Concerns, Remediation & Mitigation

Case #:	Concern	Remediation; Mitigation
0	NONE: Non-critical system failures	N/A; N/A
4	Conflict prevention, but no indirect vehicle conflict detection	Reliability Eng.; N/A
6	Mission-critical capability (C-09) unavailable	Reliability Eng.; RC-1
8	Mission-critical system failure	Reliability Eng.; RC-1
16	Mission-critical system failure	Reliability Eng.; RC-1
32	Mission-critical system failure	Reliability Eng.; RC-1
64	Mission-critical system failure	Reliability Eng.; RC-1
128	Mission-critical system failure	Reliability Eng.; RC-1
256	Mission-critical system failure	Reliability Eng.; RC-1
512	Mission-critical system failure	Reliability Eng.; RC-1
1024	Mission-critical system failure	Reliability Eng.; RC-1
2048	Sub-optimal touring	Reliability Eng.; N/A
2052	No indirect vehicle conflict detection; sub-optimal touring	Reliability Eng.; RC-2

6.7 Discussion

The IMARA methodology is meant to serve as a tool for guiding the engineering process to improve mission assurance by identifying the systems required to meet MOs and reinforcing the capabilities they provide. Potential approaches include increasing system quality, adding redundant systems that provide the same capabilities, or adding systems that provide new capabilities to reduce mission risk and make the prior systems less mission critical. IMARA's purpose is not to produce a proven design on its own, like a formal method might. Instead, IMARA provides a framework for analysis. The amount of rigor in the details of that analysis is left up to the methodology's user.

In addition to the MAAUV use case in this chapter, IMARA can be applied to a number of other general mission contexts, whether they be water, air, space, or land based. Some of these missions, and their MOs are shown in Table 6-16. Minimum-Required MOs (MRMOs) are in red to highlight what AV and RM designers should address first. As the table suggests, the design and implementation of our RM from Chapter 3 can be readily adapted to these other mission contexts, provided thorough system-level analysis is performed to assure:

- Sufficient sensor inputs are available for the RM to detect threats to safety or mission with enough time to react
- Sufficient actuator control is given the RM to override AE inaction or mis-action and recover or safely abort the mission, as necessary
- RM recover functions are well-defined, as straightforward as possible, and their implementation vetted to the maximum level the state of the art allows

Table 6-16 Mission objectives similar to MAAUV's

MAAUV (AUV Tourism)	Other Mission Contexts(1)				
Mission Objectives (MOs)(2)	**Exploration/Inspection**	**Military**	**Package Delivery**	**Public Transit**	**Taxi Service**
Prevent loss of separation from fixed objects, forbidden areas & other vessels	*Same as MAAUV*	*Same as MAAUV*	*Same as MAAUV*	*Same as MAAUV*	*Same as MAAUV*
Return tourists to a pier	Return to a dispatch site	Exit enemy territory or self-destruct; return to friendly base	Return to a shipping facility	Pickup next passenger(s), or return to a dispatch site	Pickup next passenger(s), or return to a dispatch site
Return tourists to the departure pier	Return to a specific dispatch site	Return to a specific base	Return to a specific shipping facility	Return to a specific dispatch site	Return to a specific dispatch site
Conduct tour within min/max duration limits	Explore/Inspect according to a preprogrammed timetable	Deliver ordnance within battle plan timetable	Deliver package(s) no later than promised time of day	Visit stations within posted schedule	Deliver all passengers within guaranteed times
Achieve specific vessel speed(s) between POIs			Comply with speed limits and other traffic regulations	Comply with speed limits and other traffic regulations	Comply with speed limits and other traffic regulations
Loiter for specific duration(s) at individual POIs	Loiter as needed to gather data	Loiter as needed for targeting	Wait for delivery confirmation from recipients	Wait at stations per posted schedule	Loiter at pickup locations as needed waiting for passengers
Manage fuel, oil, battery, air and other consumables efficiently	Manage consumables to allow travel to refueling station	Manage consumables to allow return to a friendly base	Manage consumables to allow travel to refueling station	Manage consumables to allow travel to refueling station when off duty	Manage consumables to allow travel to refueling station between fares
Avoid maneuvers that can damage the vehicle or distress/injure tourists(3)	Avoid maneuvers that can damage the vehicle or payload(3)	Avoid maneuvers that can damage the vehicle or ordnance(3)	Avoid maneuvers that can damage the vehicle or packages(3)	Avoid maneuvers that can damage the vehicle or distress/injure passengers(3)	Avoid maneuvers that can damage the vehicle or distress/injure passengers(3)
Maximize total LOI score	Maximize total value of sites explored/inspected	Maximize total value of enemy assets attacked(4)	Maximize total value of packages delivered(4)		Deliver passengers via shortest/fastest/cheapest paths
Visit a specific number of POIs	Explore/Inspect a specific number of sites	Attack a specific number of targets	Deliver a specific number of packages		
Visit POIs in a specific order	Explore/Inspect sites in a specific order	Attack targets in a specific order	Deliver packages in a specific order		Deliver passengers in a specific order
Visit specific POIs	Explore/Inspect specific sites	Attack specific targets	Deliver specific packages		

Notes:

1) These missions are not limited to the underwater domain; surface vessels, air-, space- and land-vehicle applications are also possible

2) Minimum-required mission objectives (MRMOs) are shown in RED

3) Whether this is a MRMO depends on vehicle capabilities, payload fragility, crowdedness of operating environment, etc.; It is not a MAAUV MRMO due to limited vehicle performance capabilities

4) This MO devolves to "Deliver payload" if all onboard ordnance, packages, or passengers are going to the same place

6.8 Conclusion

The MAAUV example in this chapter demonstrates the IMARA methodology's utility as a structured mission-assurance process that, through iterative analysis of system and architecture choices, maximizes assurance within the constraints of the mission objectives and systems involved. Iterating through all of a possibly huge number of failure cases, collapsing equivalent ones using don't-care values for system operational states that are irrelevant in the presence of other more-critical system failures, assures complete coverage while producing results understandable by humans. In addition, IMARA can be applied to the many mission contexts similar to MAAUV, broadening its utility beyond our one specific scenario.

7 SUMMARY

In this dissertation we have investigated many aspects of safety and mission assurance in autonomous underwater vehicles. We examined AUV safety and mission-assurance issues and introduced the MAAUV mission and vehicle, the DESCRETE testbed, and proposed scoring criteria to measure improvements. We researched and introduced the resilience-module concept for monitoring mission progress, preventing execution of commands from faulty or compromised control systems, and recovering when safety or mission limits are exceeded. We examined the use of simulated experimentation in mission-assurance engineering, and extended resilience-module capabilities for reacting to safety- or mission-threatening situations. We examined system-level safety and security methodologies and introduced the proposed IMARA mission-level methodology for identifying mission objectives, the systems required to meet them, and the impacts of failures in those systems. Lastly, we examined the MAAUV vehicle using the IMARA methodology and introduced extensions to the vehicle to assure mission success.

The DESCRETE testbed can simulate marine safety and mission-assurance scenarios with detailed measurement hard to obtain with real vehicles. The resilience-module concept can be extended to where we imagine an industry arising for mission assurance much like the TPM industry arose for computer security over a decade ago. With the DESCRETE testbed, the resilience-module concept, and the mission/domain-agnostic IMARA methodology to build on, we have introduced the underpinnings for a formalized AUV mission-assurance discipline, and enabled the extension of that discipline from the marine domain to air, land, and space.

8 REFERENCES

1. Abrial, Jean-Raymond. *Modeling in Event-B System and Software Engineering*. Cambridge: Cambridge University Press, 2010.

2. "Aeronautical Information Manual - AIM - ATC Clearances and Aircraft Separation." Federal Aviation Administration, n.d. https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap4_section_4.html.

3. Arrival procedures. Accessed November 9, 2021. https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap5_section_4.html.

4. Asaadi, Erfan, Ewen Denney, Jonathan Menzies, Ganesh J. Pai, and Dimo Petroff. "Dynamic Assurance Cases: A Pathway to Trusted Autonomy." *Computer* 53, no. 12 (2020): 35–46. https://doi.org/10.1109/mc.2020.3022030.

5. "BIOS and Kernel Developer's Guide (BKDG) for AMD Family ..." Accessed November 14, 2021. https://www.amd.com/system/files/TechDocs/52740_16h_Models_30h-3Fh_BKDG.pdf.

6. Bittner, Kurt, and Ian Spence. *Use Case Modeling*. Boston, MA: Addison-Wesley, 2002.

7. "Boat Worx Launches New Nemo Submarine - U-Boat Worx." U, April 17, 2020. https://www.uboatworx.com/news/u-boat-worx-launches-new-nemo-submarine.

8. Bogost, Ian. "Programmers: Stop Calling Yourselves Engineers." The Atlantic. Atlantic Media Company, September 27, 2018. https://www.theatlantic.com/technology/archive/2015/11/programmers-should-not-call-themselves-engineers/414271/.

9. Buna, Samer. "Software Engineering Is Different from Programming." Medium. jsComplete EdgeCoders, February 25, 2019. https://medium.com/edge-coders/software-engineering-is-different-from-programming-b108c135af26.

10. Calinescu, Radu, Danny Weyns, Simos Gerasimou, Muhammad Usman Iftikhar, Ibrahim Habli, and Tim Kelly. "Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases." *IEEE Transactions on Software Engineering* 44, no. 11 (November 9, 2018): 1039–69. https://doi.org/10.1109/tse.2017.2738640.

11. "Cameo Systems Modeler." CATIA - Dassault Systèmes®, 2021. https://www.3ds.com/products-services/catia/products/no-magic/cameo-systems-modeler/.

12. Carter, Bryan T., Georgios Bakirtzis, Carl R. Elks, and Cody H. Fleming. "A Systems Approach for Eliciting Mission-Centric Security Requirements." *2018 Annual IEEE International Systems Conference (SysCon)*, 2018. https://doi.org/10.1109/syscon.2018.8369539.

13. Carter, Bryan, Stephen Adams, Georgios Bakirtzis, Tim Sherburne, Peter Beling, Barry Horowitz, and Cody Fleming. "A Preliminary Design-Phase Security Methodology for Cyber–Physical Systems." *Systems* 7, no. 2 (2019): 21. https://doi.org/10.3390/systems7020021.

14. Castelvecchi, Davide. “Can We Open the Black Box of Ai?” *Nature* 538, no. 7623 (2016): 20–23. https://doi.org/10.1038/538020a.

15. Chitta, Sachin. “MoveIt!: An Introduction.” *Studies in Computational Intelligence Robot Operating System (ROS)*, 2016, 3–27. https://doi.org/10.1007/978-3-319-26054-9_1.

16. Cohn, Mike. *User Stories Applied: For Agile Software Development*. Boston, MA: Addison-Wesley, 2004.

17. “Comet's Tale.” Smithsonian.com. Smithsonian Institution, June 1, 2002. https://www.smithsonianmag.com/history/comets-tale-63573615/.

18. Consiglio, Maria, Cesar Munoz, George Hagen, Anthony Narkawicz, and Swee Balachandran. “ICAROUS: Integrated Configurable Algorithms for Reliable Operations of Unmanned Systems.” *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016. https://doi.org/10.1109/dasc.2016.7778033.

19. “Cyber-Physical Systems (CPS).” Cyber-Physical Systems (CPS) (nsf10515), n.d. https://www.nsf.gov/pubs/2010/nsf10515/nsf10515.htm.

20. Delligatti, Lenny, Rick Steiner, and Richard Soley. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Upper Saddle River, NJ: Addison-Wesley, 2014.

21. Dghaym, Dana, Stephen Turnock, Michael Butler, Jon Downes, Thai Son Hoang, and Ben Pritchard. “Developing a Framework for Trustworthy Autonomous Maritime Systems.” *Proceedings of the International Seminar on Safety and Security of Autonomous Vessels (ISSAV) and European STAMP Workshop and Conference (ESWC) 2019*, 2020, 73–82. https://doi.org/10.2478/9788395669606-007.

22. Dghaym, Dana, Thai Son Hoang, Stephen R. Turnock, Michael Butler, Jon Downes, and Ben Pritchard. “An STPA-Based Formal Composition Framework for Trustworthy Autonomous Maritime Systems.” *Safety Science* 136 (2021): 105139. https://doi.org/10.1016/j.ssci.2020.105139.

23. Dill, Evan T., Steven D. Young, and Kelly J. Hayhurst. “SAFEGUARD: An Assured Safety Net Technology for UAS.” *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016. https://doi.org/10.1109/dasc.2016.7778009.

24. D'Silva, Vijay, Daniel Kroening, and Georg Weissenbacher. “A Survey of Automated Techniques for Formal Software Verification.” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27, no. 7 (2008): 1165–78. https://doi.org/10.1109/tcad.2008.923410.

25. “Ethics, Safety, And Software Behind Self-Driving Cars In The Aftermath Of First Pedestrian Killed: IEEE Computer Society.” IEEE Computer Society Ethics Safety and Software Behind Self Driving Cars in The Aftermath of First Pedestrian Killed Comments. Accessed January 12, 2020. https://www.computer.org/publications/tech-news/trends/ethics-safety-and-software-behind-self-driving-cars-in-the-aftermath-of-first-pedestrian-killed.

26. Extensible markup language (XML) 1.0 (fifth edition). Accessed November 7, 2021. https://www.w3.org/TR/xml/.

27. Flammini, Francesco, Stefano Marrone, Roberto Nardone, Mauro Caporuscio, and Mirko D'Angelo. "Safety Integrity through Self-Adaptation for Multi-Sensor Event Detection: Methodology and Case-Study." *Future Generation Computer Systems* 112 (November 2020): 965–81. https://doi.org/10.1016/j.future.2020.06.036.

28. Fleming, Cody H, Carl Elks, Georgios Bakirtzis, Stephen Adams, Bryan Carter, Peter Beling, and Barry Horowitz. "Cyberphysical Security through Resiliency: A Systems-Centric Approach." *Computer* 54, no. 6 (2021): 36–45. https://doi.org/10.1109/mc.2020.3039491.

29. Friedberg, Ivo, Kieran Mclaughlin, Paul Smith, David Laverty, and Sakir Sezer. "STPA-SafeSec: Safety and Security Analysis for Cyber-Physical Systems." *Journal of Information Security and Applications* 34 (2017): 183–96. https://doi.org/10.1016/j.jisa.2016.05.008.

30. Garces, Lina, Silverio Martinez-Fernandez, Valdemar Vicente Graciano Neto, and Elisa Yumi Nakagawa. "Architectural Solutions for Self-Adaptive Systems." *Computer* 53, no. 12 (December 2020): 47–59. https://doi.org/10.1109/mc.2020.3017574.

31. Gopal, Kanishka, Sneha Jadoo, Jyotsna L.P. Ramgoolam, and Vimla Devi Ramdoo. "Software Quality Problems in Requirement Engineering and Proposed Solutions for an Organization in Mauritius." *International Journal of Computer Applications* 137, no. 2 (2016): 23–31. https://doi.org/10.5120/ijca2016908698.

32. Group, ECA. "Meet Us at @Underwater Intervention in New Orleans 23-25/02. Discover Our #Robotics Offer #ROVs #Auvs Booth 819 Pic.twitter.com/kptgnkeot4." Twitter. Twitter, February 21, 2016. https://twitter.com/eca_group/status/701515288123219968.

33. "Home." Unmanned Underwater Vehicle Simulator Documentation, n.d. https://uuvsimulator.github.io/.

34. "IEEE P7001 - IEEE Draft Standard for Transparency of Autonomous Systems." IEEE SA - The IEEE Standards Association - Home. Accessed December 31, 2020. https://standards.ieee.org/project/7001.html.

35. "Imara: English Translation." bab.la. Accessed October 20, 2021. https://en.bab.la/dictionary/swahili-english/imara.

36. "ISO/IEC 11889-1:2015." ISO, March 1, 2016. https://www.iso.org/standard/66510.html.

37. *ISO/IEC/IEEE 12207:2017(E) First Edition 2017-11: ISO/IEC/IEEE International Standard - Systems and Software Engineering -- Software Life Cycle Processes*. IEEE, 2017.

38. "JP 5-0, Joint Planning," December 1, 2020. https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp5_0.pdf.

39. Kabir, Sohag, Ioannis Sorokos, Koorosh Aslansefat, Yiannis Papadopoulos, Youcef Gheraibia, Jan Reich, Merve Saimler, and Ran Wei. *A Runtime Safety Analysis Concept for Open Adaptive Systems*, October 2019, 332–46. https://doi.org/10.1007/978-3-030-32872-6_22.

40. Kavallieratos, Georgios, Sokratis Katsikas, and Vasileios Gkioulos. “Cybersecurity and Safety Co-Engineering of Cyberphysical Systems—a Comprehensive Survey.” *Future Internet* 12, no. 4 (2020): 65. https://doi.org/10.3390/fi12040065.

41. Kotwicki, Jon. “Required Reports When Flying IFR.” FLY8MA Online Flight Training, January 12, 2021. https://fly8ma.com/required-reports-when-flying-ifr/.

42. Lemos, Rogério De, David Garlan, Carlo Ghezzi, Holger Giese, Jesper Andersson, Marin Litoiu, Bradley Schmerl, et al. *Software Engineering for Self-Adaptive Systems: Research Challenges in the Provision of Assurances*, 2017, 3–30. https://doi.org/10.1007/978-3-319-74183-3_1.

43. Leveson, Nancy. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: The MIT Press, 2011.

44. Lusk, Parker C., Patricia C. Glaab, Louis J. Glaab, and Randal W. Beard. “Safe2Ditch: Emergency Landing for Small Unmanned Aircraft Systems.” *Journal of Aerospace Information Systems* 16, no. 8 (2019): 327–39. https://doi.org/10.2514/1.i010706.

45. Matteson, Scott. “Autonomous versus Automated: What Each Means and Why It Matters.” TechRepublic. TechRepublic, June 7, 2019. https://www.techrepublic.com/article/autonomous-versus-automated-what-each-means-and-why-it-matters/.

46. Mello, John P. “7 Bad Habits of Highly Ineffective Software Engineers.” TechBeacon. TechBeacon, January 22, 2019. https://techbeacon.com/app-dev-testing/7-bad-habits-highly-ineffective-software-engineers.

47. “Microsoft.” Microsoft Support. Accessed November 14, 2021. https://support.microsoft.com/en-us/windows/enable-tpm-2-0-on-your-pc-1fd5a332-360d-4f46-a1e7-ae6b0c90645c.

48. Munoz, Cesar, Anthony Narkawicz, George Hagen, Jason Upchuch, Aaron Dutle, Maria Consiglio, and James Chamberlain. “DAIDALUS: Detect and Avoid Alerting Logic for Unmanned Systems.” *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015. https://doi.org/10.1109/dasc.2015.7311588.

49. “Navigation Rules - Navcen.uscg.gov.” Accessed October 27, 2021. https://www.navcen.uscg.gov/pdf/navRules/Handbook/NavRulesAmalgamatedwAnnexes.pdf.

50. “Nemo Submarine.” NEMO Submarine, October 19, 2021. https://nemo-submarine.com/.

51. “New Microarchitecture for 4th Gen Intel® Core™ Processor ...” Accessed November 14, 2021. https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf.

52. Nicole.keller@nist.gov. “The Five Functions.” NIST, August 10, 2018. https://www.nist.gov/cyberframework/online-learning/five-functions.

53. “The Official YAML Web Site.” The Official YAML Web Site. Accessed November 14, 2021. https://yaml.org/.

54. Oreizy, P., M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf. “An Architecture-Based Approach to Self-Adaptive Software.” *IEEE Intelligent Systems* 14, no. 3 (1999): 54–62. https://doi.org/10.1109/5254.769885.

55. Parnas, D.l. “Software Engineering Programs Are Not Computer Science Programs.” *IEEE Software* 16, no. 6 (1999): 19–30. https://doi.org/10.1109/52.805469.

56. Press. “NASA Starts Using FLARM for Drone UTM.” sUAS News - The Business of Drones, November 29, 2019. https://www.suasnews.com/2019/11/nasa-starts-using-flarm-for-drone-utm/.

57. “Products.” Nuytco Research Ltd. Accessed October 2, 2021. https://nuytco.com/products/deepworker/.

58. Rudin, Cynthia. “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead.” *Nature Machine Intelligence* 1, no. 5 (2019): 206–15. https://doi.org/10.1038/s42256-019-0048-x.

59. “Sans Institute.” What You Need to Know About the SolarWinds Supply-Chain Attack | SANS Institute, December 15, 2020. https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/.

60. Siil, Karl, Aviel Rubin, Matthew Elder, Anton Dahbura, Matthew Green, and Lanier Watkins. “Cost-Effective Mission Assurance Engineering Through Simulation.” *2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS) (IoTaIS'2021)*, November 22, 2021.

61. Siil, Karl, Aviel Rubin, Matthew Elder, Anton Dahbura, Matthew Green, and Lanier Watkins. “Mission Assurance for Autonomous Undersea Vehicles.” *2020 IEEE Security and Privacy Workshops (SPW)*, 2020. https://doi.org/10.1109/spw50608.2020.00056.

62. Siil, Karl, Aviel Rubin, Matthew Elder, Anton Dahbura, Matthew Green, and Lanier Watkins. Ms. *A Methodology for Iterative Mission-Assurance Refinement Analysis*. Johns Hopkins University, n.d.

63. “A Snapshot of Engineering for Resilience at APL.” JHU/APL Technical Digest, July 2019. https://www.jhuapl.edu/TechDigest/Detail?Journal=J&VolumeID=34&IssueID=4.

64. Spafford, Eugene H. “Quotable Spaf.” Gene Spafford's Personal Pages: Quotable Spaf, 2019. https://spaf.cerias.purdue.edu/quotes.html.

65. “Static Code Analysis for C, C++, C#, Java, and JavaScript.” Perforce. Accessed January 20, 2020. https://www.perforce.com/products/klocwork.

66. Stevens, Mia N., and Ella M. Atkins. “Multi-Mode Guidance for an Independent Multicopter Geofencing System.” *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016. https://doi.org/10.2514/6.2016-3150.

67. “Submarine Technology Symposium Pushes to Expand the Competitive Space from Undersea.” JHU/APL Brand, June 21, 2019. https://www.jhuapl.edu/PressRelease/190621.

68. Submarines and ROVs for sale and hire by Silvercrest Submarines, n.d. https://www.silvercrestsubmarines.co.uk/mediumtouristsubinfo.html.

69. “Submarines for Sale- Buy Personal New or Used Subs, See Price & Specs.” MySubmarines, October 21, 2021. https://www.mysubmarines.com/.

70. “Top Five Causes of Poor Software Quality.” Datamation. Accessed January 16, 2020. https://www.datamation.com/entdev/article.php/3827841/Top-Five-Causes-of-Poor-Software-Quality.htm.

71. “TPM: Certified: Products: Trusted: Computing: Group.” Trusted Computing Group, July 30, 2019. https://trustedcomputinggroup.org/membership/certification/tpm-certified-products/.

72. “What Is SDFormat.” SDFormat. Accessed November 7, 2021. http://sdformat.org/.

73. “Why Gazebo?” gazebo. Accessed January 18, 2021. http://gazebosim.org/.

74. Xanthidis, Marios, Nare Karapetyan, Hunter Damron, Sharmin Rahman, James Johnson, Jason M. O'Kane, and JaIoannisson Rekleitis. “Navigation in the Presence of Obstacles for an Agile Autonomous Underwater Vehicle.” arXiv.org, March 28, 2019. https://arxiv.org/abs/1903.11750.

75. Xiao, Yang, Guoqi Li, and Yuchao Zhang. “A Rule-Based Safety Kernel for Unmanned System.” INFONA. Accessed January 12, 2020. http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ieee-000006321097.

76. Young, William, and Nancy Leveson. “Systems Thinking for Safety and Security.” *Proceedings of the 29th Annual Computer Security Applications Conference*, December 9, 2013. https://doi.org/10.1145/2523649.2530277.

77. Zhou, Xiang-Yu, Zheng-Jiang Liu, Feng-Wu Wang, and Zhao-Lin Wu. “A System-Theoretic Approach to Safety and Security Co-Analysis of Autonomous Ships.” *Ocean Engineering* 222 (2021): 108569. https://doi.org/10.1016/j.oceaneng.2021.108569.

Appendix A. DESCRETE Testbed Functionality Details

The Digital Environment for Simulated Cyber Resilience Engineering, Test and Experimentation (DESCRETE) testbed enables the configuration of test environments and execution of test events with simulated marine vessels. DESCRETE is built on the Gazebo robot simulation environment [73] and a Gazebo-based unmanned underwater vehicle (UUV) Simulator [33] that implements underwater physics. The UUV Simulator also provides example vehicles, on one of which we based our autonomous underwater vehicle (AUV) models. Both Gazebo and the UUV Simulator are open-source software.

Each test environment can be configured with the following elements to support the touring missions detailed in Section 2.6:

- Land and marine topography
- Points of Interest (POIs) and piers
- Partner submersibles, including a mission-assured AUV (MAAUV)

We use Gazebo and the UUV Simulator to generate marine and land topography, as well as visual and inertial models for the other objects. DESCRETE implements our boundary-based system for detecting and avoiding potential collisions, as well as the mechanisms for executing and measuring tours, which are described in the sections that follow.

Testbed Objects

Vehicles, fixed objects, terrain, and other elements of a Gazebo simulation are expressed in SDF [72], which is an extension of XML [26]. Objects in DESCRETE test environments are expressed in YAML [53]. This section describes each DESCRETE object, followed by a corresponding snippet from an example configuration file. DESCRETE provides the ability to configure objects individually, which is described in this section, and generate sets of objects randomly within defined parameters, which is described in the next section. Combining manually configured and randomly generated objects is also possible and allows,

for example, the touring of many different POI layouts with the same set of AUVs operating from the same piers for each tour, which is what was done in our experimentation.

In addition to simulated objects, a DESCRETE configuration contains the following global parameters:

- A pseudo-random number generator (PRNG) seed for variety/repeatability
- The testbed's three-dimensional size with the origin in the center and on the water's surface
- The number of times to run a configuration where at least one object is randomly generated

The following is an example DESCRETE global parameter configuration:

```
#   Property    Type (Units)   Description
#   --------    ------------   -----------
#   seed        unsigned int   Seed to initialize the PRNG
#   size        double[3] (m)  XY origin in the middle, Z origin on the surface
#   num_runs    unsigned int   Number of testbed iterations to run

testbed:
  seed:     3263827
  size:     [ 5000.0, 5000.0, 80.0 ]
  num_runs: 1
```

Each manually configured POI has the following parameters:

- A name that is unique across all POIs, piers, and vehicles in the configuration
- A pose comprised of an absolute position and role/pitch/yaw (RPY) orientation
- The POI's boundaries, as described in Section 2.4 and Section 3.4.3
- A level of interest (LOI), relative to all other POI LOIs in this configuration

The following is an example DESCRETE POI-set configuration:

```
#   Property      Type (Units)       Description
#   --------      ------------       -----------
#   name          string             unique name
#   pose          double[6] (m,rad)  absolute XYZ and RPY
#   boundaries    map                boundary distances
#     physical      double (m)         crossing this is a collision
#     separation    double (m)         crossing this is a LOS
#     response      double (m)         crossing this invokes a correction
#     warning       double (m)         crossing this causes a warning
#     visit         double (m)         crossing this is a visit
#   interest      double (pts)       relative to all other POIs

poi_list:
  -
    name: POI_01
    pose: [   0.0,   2.0, -80.0, 0.0, 0.0, 0.0 ]
    boundaries:
      physical:    5.0
      separation: 10.0
      response:   15.0
      warning:    20.0
      visit:      30.0
    interest: 100.0
```

Each manually configured pier has the following parameters:

- A name that is unique across all POIs, piers, and vehicles in the configuration
- A pose comprised of an absolute position and RPY orientation
- The pier's boundaries, as described in Section 2.4 and Section 3.4.3

The following is an example DESCRETE pier-set configuration:

```
#   Property      Type (Units)       Description
#   --------      ------------       -----------
#   name          string             unique name
#   pose          double[6] (m,rad)  absolute XYZ and RPY
#   boundaries    map                boundary distances
#     physical      double (m)         crossing this is a collision
#     separation    double (m)         crossing this is a LOS
#     response      double (m)         crossing this invokes a correction
#     warning       double (m)         crossing this invokes a warning
#     docking       double (m)         crossing this is docking

pier_list:
  -
    name: Pier_Q1
    pose: [  155.0,  155.0, 0.0, 0.0, 0.0, 0.0 ]
    boundaries:
      physical:   4.5
      separation: 5.5
      response:   7.0
      warning:    8.0
      docking:    9.0
```

Each manually configured non-MAAUV submersible has the following parameters:

- A name that is unique across all POIs, piers, and vehicles in the configuration
- Whether the submersible participates in the partnership described in Section 2.2
- A starting pose comprised of an absolute position and RPY orientation
- The submersible's boundaries, as described in Section 2.4 and Section 3.4.3
- Performance parameters, comprised of turn radius and maximum speed
- Navigation behaviors, currently the same as the MAAUV's (see below), but without constraints
- Speed behaviors, currently limited to maximum speed

The following is an example DESCRETE non-MAAUV submersible-set configuration:

```
#   Property          Type (Units)         Description
#   --------          ------------         -----------
#   name              string               unique name
#   partner           boolean              participates in separation services
#   pose              double[6] (m,rad)    absolute XYZ and RPY
#   boundaries        map                  boundary distances
#     physical          double (m)           crossing this is a collision
#     separation        double (m)           crossing this is a LOS
#     response          double (m)           crossing this invokes a correction
#     warning           double (m)           crossing this invokes a warning
#   performance       map                  performance properties
#     turn_radius       double (m)           turn radius at maximum speed
#     speeds            map                  speeds through the water
#       maximum           double (m/s)         top speed
#   behaviors         map                  submersible operational behaviors
#     navigation        map                  navigate as described
#       type              string               type, chosen from:
#                                                "tour_list"    - sequential
#                                                "tour_max_loi" - max-LOI
#                                                "tour_rand"    - random
#       points            sequence             depends on "type"
#                           string               POI names
#     speed             map                  speed as described
#       type              string               type, chosen from:
#                                                "maximum" - full throttle

submersible_list:
  -
    name: "Partner_Sub_Q2"
    partner: true
    pose: [ -146.5,  146.5, 0.0, 0.0, 0.0, 0.0     ] #  90-deg heading
    boundaries:
      physical:    1.1
      separation: 10.0
      response:   15.0
      warning:    20.0
    performance:
      turn_radius: 6.0
      speeds:
        maximum: 1.5 # 3 knots
    behaviors:
      navigation:
        type: "tour_rand"
      speed:
        type: "maximum"
```

Each manually configured MAAUV submersible has the following parameters:

- A name that is unique across all POIs, piers, and vehicles in the configuration
- Whether the submersible participates in the partnership described in Section 2.2
- A starting pose comprised of an absolute position and RPY orientation
- The submersible's boundaries, as described in Section 2.4 and Section 3.4.3
- Performance parameters, comprised of turn radius, maximum speed and endurance
- Navigation behaviors, which are tours constrained by duration limits, per Section 2.2

- Speed behaviors, currently limited to maximum speed
- Faults that may occur on the tour, detailed below

The following is an example DESCRETE MAAUV submersible-set configuration:

```
#   Property           Type (Units)         Description
#   --------           ------------         -----------
#   name               string               unique name
#   partner            boolean              participates in separation services
#   pose               double[6] (m,rad)    absolute XYZ and RPY
#   boundaries         map                  boundary distances
#     physical           double (m)           crossing this is a collision
#     separation         double (m)           crossing this is a LOS
#     response           double (m)           crossing this invokes a response
#     warning            double (m)           crossing this invokes a warning
#   performance        map                  performance properties
#     turn_radius        double (m)           turn radius at maximum speed
#     speeds             map                  speeds through the water
#       maximum            double (m/s)         top speed
#     endurance          double (s)           maximum time away from the pier
#   behaviors          map                  MAAUV operational behaviors
#     navigation         map                  navigate as described
#       type               string               type, chosen from:
#                                                 "tour_list"    - sequential
#                                                 "tour_max_loi" - max-LOI
#                                                 "tour_rand"    - random
#       points             sequence             depends on "type"
#                            string               POI names
#       constraints        map                  Tour limits
#         maxTourDur         double (s)           maximum time a tour can last
#         minTourDur         double (s)           minimum time a tour must last
#     speed              map                  speed as described
#       type               string               type, chosen from:
#                                                 "maximum" - full throttle
#   faults             sequence             Autonomy-engine faults
#     control            string               faulty control, chosen from:
#                                               "heading"
#                                               "depth"
#                                               "speed"
#     deviation          map                  reference frame, extent, duration
#       frame              string               reference frame, chosen from:
#                                                 "relative"
#                                                 "absolute"
#       amounts            double[2] (*)        minimum/maximum deviation
#       durations          double[2] (s)        minimum/maximum duration
#     trigger            string               chosen from:
#                                               "simulation" - simulation start
#                                               "departure"  - DTP activation
#                                               "enroute"    - ETP activation
#                                               "tourplan"   - Any activation
#                                               "proximity"  - object range
#                                               "point"      - XYZ range
#                                               "position"   - XY range
#                                               "depth"      - depth range
#                                               "random"     - with probability
#     delay              double (s)           time after "trigger"
#     range              double (m)           maximum distance to trigger
#     object             map                  triggering object
#       type               string               object type, e.g., "POI"
#       name               string               object name
#     coordinates        double[3] (m)        absolute XYZ values
#     probability        double (prob)        probability [0,1] per update
#     recurrence         string               repetitiveness, chosen from:
#                                               "on trigger"
#                                               "one time"
```

(MAAUV submersible-set configuration example, continued)

```
maauv_list:
  -
    name: "MAAUV_Lite"
    partner: true
    pose: [  146.5,  146.5, 0.0, 0.0, 0.0, 3.14159 ] # 270-deg heading
    boundaries:
      physical:    1.1
      separation: 10.0
      response:   15.0
      warning:    20.0
    performance:
      turn_radius:  6.0
      speeds:
        maximum:    1.5 # 3 knots
      endurance: 7200.0 # Vessel can stay out two hours
    behaviors:
      navigation:
        type: "tour_max_loi"
        constraints:
          maxTourDur:  3600.0 # Limit tours to one hour
          minTourDur:     0.0 # No minimum tour time limit
      speed:
        type: "maximum"
    faults:
      -
        control: "heading"    # 50 m from a sub, right 180 deg. for 1-2 min.
        deviation:
          frame: "relative"
          amounts: [ -3.14159, -3.14159 ]
          durations: [ 60.0, 120.0 ]
        trigger: "proximity"
        range: 50.0
        object:
          type: "submersible"
      -
        control: "depth"      # disrupt depth-holding at 40 m for 30-60 s
        deviation:
          frame: "relative"
          amounts: [ -5.0, +5.0 ]
          durations: [ 30.0, 60.0 ]
        trigger: "depth"
        range: 1.0
        coordinates: [ 0.0, 0.0, -40.0 ]
      -
        control: "speed"      # go DIW for 30 s, one minute after POI #7
        deviation:
          frame: "absolute"
          amounts: [ 0.0, 0.0 ]
          durations: [ 30.0, 30.0 ]
        trigger: "proximity"
        delay: 60.0
        range: 30.0
        object:
          name: "POI_07"
```

A MAAUV submersible configuration includes the ability to describe a set of potential faults in detail, using the following parameters for each fault:

- The affected control

- The reference frame and bounds (magnitude and duration) of the deviation caused by the fault
- The fault's trigger, and how long after the trigger the fault occurs
- The triggering position/object, if any, and how far from the position/object the fault occurs
- The probability that the fault will occur, and how often

Randomly Generating Testbed Objects

DESCRETE provides the ability to randomly generate sets of the testbed objects described in the previous section except MAAUV submersibles, which is a planned feature for the future. The manually configured objects are generated first, and then the parameters below are used to generate the random objects. The order of generation is POIs, piers, non-MAAUV submersibles, and MAAUV submersibles. DESCRETE attempts to place an object a fixed number of times, but if the spacing requirements cannot be met (see below), it eventually gives up and ends the simulation with an error message.

POIs are generated randomly within the bounds of the following parameters:

- Minimum and maximum number of POIs to generate
- Prefix that guarantees namespace uniqueness, to which numeric IDs are appended
- Minimum and maximum range for each dimension of a generated position and RPY
- Minimum and maximum spacing required from other objects for initial placement
- Common boundaries for all generated POIs, as described in Section 2.4 and Section 3.4.3
- Minimum and maximum LOI values

The following is an example DESCRETE POI-generation configuration:

```
#   Property       Type (Units)          Description
#   --------       ------------          -----------
#   counts         unsigned int[2]       minimum/maximum number to generate
#   prefix         string                unique name prefix
#   placement      double[2][6] (m,rad)  minimum/maximum absolute XYZ and RPY
#   spacing        map                   minimum/maximum spacing
#     poi            double[2] (m)         min/max spacing from POIs
#     pier           double[2] (m)         min/max spacing from piers
#     submersible    double[2] (m)         min/max spacing from submersibles
#     maauv          double[2] (m)         min/max spacing from MAAUVs
#   boundaries     map                   boundaries for all POIs generated
#     physical       double (m)            crossing this is a collision
#     separation     double (m)            crossing this is a LOS
#     response       double (m)            crossing this invokes a correction
#     warning        double (m)            crossing this invokes a warning
#     visit          double (m)            crossing this is a visit
#   interest       double[2] (pts)       minimum/maximum level of interest

poi_generation:
  counts: [ 0, 10 ]
  prefix: "POI_"
  placement:
    - [ -2500.0, -2500.0, -80.0, 0.0, 0.0, 0.0 ]
    - [  2500.0,  2500.0, -80.0, 0.0, 0.0, 0.0 ]
  spacing:
    poi:         [  10.0, 5000.0 ]
    pier:        [ 200.0, 5000.0 ]
    submersible: [  50.0, 5000.0 ]
    maauv:       [  50.0, 5000.0 ]
  boundaries:
    physical:    5.0
    separation: 10.0
    response:   15.0
    warning:    20.0
    visit:      30.0
  interest: [ 1.0, 100.0 ]
```

Piers are generated randomly within the bounds of the following parameters:

- Minimum and maximum number of piers to generate
- Prefix that guarantees namespace uniqueness, to which numeric IDs are appended
- Minimum and maximum range for each dimension of a generated position and RPY
- Minimum and maximum spacing required from other objects for initial placement
- Common boundaries for all generated piers, as described in Section 2.4 and Section 3.4.3

The following is an example DESCRETE pier-generation configuration:

```
#   Property       Type (Units)          Description
#   --------       ------------          -----------
#   counts         unsigned int[2]       minimum/maximum number to generate
#   prefix         string                unique name prefix
#   placement      double[2][6] (m,rad)  minimum/maximum absolute XYZ and RPY
#   spacing        map                   minimum/maximum spacing
#     poi            double[2] (m)         min/max spacing from POIs
#     pier           double[2] (m)         min/max spacing from piers
#     submersible    double[2] (m)         min/max spacing from submersibles
#     maauv          double[2] (m)         min/max spacing from MAAUVs
#   boundaries     map                   boundaries for all piers generated
#     physical       double (m)            crossing this is a collision
#     separation     double (m)            crossing this is a LOS
#     response       double (m)            crossing this invokes a correction
#     warning        double (m)            crossing this invokes a warning
#     docking        double (m)            crossing this is docking

pier_generation:
  counts: [ 0, 0 ]
  prefix: "Pier_"
  placement:
    - [ -2500.0, -2500.0, 0.0, 0.0, 0.0, 0.0 ]
    - [  2500.0,  2500.0, 0.0, 0.0, 0.0, 0.0 ]
  spacing:
    poi:         [ 200.0, 5000.0 ]
    pier:        [ 500.0, 5000.0 ]
    submersible: [  25.0, 5000.0 ]
    maauv:       [  25.0, 5000.0 ]
  boundaries:
    physical:   4.5
    separation: 5.5
    response:   7.0
    warning:    8.0
    docking:    9.0
```

Non-MAAUV submersibles are generated randomly within the bounds of the following parameters:

- Minimum and maximum number of submersibles to generate
- Prefix that guarantees namespace uniqueness, to which numeric IDs are appended
- Common value for all generated submersibles regarding partnership participation (Section 2.2)
- Minimum and maximum range for each dimension of a generated position and RPY
- Minimum and maximum spacing required from other objects for initial placement
- Common boundaries for all generated submersibles, as described in Section 2.4 and Section 3.4.3
- Minimum and maximum performance parameters, comprised of turn radius and maximum speed
- Set of navigation behaviors from which to choose for each generated submersible
- Set of speed behaviors from which to choose, currently limited to maximum speed

The following is an example DESCRETE non-MAAUV submersible-generation configuration:

```
#   Property         Type (Units)          Description
#   --------         ------------          -----------
#   counts           unsigned int[2]       minimum/maximum number to generate
#   prefix           string                unique name prefix
#   partner          boolean               participates in separation services
#   placement        double[2][6] (m,rad)  minimum/maximum absolute XYZ and RPY
#   spacing          map                   minimum/maximum spacing
#     poi              double[2] (m)         min/max spacing from POIs
#     pier             double[2] (m)         min/max spacing from piers
#     submersible      double[2] (m)         min/max spacing from submersibles
#     maauv            double[2] (m)         min/max spacing from MAAUVs
#   boundaries       map                   boundaries for all submersibles
#     physical         double (m)            crossing this is a collision
#     separation       double (m)            crossing this is a LOS
#     response         double (m)            crossing this invokes a correction
#     warning          double (m)            crossing this invokes a warning
#   performance      map                   performance properties
#     turn_radius      double[2] (m)         min/max turn radius at max. speed
#     speeds           map                   speeds through the water
#       maximum          double[2] (m/s)       min/max top speed
#   behaviors        map                   submersible operational behaviors
#     navigation       map                   navigate as described
#       types            string[]              types to pick from randomly:
#                                                "tour_list"    - sequential
#                                                "tour_max_loi" - max-LOI
#                                                "tour_rand"    - random
#       points           sequence              depends on "type"
#                          string                POI names
#     speed            map                   speed as described
#       types            string[]              types to pick from randomly:
#                                                "maximum" - full throttle

submersible_generation:
  counts: [ 0, 7 ]
  prefix: "Partner_Submersible_"
  partner: true
  placement:
    - [ -2500.0, -2500.0, 0.0, 0.0, 0.0, -3.14159 ]
    - [  2500.0,  2500.0, 0.0, 0.0, 0.0,  3.14159 ]
  spacing:
    poi:         [  50.0, 5000.0 ]
    pier:        [  25.0, 5000.0 ]
    submersible: [ 100.0, 5000.0 ]
    maauv:       [ 100.0, 5000.0 ]
  boundaries:
    physical:    1.1
    separation: 10.0
    response:   15.0
    warning:    20.0
  performance:
    turn_radius: [ 5.0, 7.0 ]
    speeds:
      maximum: [ 1.0, 2.0 ]     # 2 to 4 kts
  behaviors:
    navigation:
      types: [ "tour_list", "tour_max_loi", "tour_rand" ]
      points: [ "POI_01", "POI_02", "POI_03", "POI_05", "POI_08" ]
    speed:
      types: [ "maximum" ]
```

Testbed Logging

DESCRETE's logs are the basis for the results analysis throughout this dissertation. There are log records for all significant testbed events, which are:

- Simulator startup, where the defining configuration file is recorded
- Simulation load/reload and object placement at the start of each test run
- Tour-plan generation and tour start
- Mission updates and maneuver orders (see details below)
- Mission success or failure at the end of a tour

Mission updates and maneuver orders constitute the vast majority of the logs. Each mission update, except for the last one, which logs mission success or failure, is followed by maneuver orders indicating how the vehicle wants to react to the situation, based on the *influencing object*. Influencing objects include POIs, piers, other vehicles, the seafloor, and the surface. Collision avoidance can be influenced by any of these objects. Transiting to and visiting POIs is influenced by POIs, and returning home and docking is influenced by piers.

Mission updates capture each vehicle's state approximately 100 times per second, logging:

- Absolute three-dimensional position, RPY, and speed
- Influencing object, along with the distance, bearing, and elevation to it
- Tour estimated time in route (ETE), which is the sum of the elapsed time since the tour started and the remaining ETE in the tour plan
- State, i.e., whether the vehicle is transiting to a POI/pier, visiting, avoiding collision, etc.

Maneuver orders capture desired heading (yaw), angle of attack (pitch), depth, and speed, along with whether the orders are from the autonomy engine (AE) or resilience module (RM). The current vehicles do not bank into turns; therefore, roll orders are only given to keep an even keel.

Planned Features

The following features are planned as future extensions of DESCRETE's mission-assurance experimentation capabilities.

The UUV Simulator for Gazebo implements basic currents that act on the vehicles. We would like to add finer-grain configuration of how currents behave during a test event, as well as forecasts for mission planning, and measure vehicle performance against the actual currents encountered.

We would like to add surface vessels and non-partner submersibles, per the mission description in Section 2.2. Surface vessels require little more than new visual and inertial models. Non-partner submersibles require adding simulated sonar for potential collision detection. Gazebo and the UUV Simulator have sonar simulations, which we can integrate into our AUV models.

We would like to augment the behaviors of our non-MAAUV vehicles to add variety and complexity to the touring scenarios. This includes random speed changes, and non-touring navigation behaviors like:

- Continuous "circuits" comprised of two or more points
- One-time "runs" of one or more points
- Continuous "patrols" of polygons defined by three or more points

To non-MAAUV submersibles, we would like to add depth-change behaviors, including periodic and random changes within depth bounds, and configurable times spent at a given depth, or on the surface.

We would like to add new types of tours, e.g., fastest, most POIs, high-value POIs first, POIs in a bounded area. We would also like to add tour parameters, such as minimum-required speed for transiting to POIs or returning to a pier, as well as minimum and maximum loiter times at POIs.

We would like to implement and experiment with the Harbor Control Touring Service (HC-T), described in Section 6.6.1.

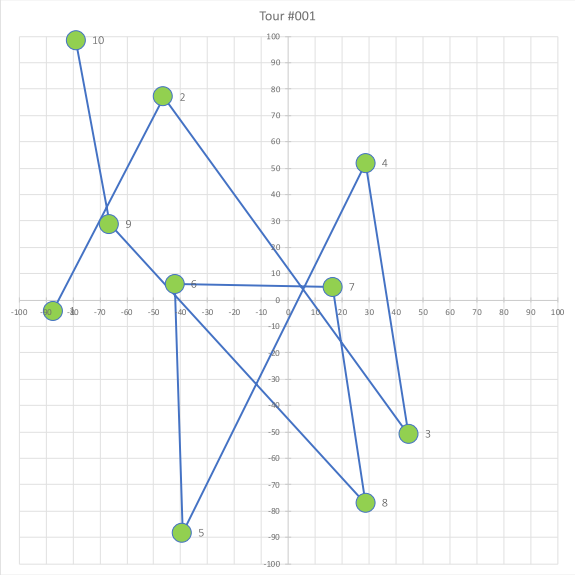
Lastly, we would like to randomly generate sets of MAAUV submersibles that interact with each other to investigate mission assurance in cooperating swarms of vehicles.

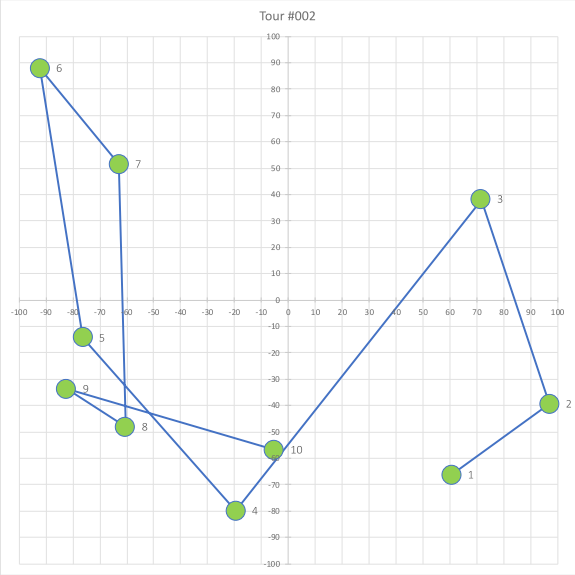
Appendix B. DESCRETE Experimentation Layouts

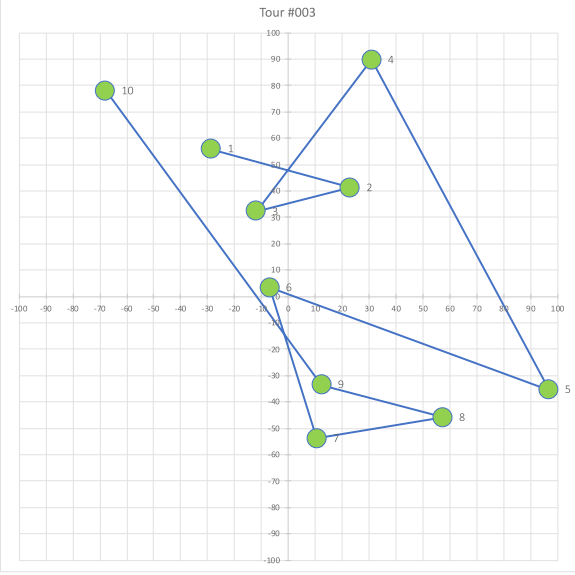
Graphical and tabular representations of the 100 point-of-interest (POI) layouts used in our experimentation are provided below. See Section 4.3.1 for more information.

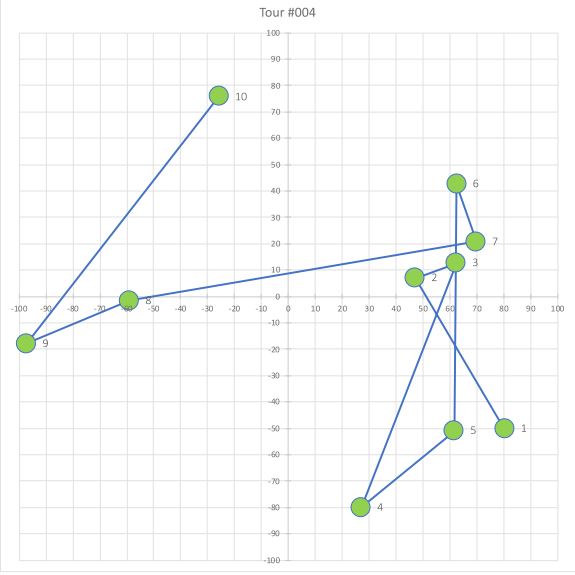
Graphical POI Layouts

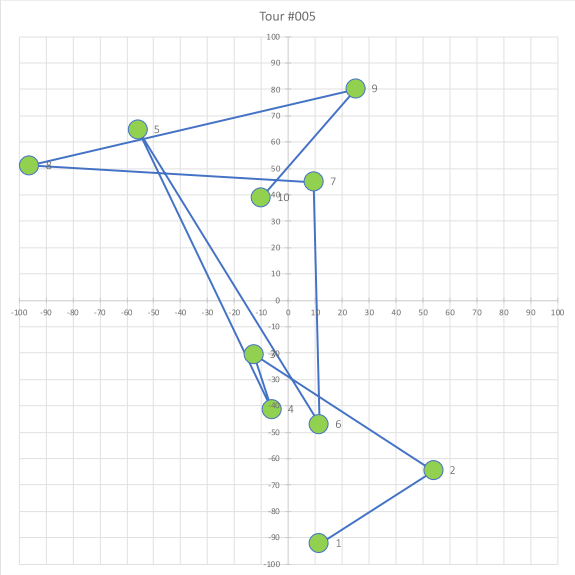
Green circles represent the POIs. Blue lines represent the path each tour followed.

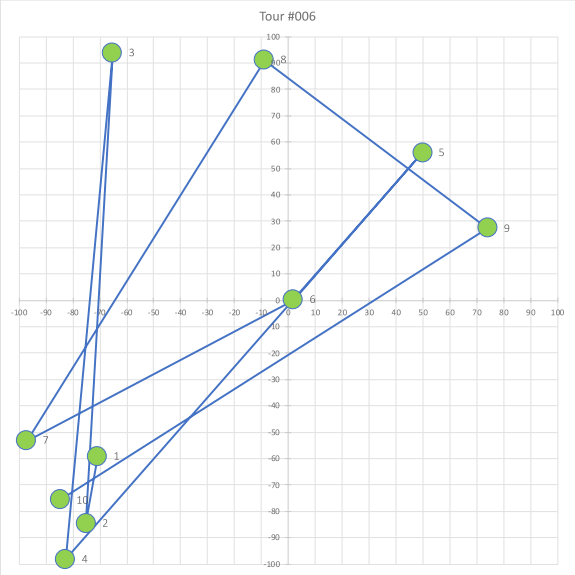


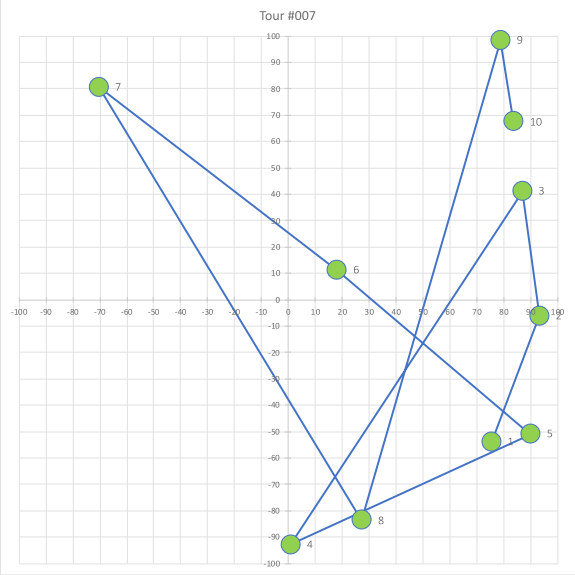


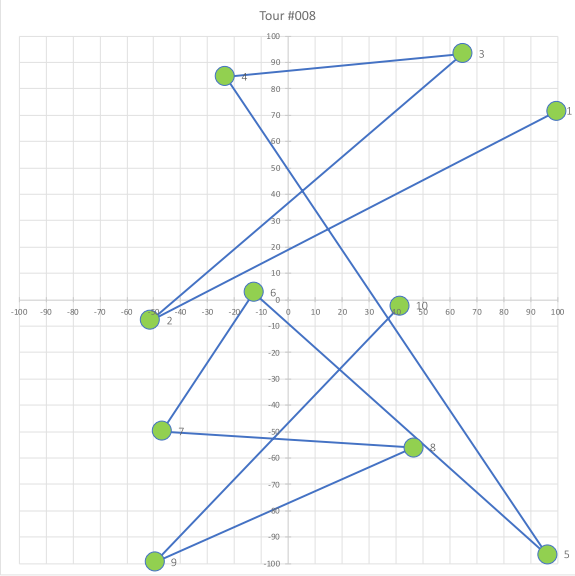


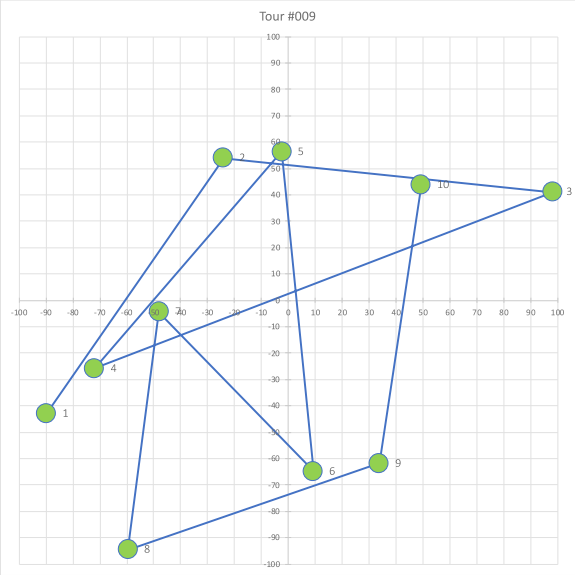


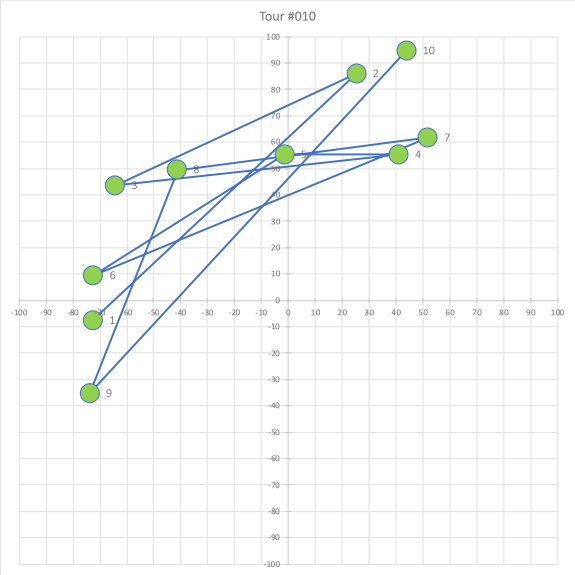

Tour #005

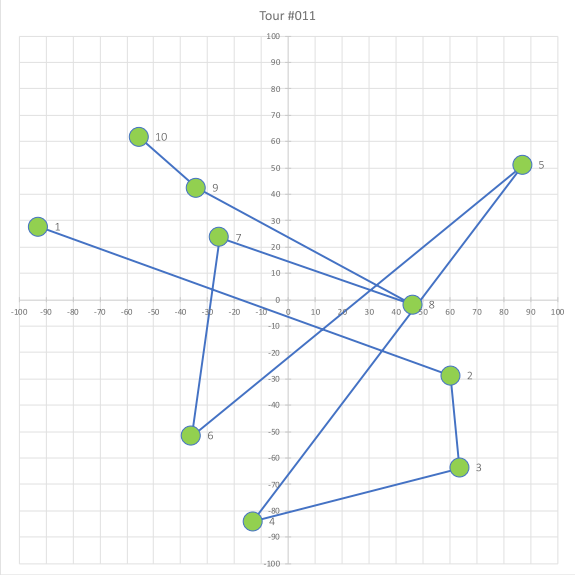

Tour #006

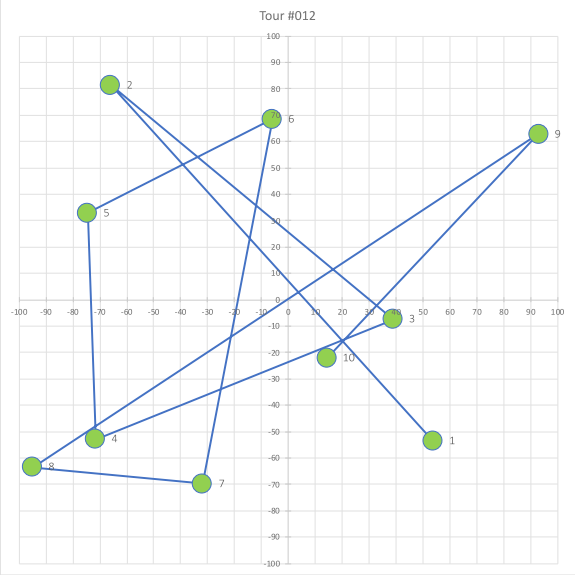

Tour #007

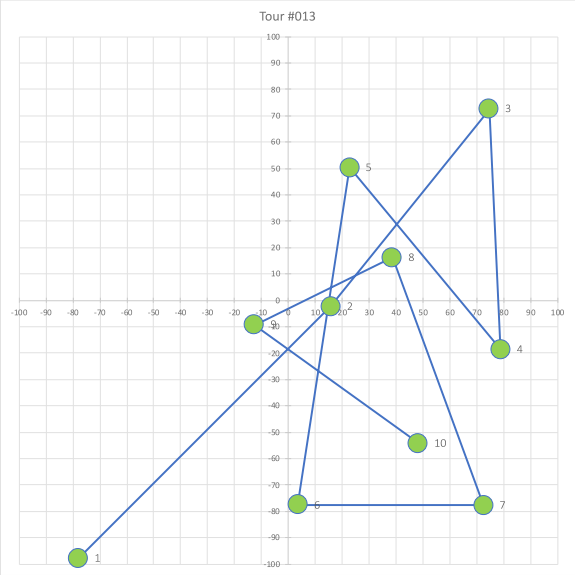

Tour #008

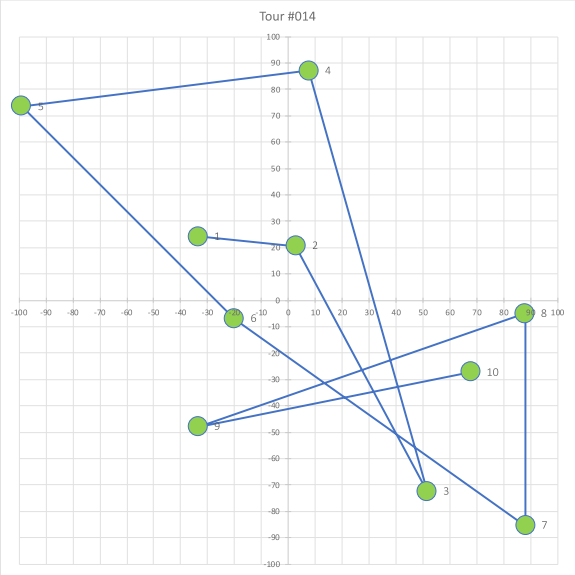

Tour #009
1
2
3
4
5
6
7
8
9
10

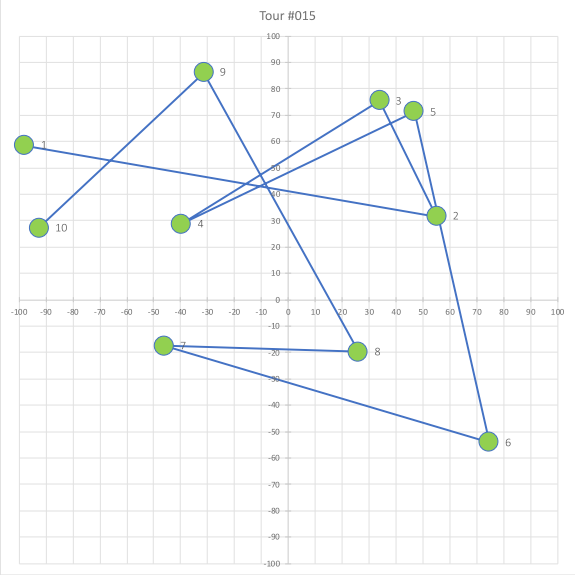

Tour #010
1
2
3
4
5
6
7
8
9
10


Tour #011
1
2
3
4
5
6
7
8
9
10

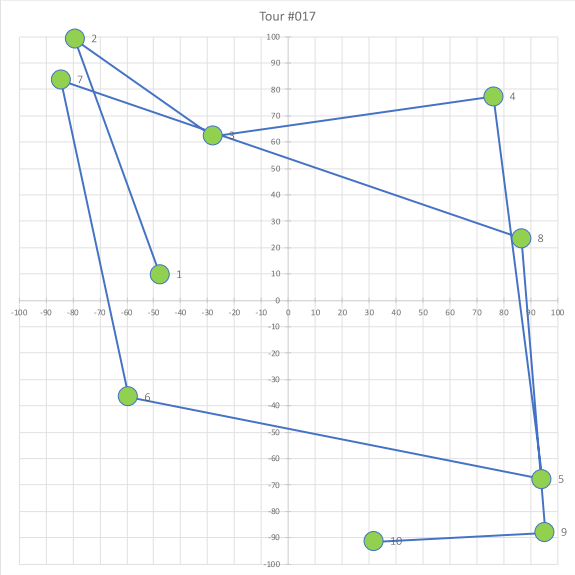

Tour #012
1
2
3
4
5
6
7
8
9
10

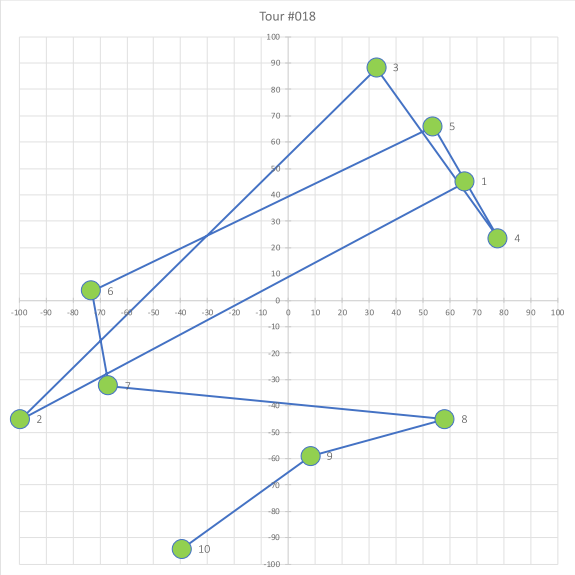

Tour #013
3
5
8
2
9
4
10
6
7
1

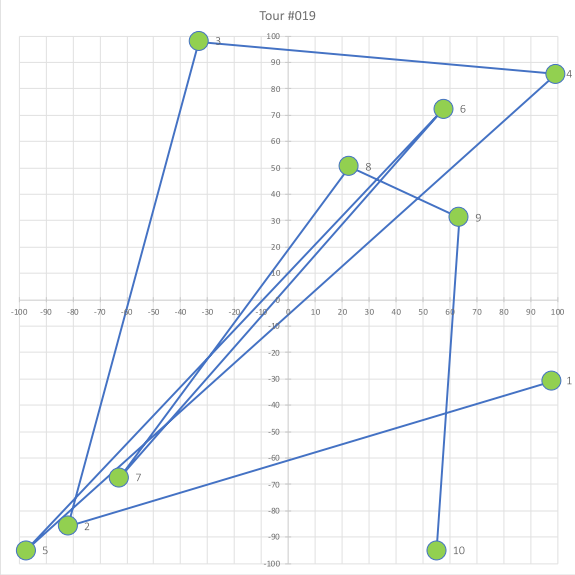

Tour #014
4
5
1
2
6
8
10
9
3
7

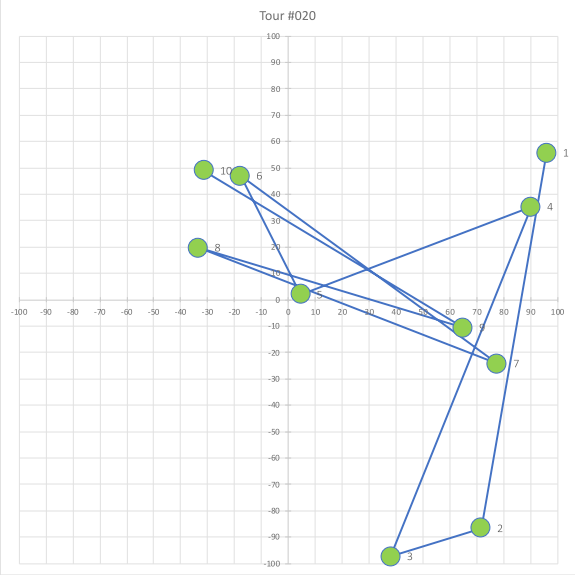

Tour #015
9
3
5
1
2
10
4
7
8
6

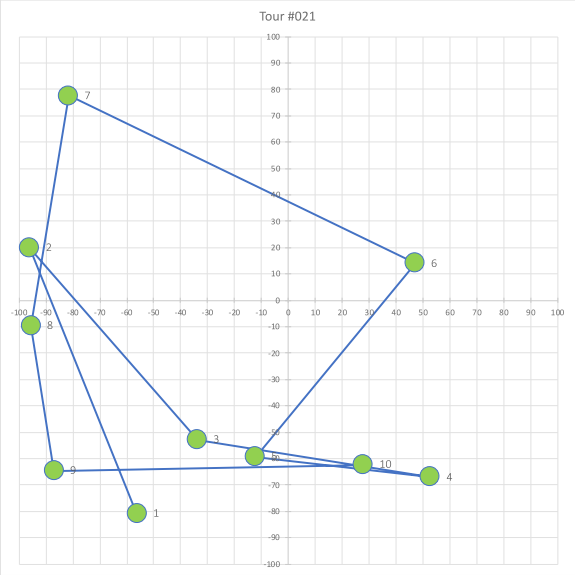

Tour #016
5
10
8
7
6
2
4
3
9
1

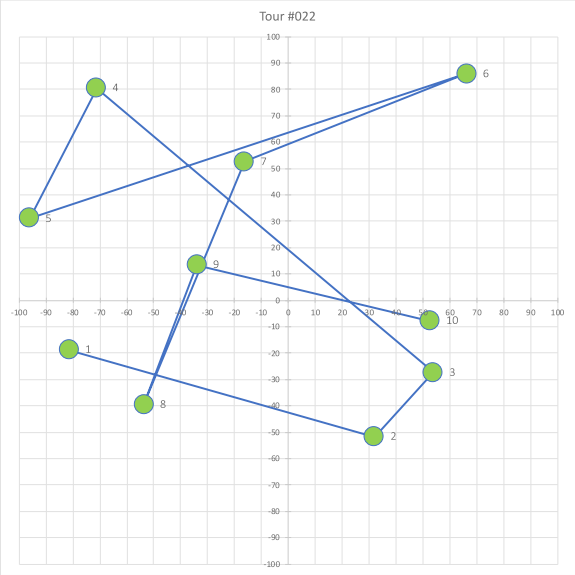

Tour #017
2
7
3
4
8
1
6
5
9
10

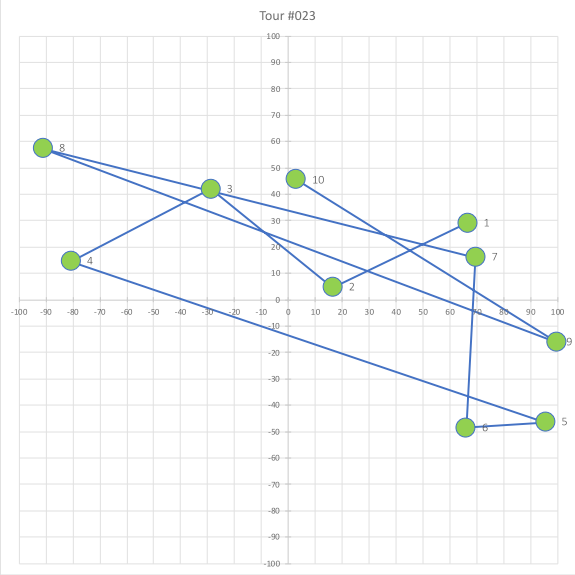

Tour #018
3
5
1
4
6
7
2
8
9
10

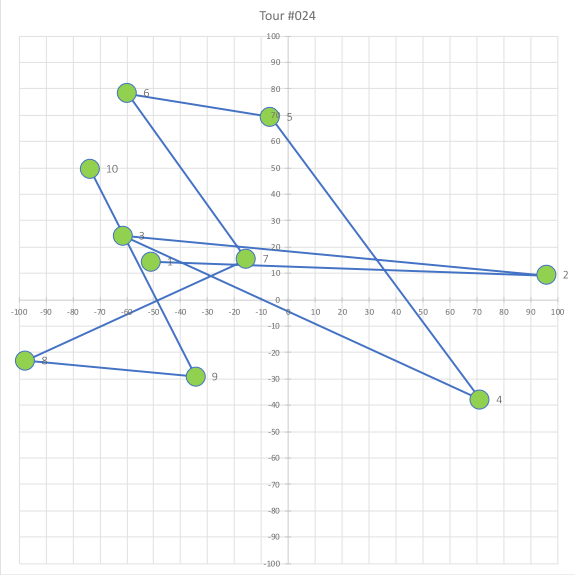

Tour #019
3
4
6
8
9
1
7
2
5
10

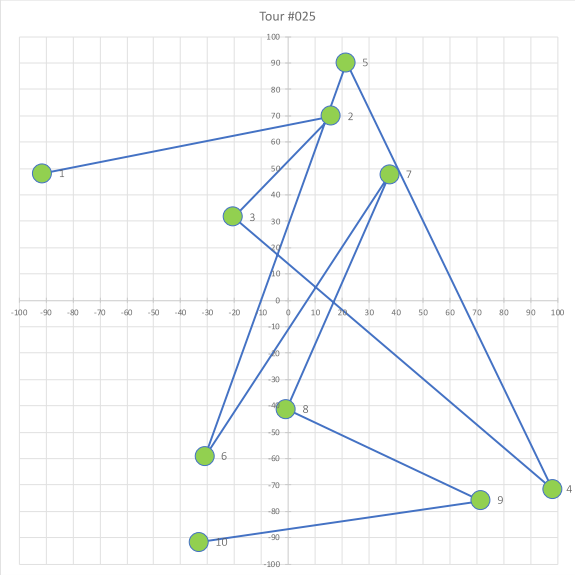

Tour #020
1
10
6
4
8
7
2
3

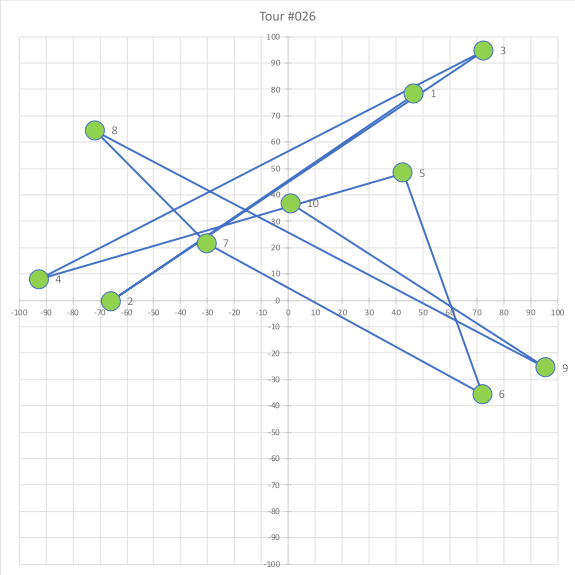

Tour #021
7
6
2
8
3
9
10
4
1

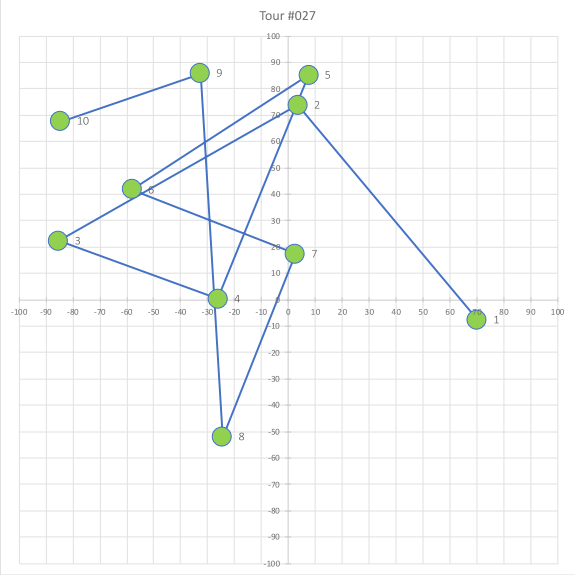

Tour #022
6
4
7
5
9
10
1
3
8
2

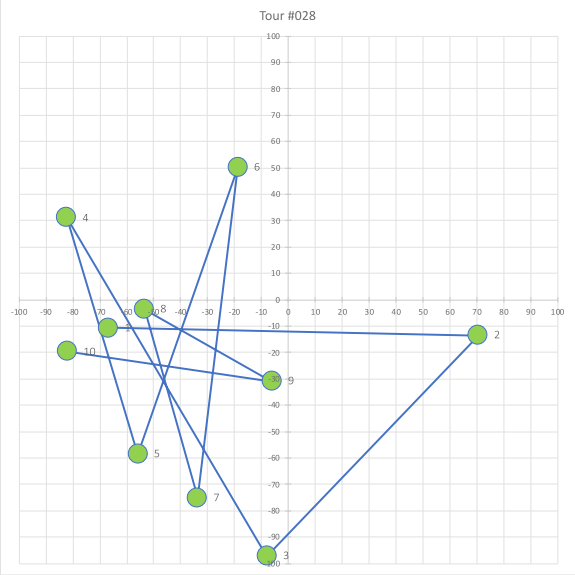

Tour #023
8
10
3
1
4
7
2
9
6
5

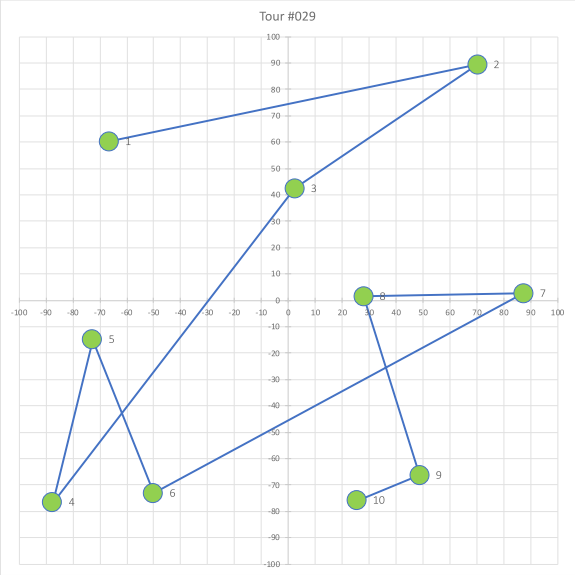

Tour #024
6
5
10
3
1
7
2
8
9
4

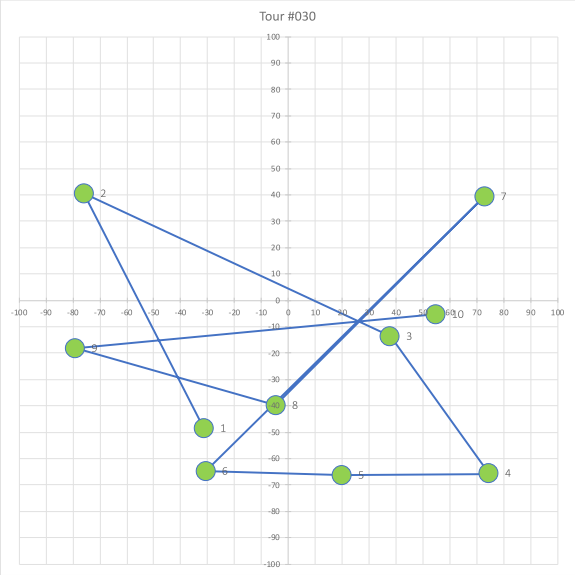

Tour #025

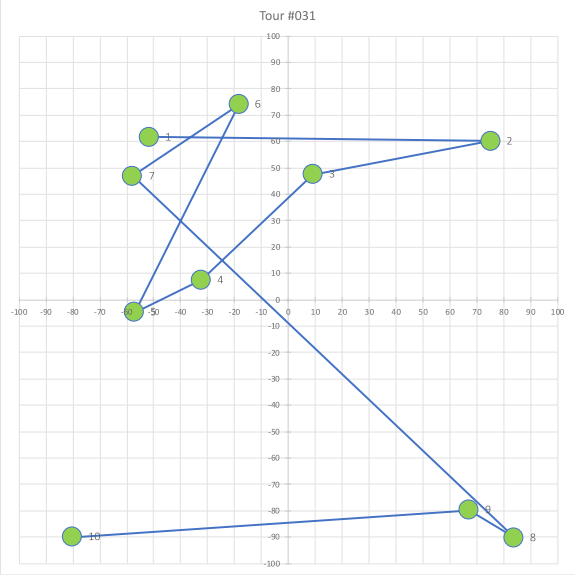

Tour #026

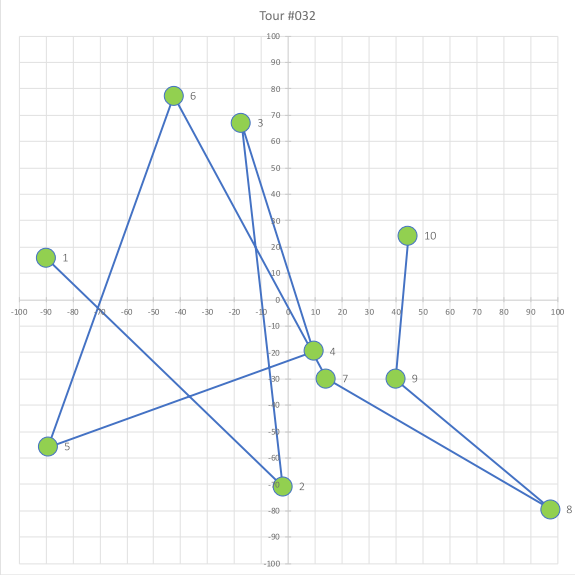

Tour #027

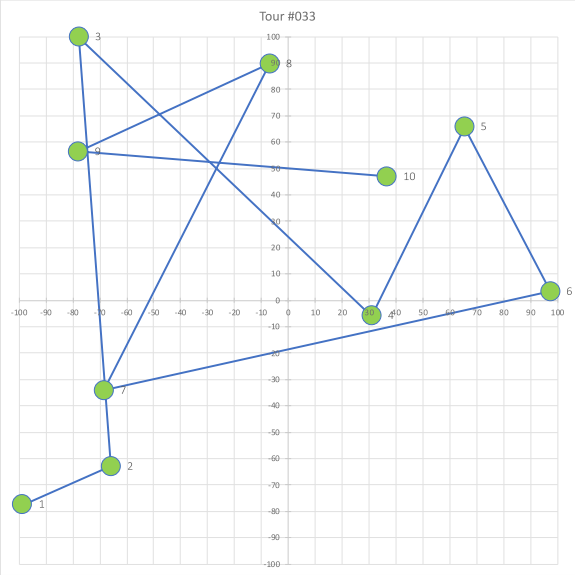

Tour #028

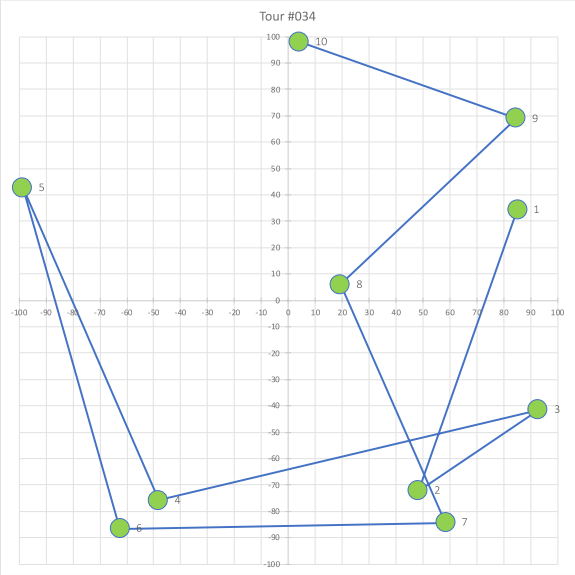

Tour #029
1
2
3
4
5
6
7
8
9
10

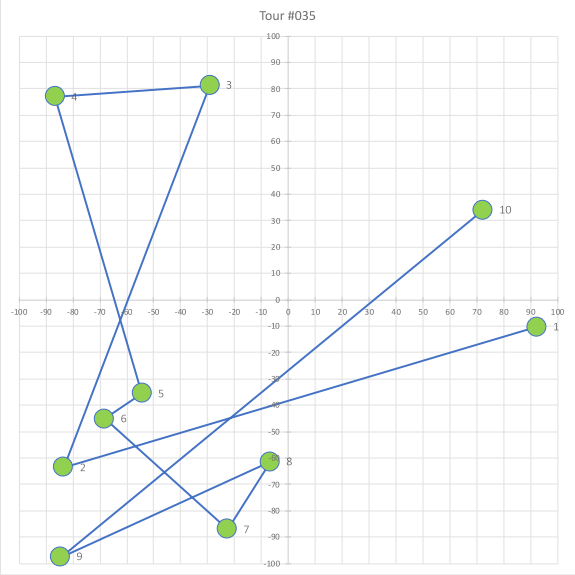

Tour #030
1
2
3
4
5
6
7
8
9
10

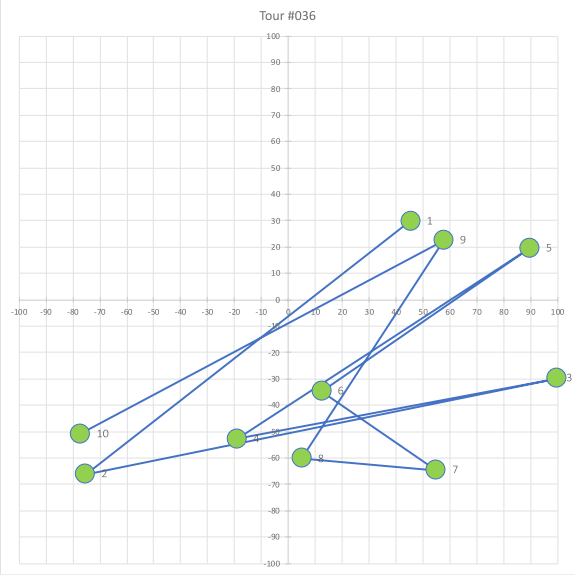

Tour #031
1
2
3
4
5
6
7
8
9
10

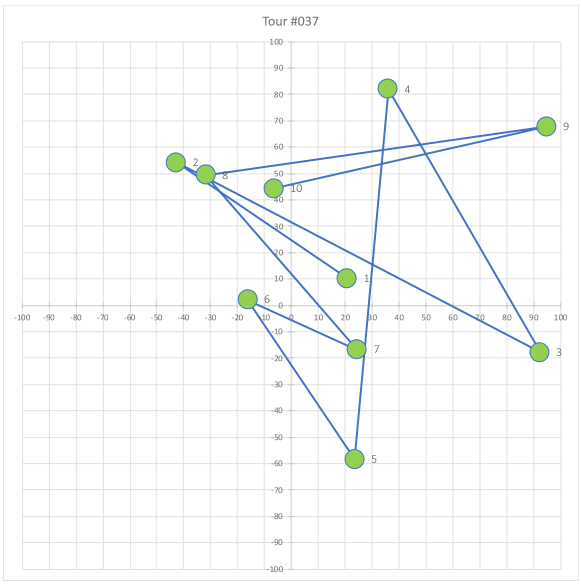

Tour #032
1
2
3
4
5
6
7
8
9
10

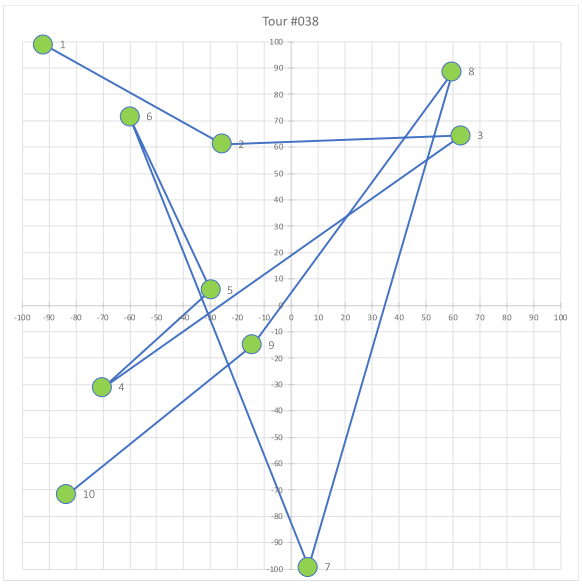

Tour #033

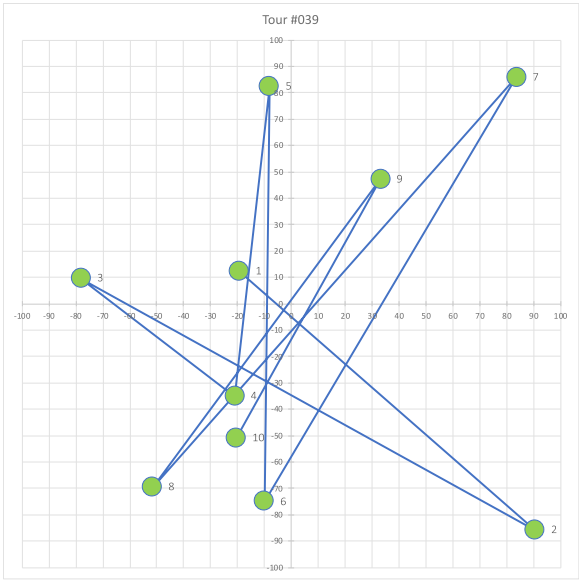

Tour #034

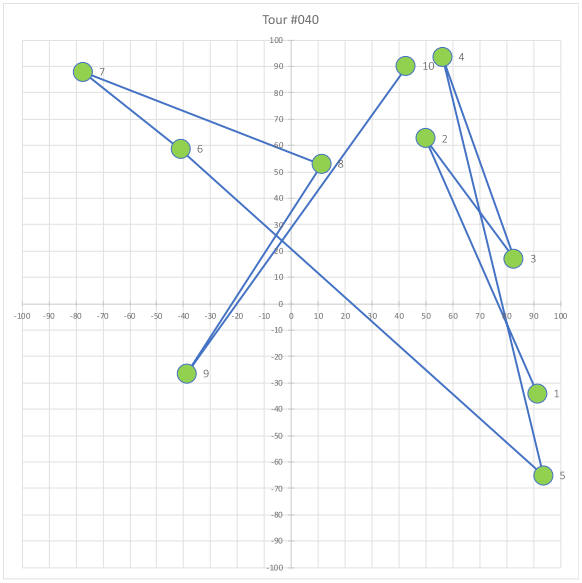

Tour #035

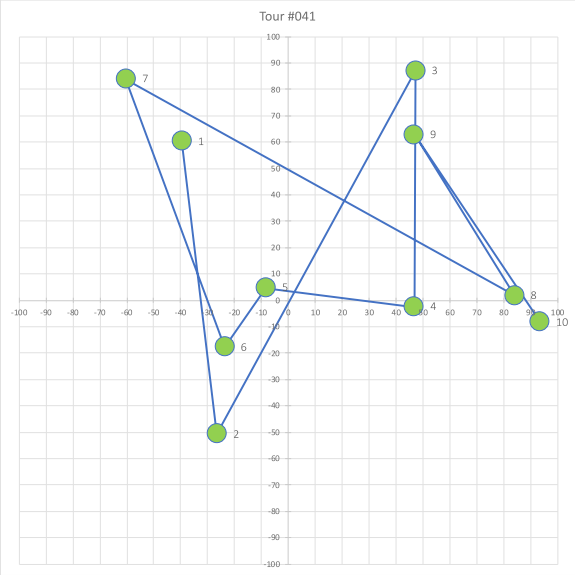

Tour #036

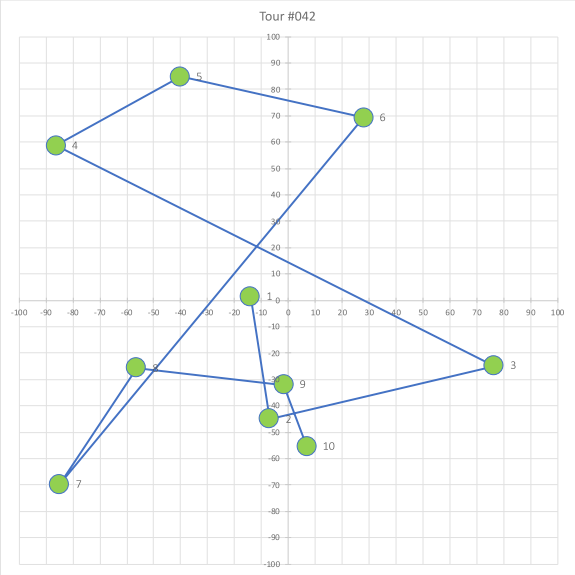

Tour #037
4
9
2
8
10
1
6
7
3
5

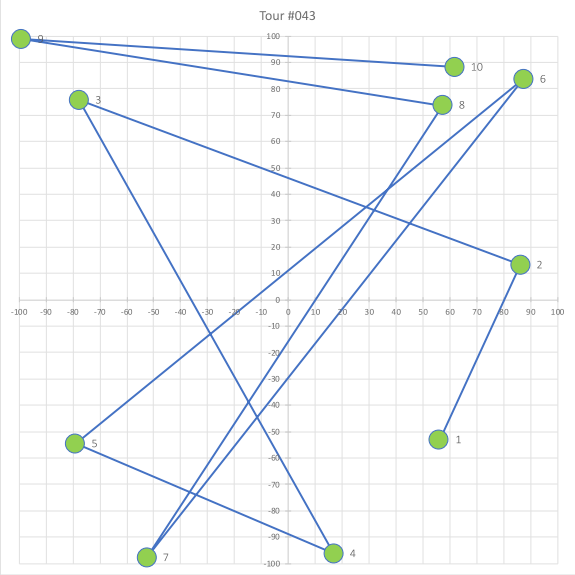

Tour #038
1
8
6
2
3
5
9
4
10
7

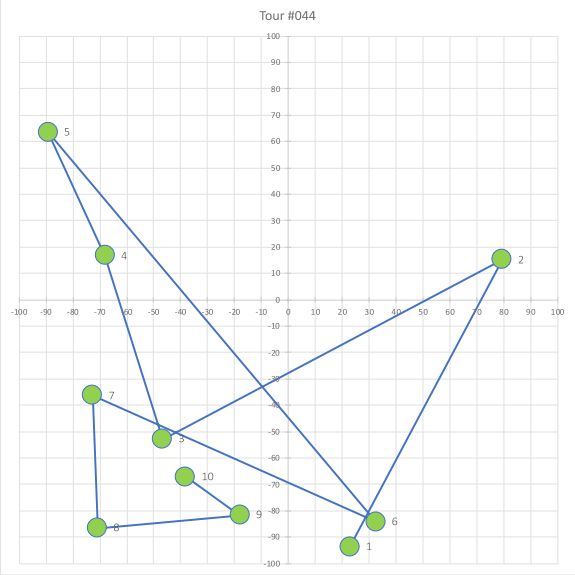

Tour #039
7
5
9
3
1
4
10
8
6
2

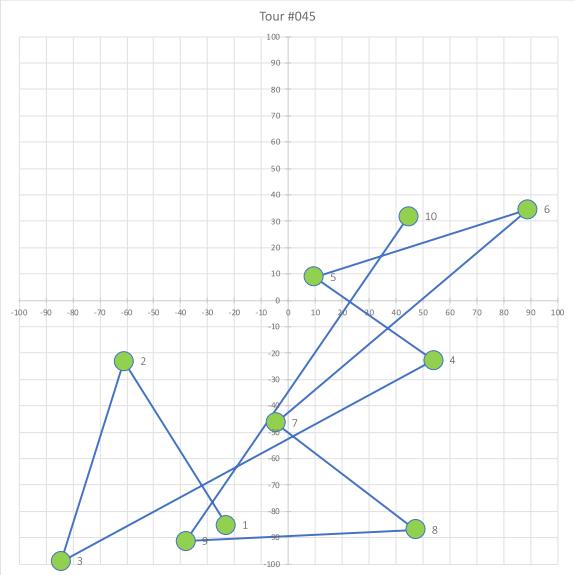

Tour #040
7
10
4
6
2
8
3
9
1
5

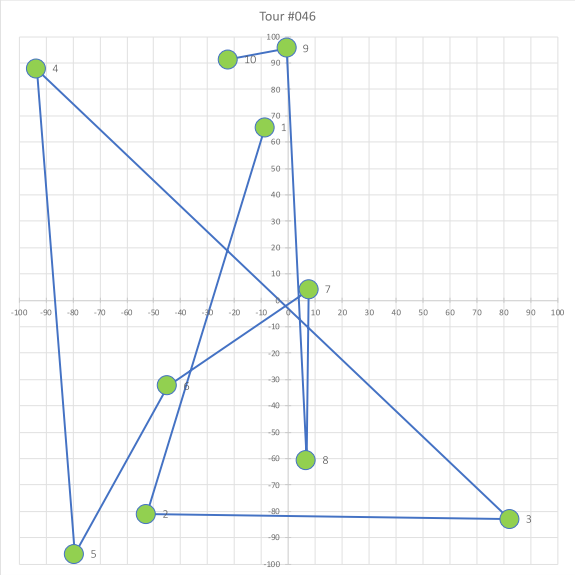

Tour #041
7
1
3
9
5
4
8
10
6
2

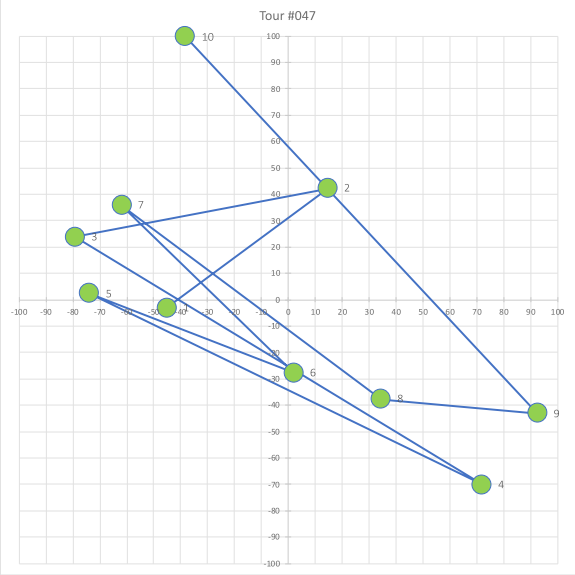

Tour #042
5
6
4
1
3
8
9
2
10
7

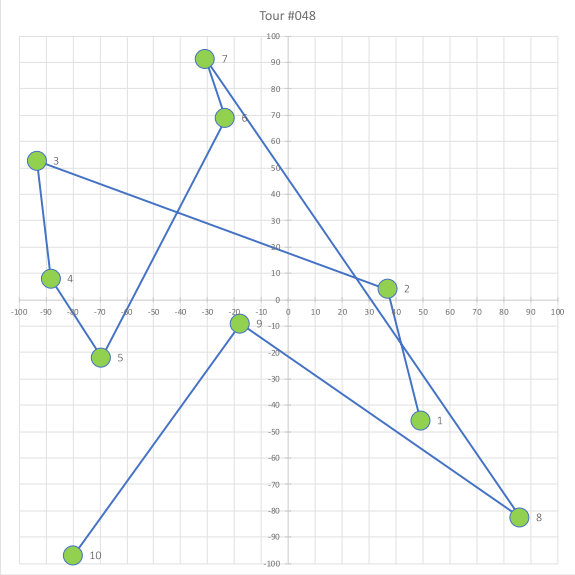

Tour #043
9
10
6
3
8
2
5
1
4
7

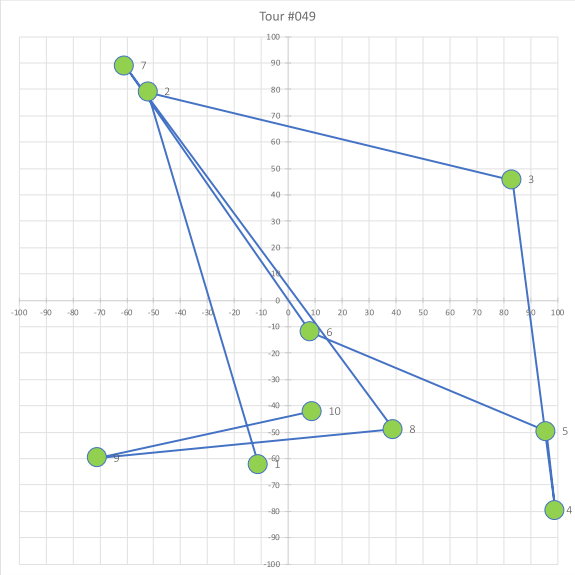

Tour #044
5
4
2
7
3
10
9
8
6
1

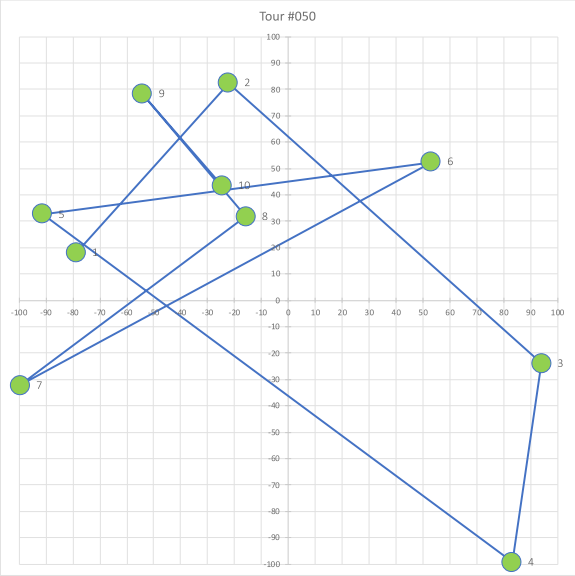

Tour #045
10
6
5
4
2
7
1
8
9
3

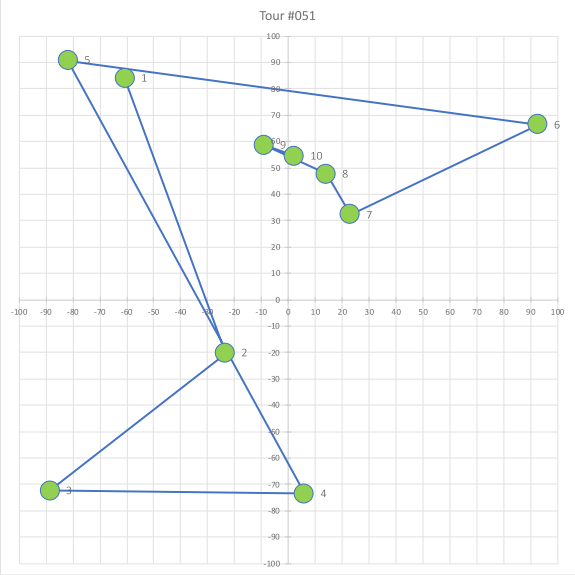

Tour #046
9
10
4
1
7
6
8
2
3
5

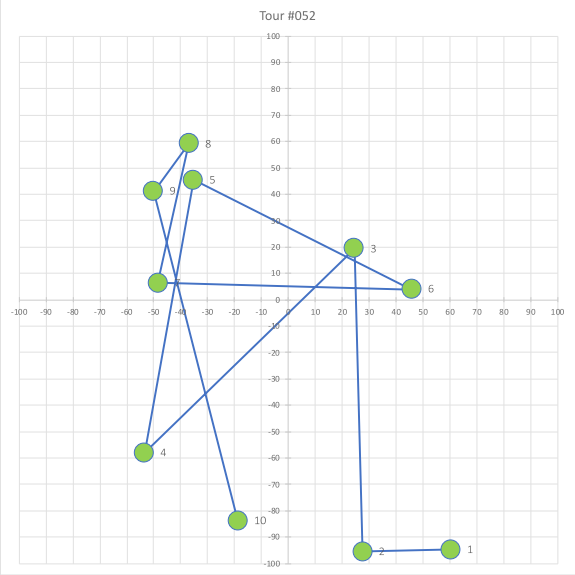

Tour #047
10
2
7
3
5
9
6
8
4

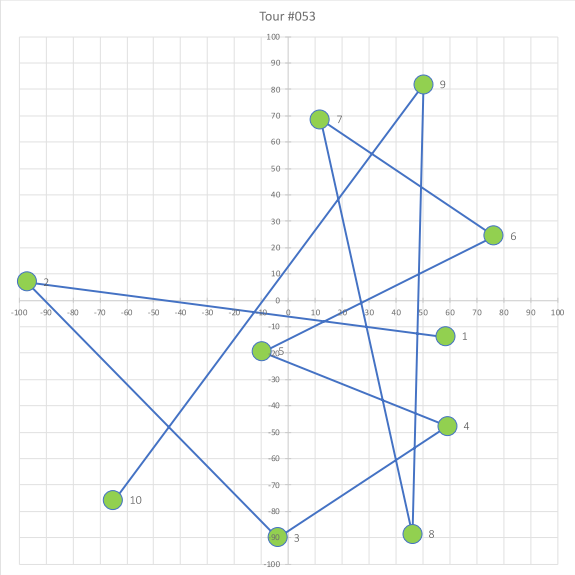

Tour #048
7
6
3
4
2
5
9
1
8
10

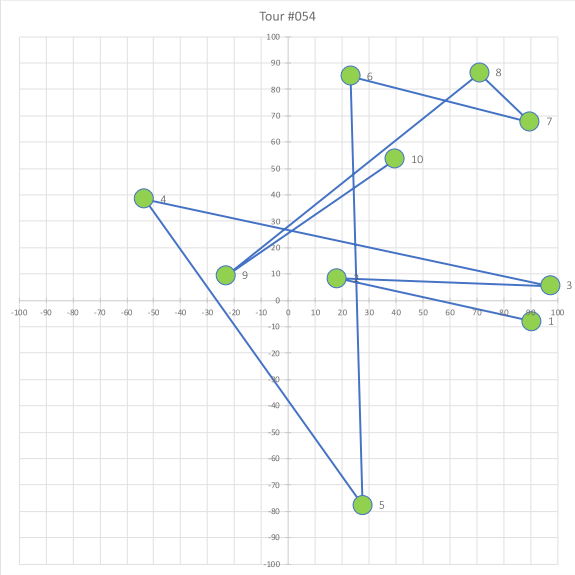

Tour #049
7
2
3
6
10
8
5
9
1
4

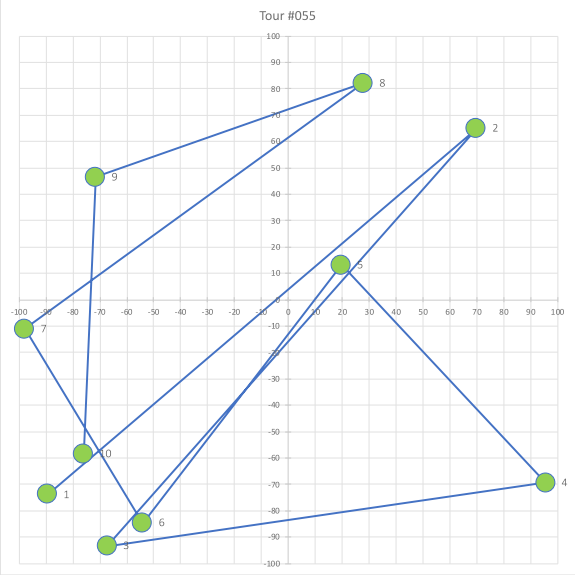

Tour #050
9
2
6
10
5
8
1
7
3
4

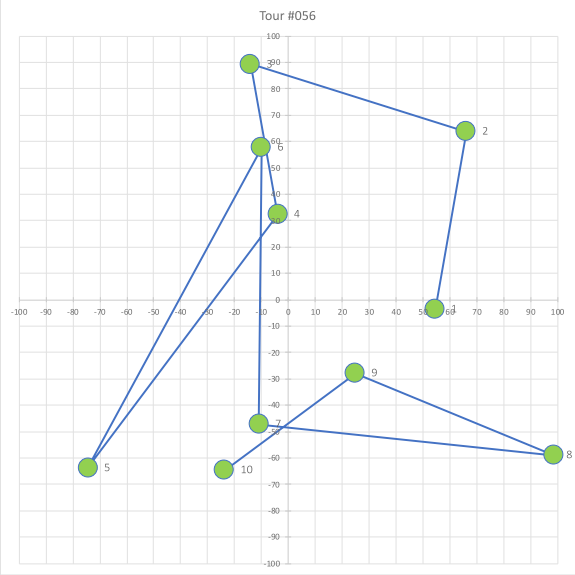

Tour #051
5
1
6
9
10
8
7
2
3
4

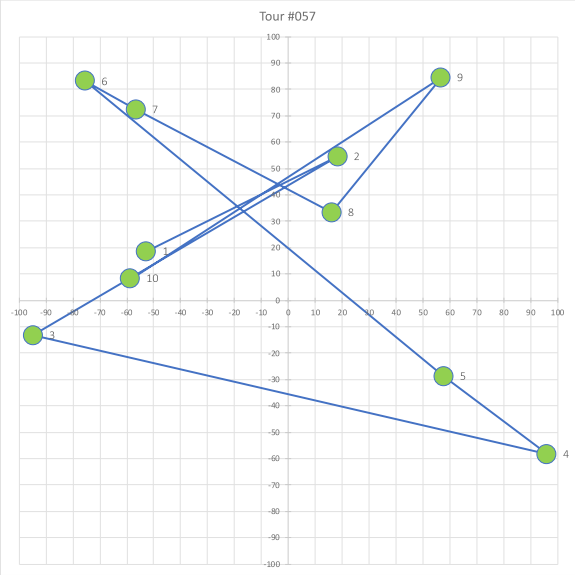

Tour #052
8
9
5
3
6
4
10
2
1

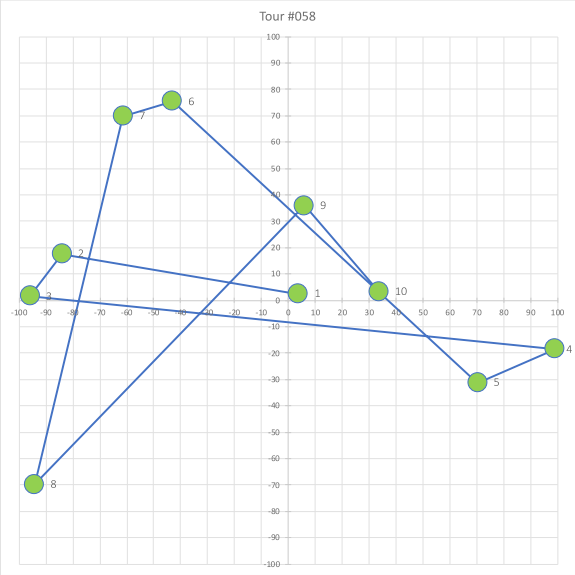

Tour #053
9
7
6
2
1
5
4
10
3
8

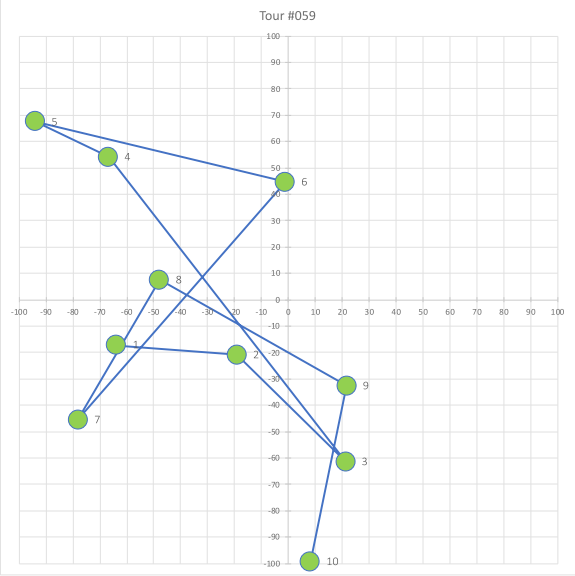

Tour #054
6
8
7
10
4
9
3
1
5

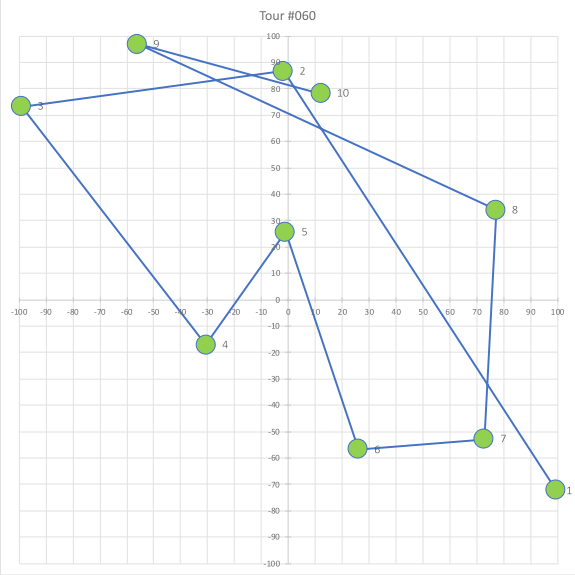

Tour #055
8
2
9
5
7
10
1
6
3
4

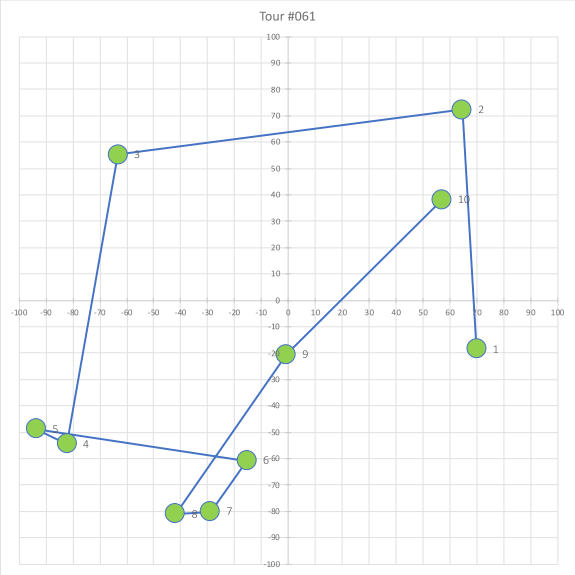

Tour #056
2
6
4
9
7
8
5
10

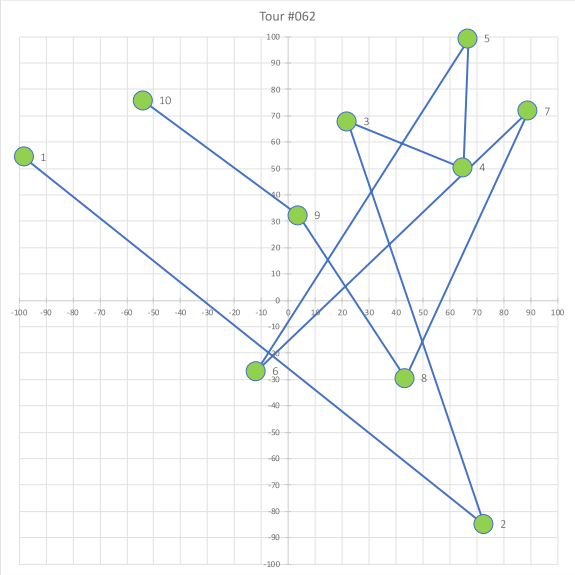

Tour #057
6
7
9
2
8
1
10
3
5
4

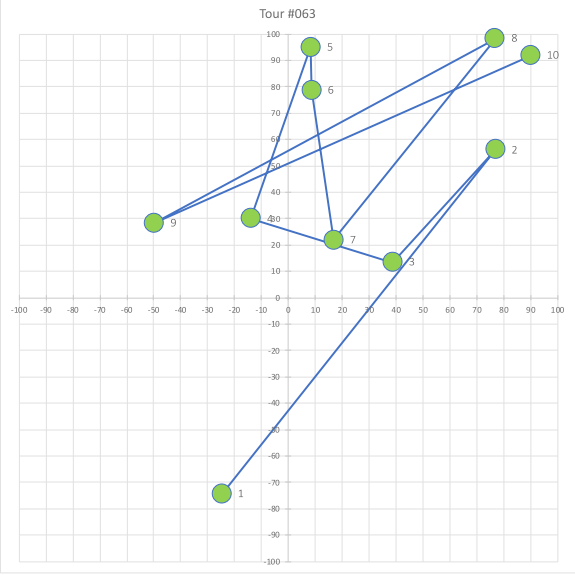

Tour #058
6
7
9
2
1
10
3
4
5
8

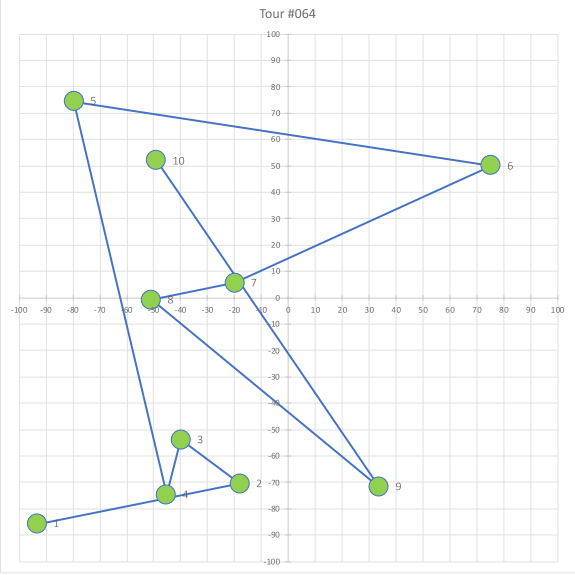

Tour #059
5
4
6
8
1
2
9
7
3
10

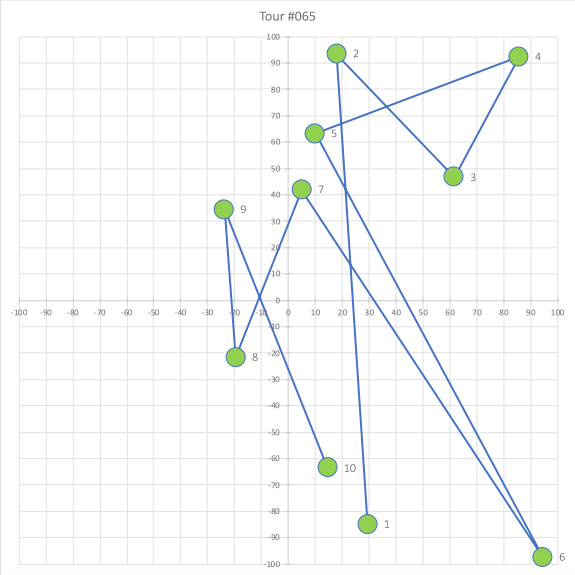

Tour #060
9
2
10
3
8
5
4
6
7
1

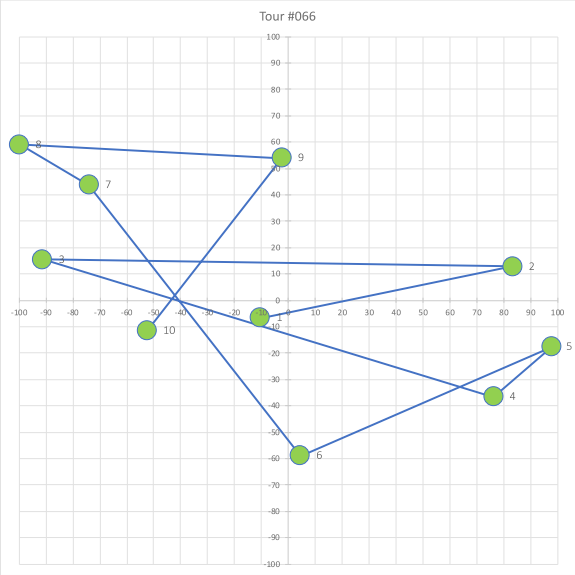

Tour #061
2
3
10
1
9
5
4
6
8
7

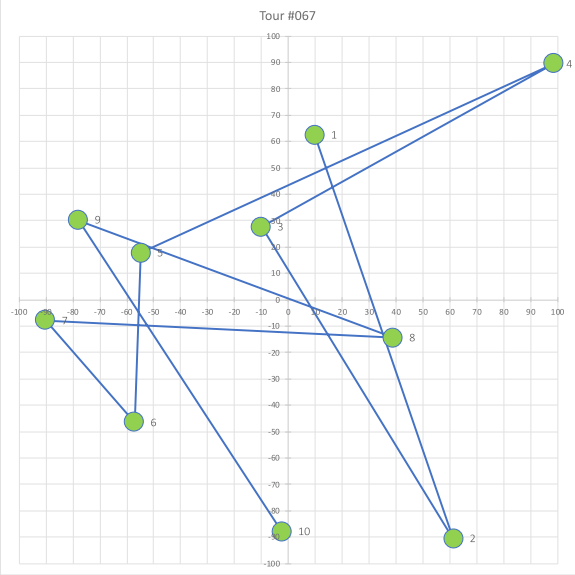

Tour #062
5
10
3
7
1
4
9
6
8
2

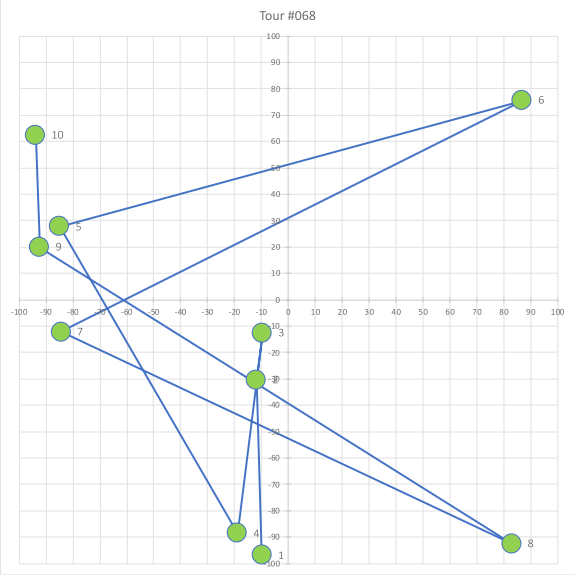

Tour #063
8
10
5
6
2
9
7
3
1

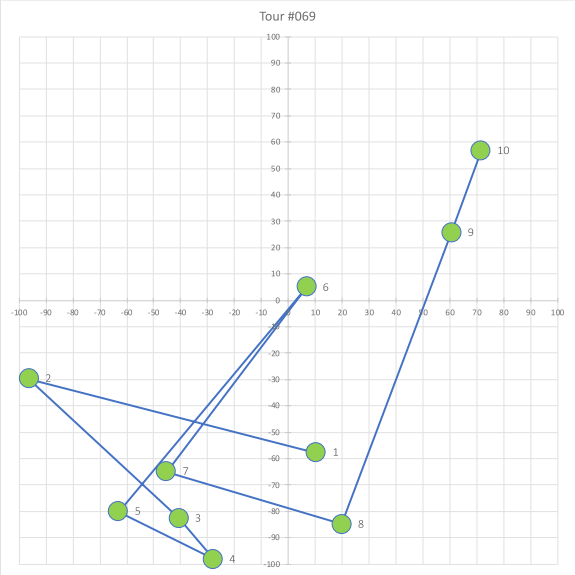

Tour #064
5
6
10
7
8
3
2
9
4
1

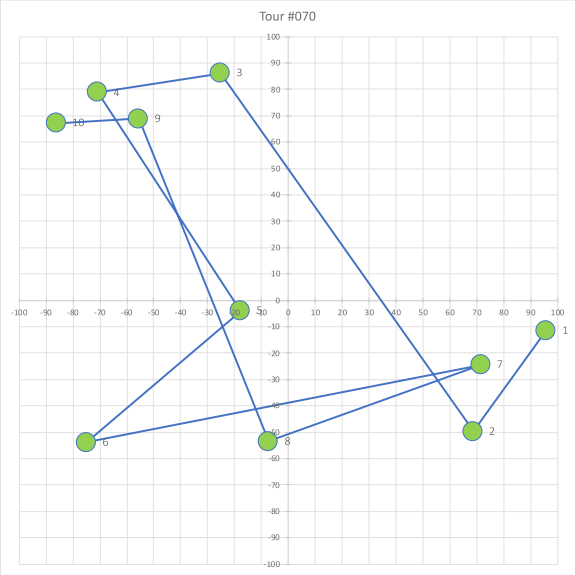

Tour #065
2
4
5
3
7
9
8
10
1
6

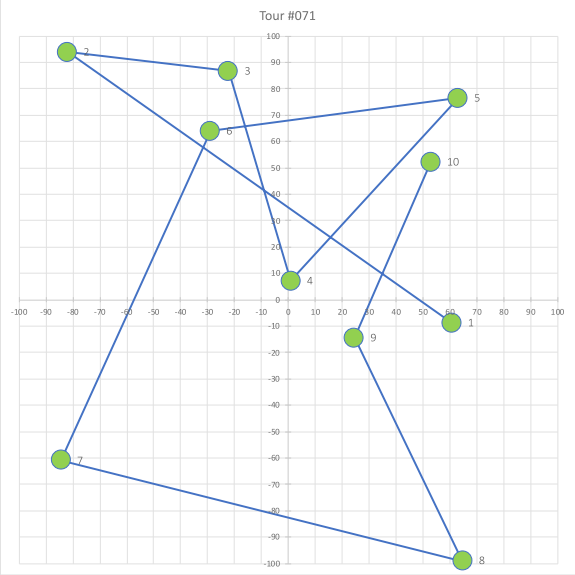

Tour #066
8
9
7
3
2
10
5
4
6

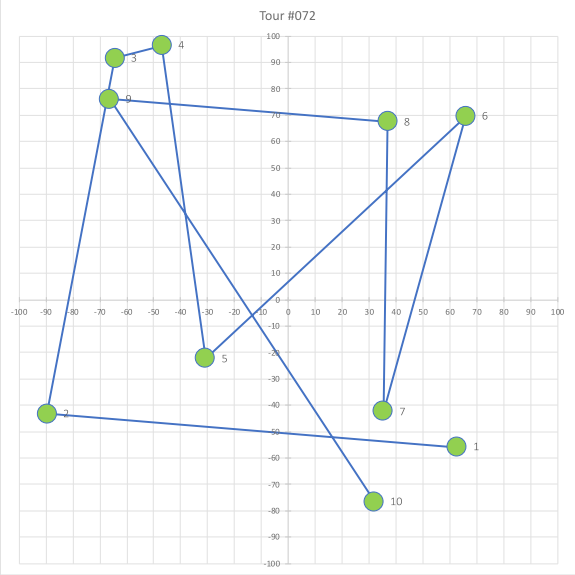

Tour #067
4
1
9
3
5
7
8
6
10
2

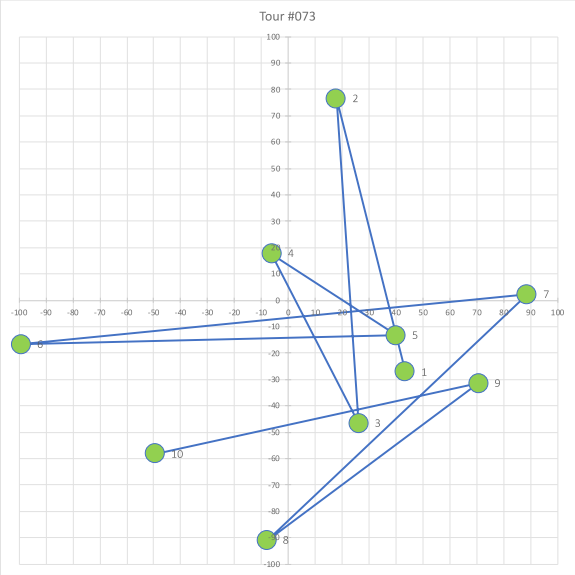

Tour #068
6
10
5
9
7
3
4
8
1

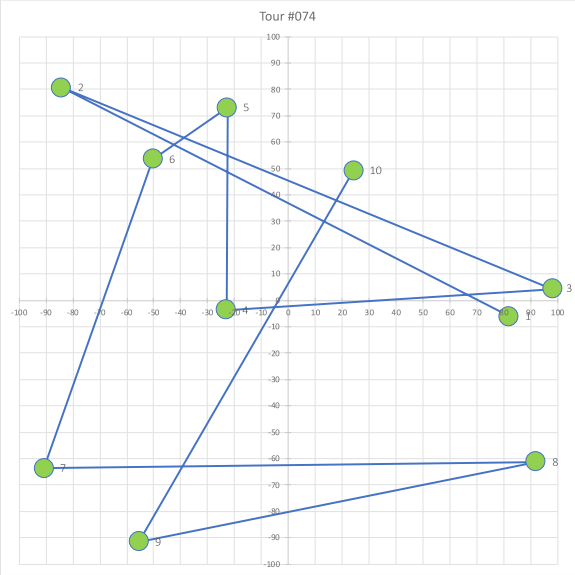

Tour #069
10
9
6
2
1
7
5
3
8
4

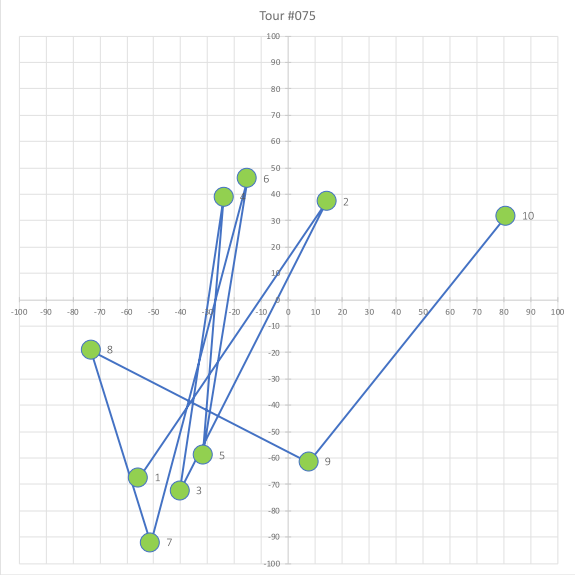

Tour #070
3
4
10
9
5
1
7
6
8
2

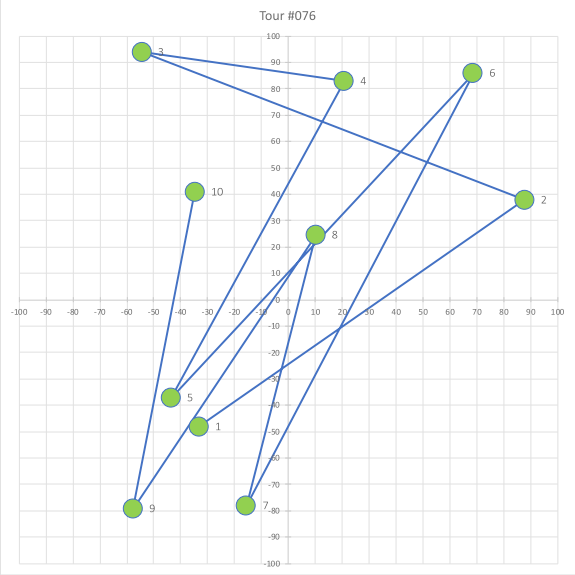

Tour #071
2
3
5
6
10
4
1
9
7
8

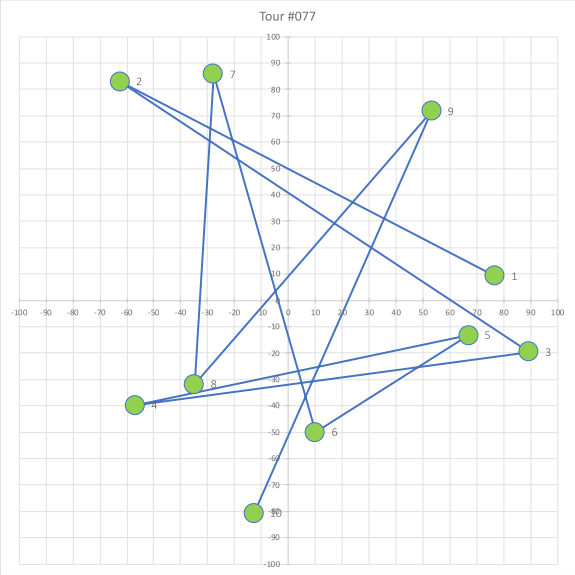

Tour #072
4
3
9
8
6
5
7
2
1
10

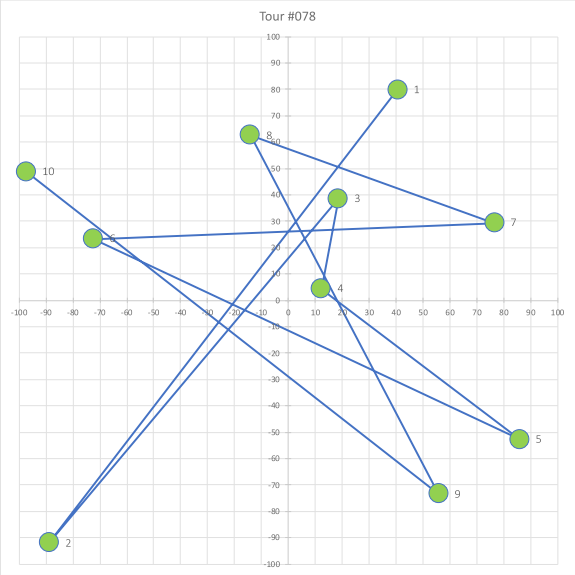

Tour #073

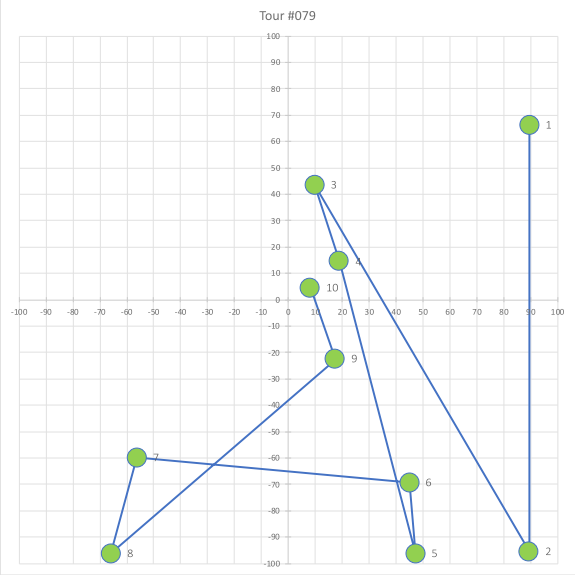

Tour #074

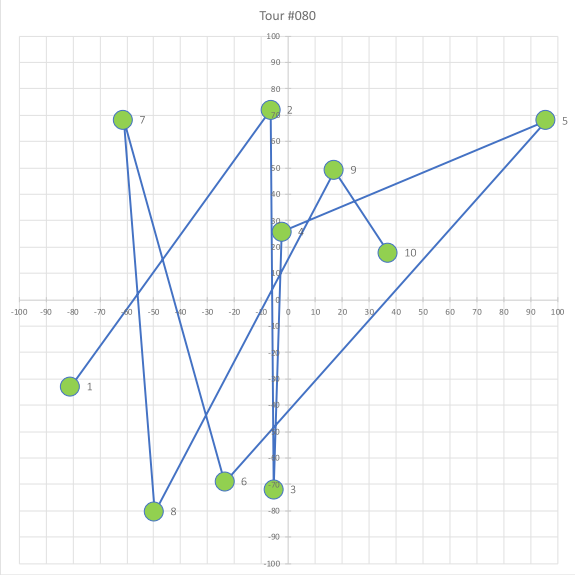

Tour #075

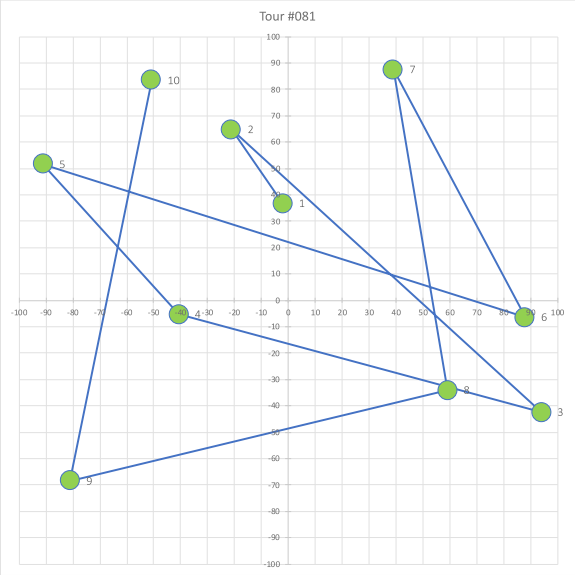

Tour #076

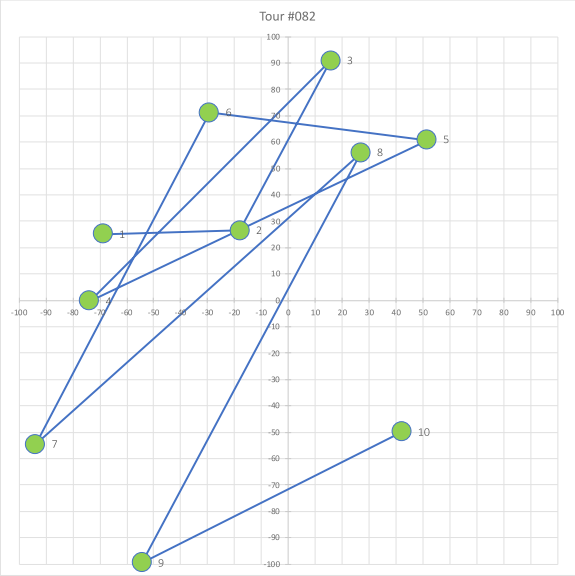

Tour #077

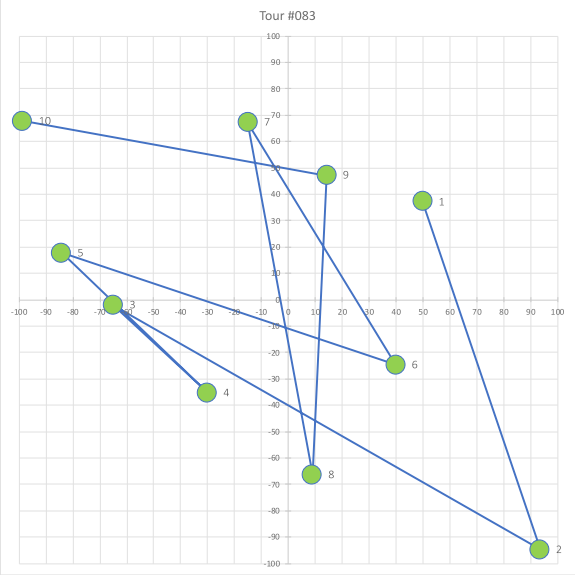

Tour #078

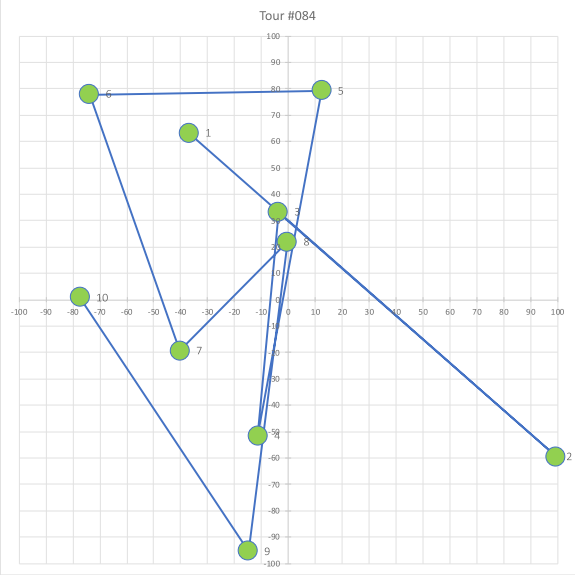

Tour #079

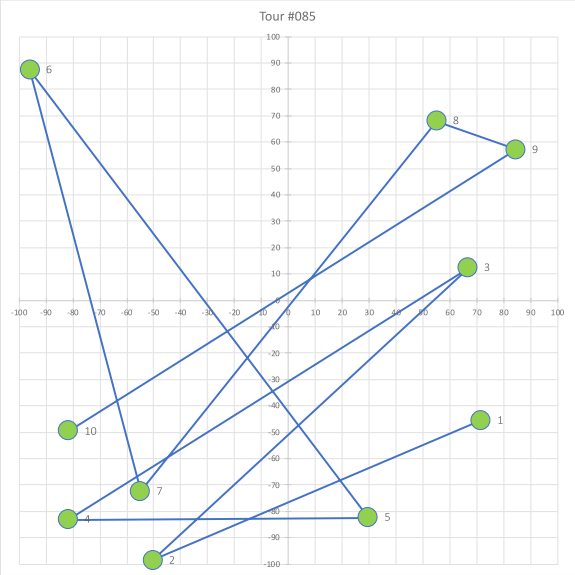

Tour #080

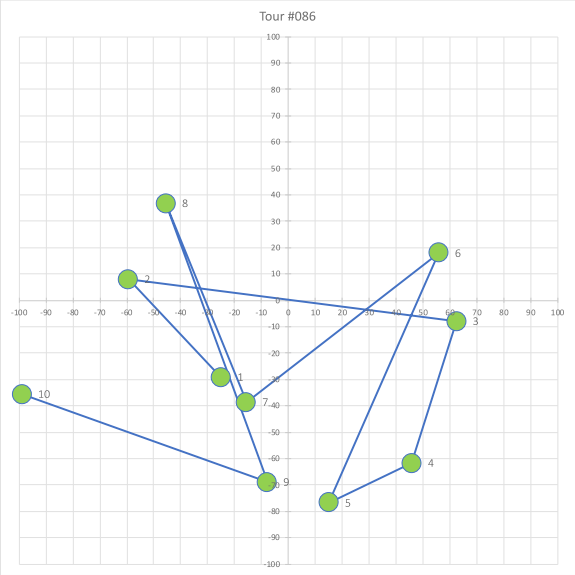

Tour #081
10
7
2
5
1
6
4
8
3
9

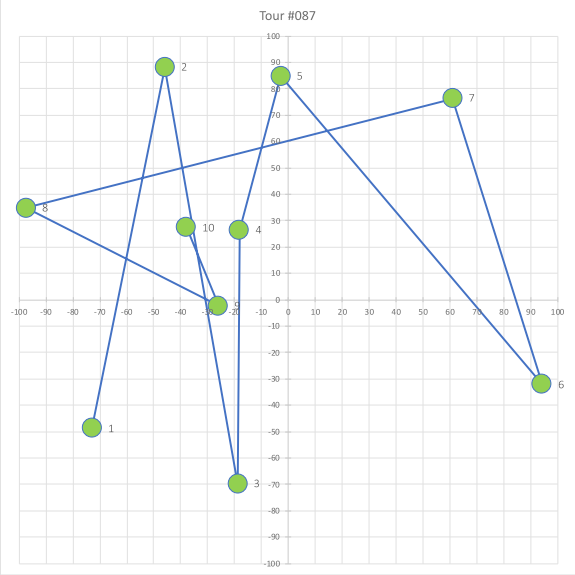

Tour #082
3
6
5
8
1
2
4
10
7
9

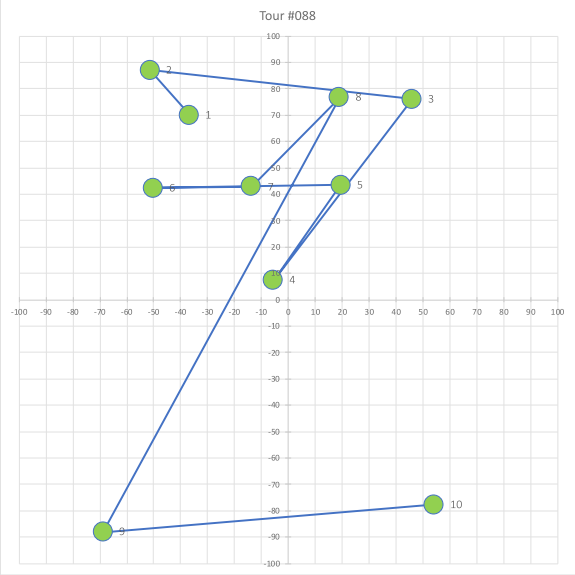

Tour #083
10
7
9
1
5
3
6
4
8
2

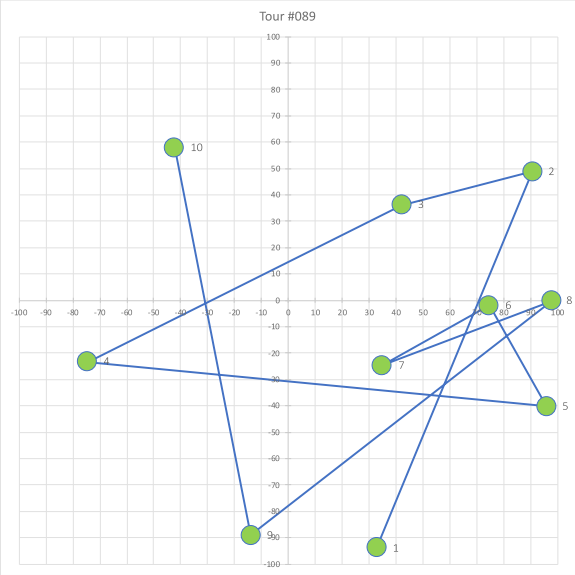

Tour #084
6
5
1
3
8
10
7
4
2
9

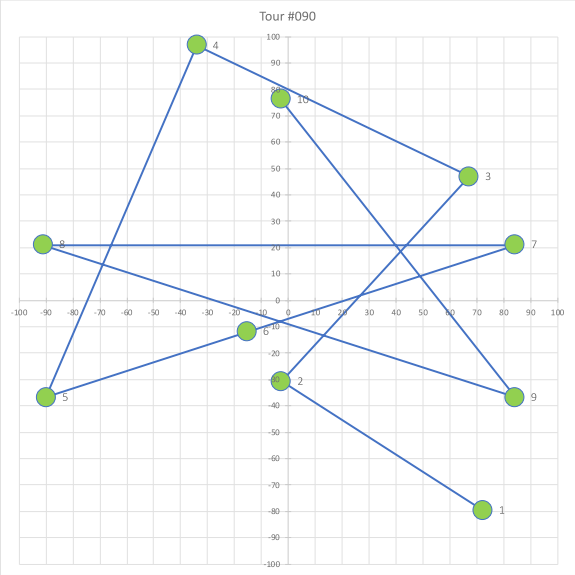

Tour #085
6
8
9
3
10
1
7
4
5
2

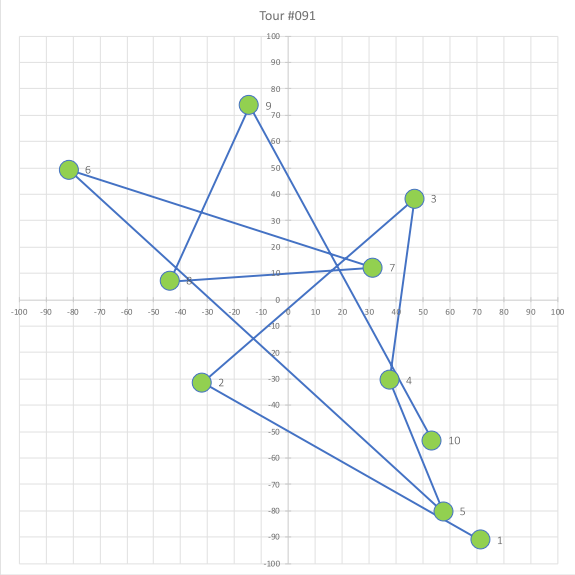

Tour #086
8
6
2
3
1
7
10
9
4
5

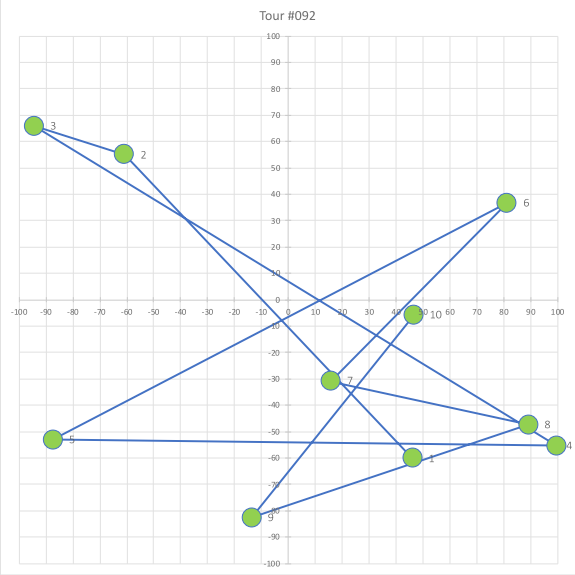

Tour #087
2
5
7
8
10
4
6
1
3

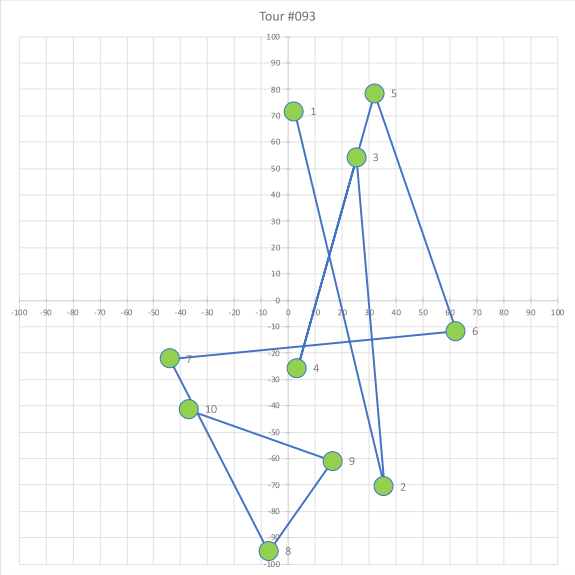

Tour #088
2
8
3
1
6
7
5
4
9
10

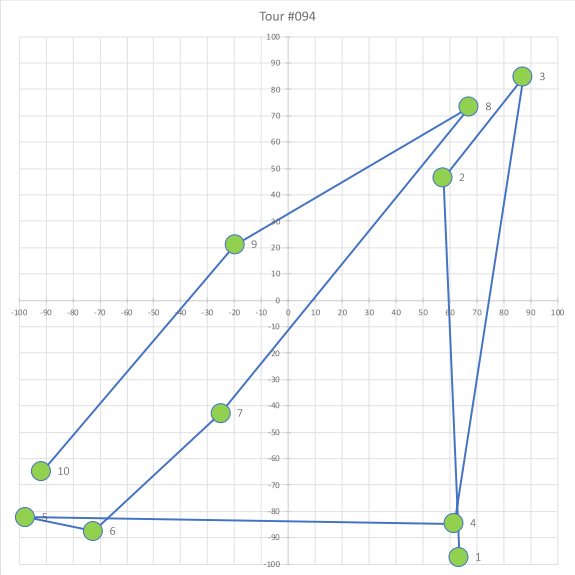

Tour #089
10
2
3
8
6
4
7
5
9
1

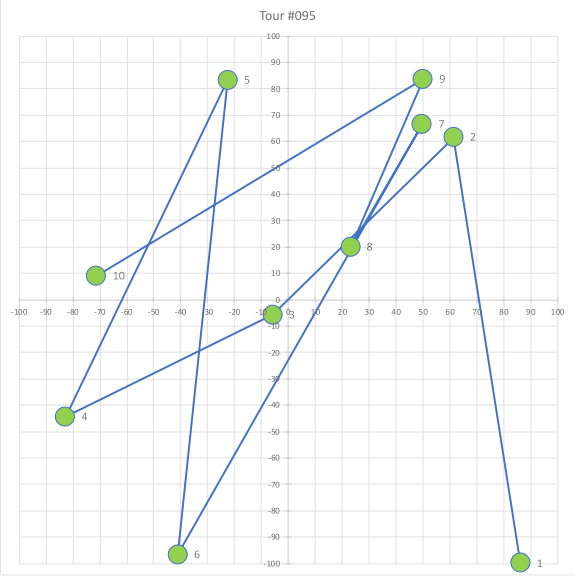

Tour #090
4
10
3
8
7
6
5
2
9
1

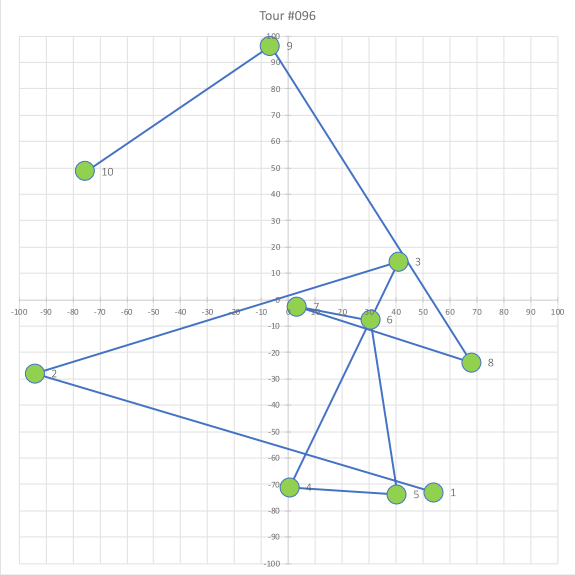

Tour #091
9
6
3
7
8
2
4
10
5
1

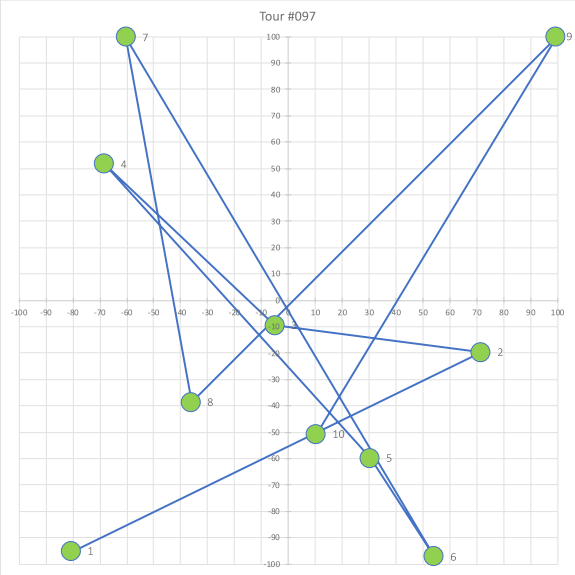

Tour #092
3
2
6
10
7
8
4
5
1
9

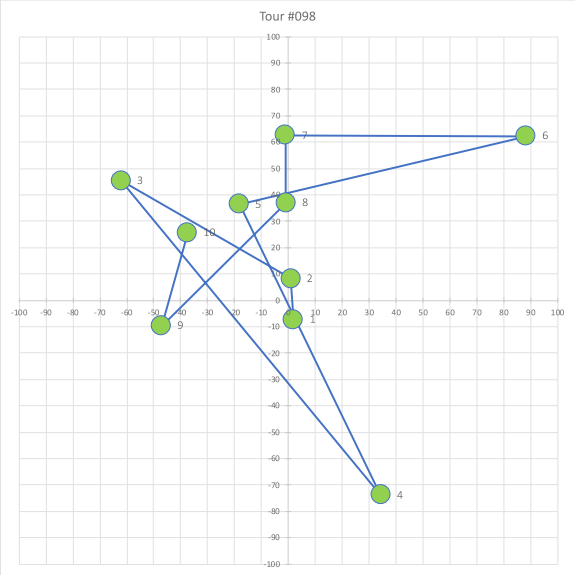

Tour #093

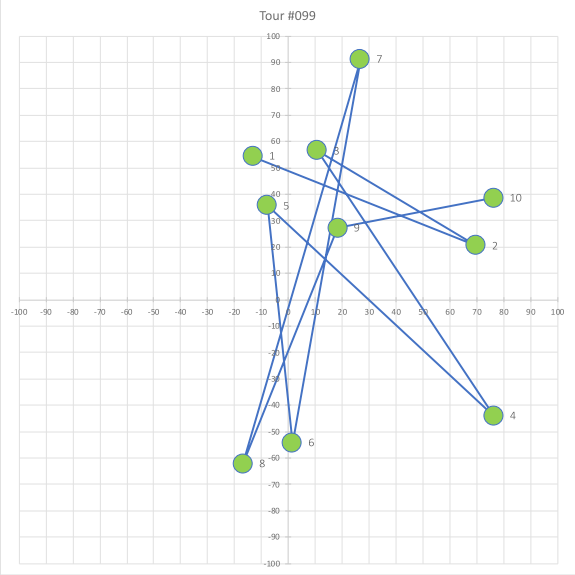

Tour #094

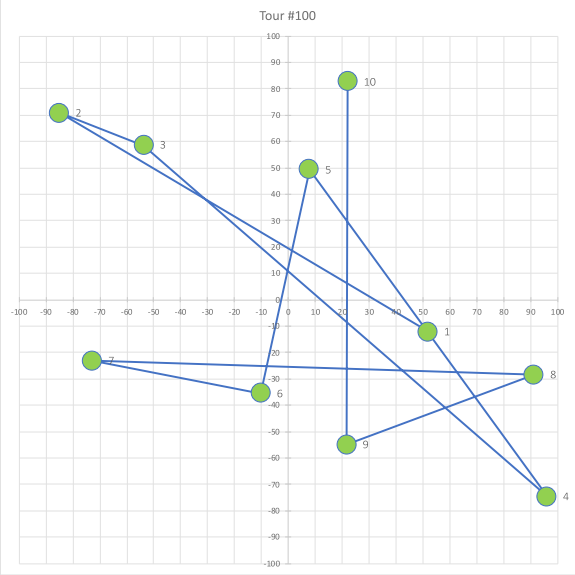

Tour #095


Tour #096


Tour #097
7
9
4
2
8
10
5
6
1


Tour #098
7
6
3
5
8
10
2
9
1
4


Tour #099
7
1
3
10
5
9
2
4
6
8


Tour #100
10
2
3
5
1
7
8
6
9
4

Tabular POI-Layout Coordinates

The XY coordinates of each POI in our experimentation's 100 layouts are provided below. All POIs were at a depth of 80 m.

	POI_01		POI_02		POI_03		POI_04		POI_05		POI_06		POI_07		POI_08		POI_09		POI_10	
Tour	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
1	-87.31	-4.18	-46.35	76.95	44.83	-50.97	28.79	51.83	-39.38	-88.41	-42.18	5.97	16.49	4.90	28.79	-77.18	-66.44	28.64	-78.88	98.29
2	60.72	-66.42	97.11	-39.41	71.62	38.04	-19.48	-80.06	-76.30	-14.35	-92.21	87.85	-62.85	51.49	-60.39	-48.24	-82.28	-33.77	-5.17	-56.92
3	-28.46	55.84	23.07	41.26	-11.92	32.23	31.04	89.78	96.79	-35.43	-6.87	3.34	10.69	-53.87	57.48	-45.86	12.48	-33.63	-67.94	77.91
4	80.41	-50.25	47.18	7.02	62.40	12.66	27.07	-79.94	61.68	-50.97	62.57	42.52	69.71	20.69	-59.07	-1.59	-97.27	-17.96	-25.77	75.82
5	11.60	-92.24	53.95	-64.69	-12.87	-20.62	-6.13	-41.43	-55.91	64.71	11.51	-47.05	9.57	44.86	-96.01	51.03	24.98	80.11	-10.02	38.93
6	-70.89	-59.12	-75.08	-84.55	-65.23	93.88	-82.68	-98.13	49.87	55.98	1.96	0.10	-97.24	-53.22	-8.88	91.01	74.13	27.36	-84.69	-75.71
7	75.47	-54.04	93.37	-6.12	87.00	41.05	0.96	-92.91	90.10	-50.94	18.17	11.09	-70.32	80.61	27.38	-83.70	78.94	98.32	83.78	67.48
8	99.90	71.51	-51.17	-7.91	64.76	93.24	-23.35	84.55	96.39	-96.85	-12.71	2.72	-46.84	-49.99	46.63	-56.09	-49.57	-99.58	41.37	-2.52
9	-89.84	-42.97	-24.17	53.89	98.15	41.12	-71.94	-25.84	-2.44	56.42	9.21	-64.94	-48.02	-4.45	-59.53	-94.59	33.75	-61.79	49.30	43.89
10	-72.46	-7.70	25.41	85.96	-64.19	43.50	41.13	55.16	-1.25	55.23	-72.30	9.39	51.97	61.61	-41.23	49.37	-73.67	-35.29	43.96	94.59
11	-92.78	27.61	60.36	-28.83	63.71	-63.90	-13.10	-84.26	86.97	50.87	-35.99	-51.72	-25.57	23.54	46.21	-2.05	-34.17	42.27	-55.45	61.71
12	53.80	-53.60	-66.02	81.32	38.90	-7.49	-71.61	-52.94	-74.61	32.87	-5.98	68.42	-31.83	-69.86	-95.06	-63.49	93.03	62.90	14.25	-22.24
13	-77.94	-97.87	15.81	-2.31	74.59	72.47	78.94	-18.73	22.76	50.29	3.62	-77.63	72.49	-77.79	38.65	16.29	-12.58	-9.27	48.26	-54.18
14	-33.48	24.15	2.89	20.63	51.61	-72.54	7.67	87.04	-98.99	73.54	-19.97	-7.02	88.07	-85.47	87.88	-5.09	-33.50	-48.03	67.70	-27.20
15	-97.88	58.46	55.14	31.64	33.90	75.44	-39.64	28.71	46.74	71.30	74.58	-54.12	-46.17	-17.48	26.08	-19.78	-31.38	86.09	-92.56	27.22
16	45.59	-77.42	-5.39	-39.58	-37.99	-60.57	8.46	-51.95	-63.40	97.16	18.71	-39.15	68.77	-5.99	27.55	6.64	81.06	-58.54	-27.30	66.05
17	-47.65	9.69	-79.19	98.99	-28.05	62.25	76.28	77.29	94.29	-67.95	-59.39	-36.68	-84.44	83.51	86.67	23.36	95.37	-88.14	31.82	-91.54
18	65.64	44.86	-99.56	-45.35	32.87	88.01	78.02	23.37	53.83	65.81	-73.15	3.46	-66.71	-32.52	58.36	-45.10	8.31	-59.15	-39.46	-94.35
19	97.75	-30.97	-81.86	-85.97	-33.12	97.85	99.56	85.53	-97.35	-95.20	57.79	72.25	-62.77	-67.56	22.68	50.43	63.53	31.09	55.14	-95.28
20	96.03	55.41	71.61	-86.68	38.11	-97.37	90.17	35.21	4.57	1.94	-17.83	46.95	77.62	-24.32	-33.34	19.70	64.95	-10.71	-31.34	48.94
21	-56.16	-80.93	-96.01	19.84	-33.81	-52.87	52.81	-67.03	-12.23	-59.35	47.07	14.10	-81.67	77.67	-95.60	-9.70	-87.01	-64.69	27.89	-62.40
22	-81.40	-18.79	32.00	-51.75	53.91	-27.28	-71.25	80.49	-96.15	31.13	66.33	85.83	-16.23	52.62	-53.61	-39.45	-33.90	13.46	52.69	-7.84
23	66.59	29.12	16.53	4.73	-28.82	42.04	-80.74	14.51	95.54	-46.50	66.02	-48.48	69.58	16.09	-91.09	57.39	99.93	-16.15	2.90	45.51
24	-51.06	14.28	95.98	9.23	-61.48	24.22	71.30	-37.92	-6.57	69.19	-59.76	78.16	-15.50	15.30	-97.63	-23.20	-34.36	-29.29	-73.73	49.46
25	-91.22	48.07	16.06	69.76	-20.40	31.52	98.12	-71.64	21.54	90.02	-30.96	-59.32	37.70	47.67	-0.65	-41.32	71.72	-75.95	-33.03	-91.71
26	46.86	78.19	-65.76	-0.45	72.74	94.55	-92.49	7.96	42.70	48.15	72.22	-35.73	-30.31	21.53	-71.60	64.13	95.80	-25.72	0.89	36.76
27	70.23	-7.65	3.63	73.71	-85.28	22.36	-26.05	0.22	7.57	85.20	-58.06	41.71	2.61	17.40	-24.43	-52.14	-32.69	85.80	-84.50	67.63
28	-66.75	-10.60	70.48	-13.53	-7.87	-97.27	-82.39	31.13	-55.88	-58.60	-18.70	50.15	-33.76	-75.15	-53.49	-3.65	-6.04	-30.89	-82.04	-19.66
29	-66.55	60.10	70.28	89.26	2.42	42.32	-87.51	-76.78	-72.68	-15.00	-50.08	-73.29	87.49	2.49	27.95	1.38	48.84	-66.30	25.42	-75.96
30	-31.26	-48.49	-75.78	40.39	37.86	-13.63	74.55	-65.83	19.99	-66.36	-30.64	-64.80	72.89	39.26	-4.58	-39.77	-79.09	-18.35	54.74	-5.47
31	-51.69	61.61	75.08	60.14	9.16	47.54	-32.32	7.52	-57.19	-4.74	-18.42	74.04	-57.80	46.93	83.80	-90.24	67.13	-79.79	-80.26	-90.02
32	-89.89	15.77	-2.02	-70.93	-17.41	66.81	9.41	-19.65	-89.17	-55.98	-42.46	77.19	14.06	-29.95	97.54	-79.64	39.93	-30.11	44.51	24.16
33	-98.65	-77.53	-65.81	-62.95	-77.73	99.79	31.06	-5.97	65.50	65.71	97.39	3.36	-68.43	-34.14	-6.82	89.80	-77.84	56.30	36.78	46.98
34	85.10	34.31	48.28	-72.23	92.84	-41.47	-48.17	-75.94	-98.68	42.79	-62.51	-86.56	58.39	-84.39	19.34	5.91	84.61	69.00	4.00	98.09
35	92.38	-10.34	-83.39	-63.57	-29.09	81.24	-86.59	76.98	-54.32	-35.57	-68.22	-45.21	-22.66	-87.09	-6.71	-61.50	-84.65	-97.56	72.42	33.93
36	45.46	29.77	-75.39	-66.16	99.93	-29.83	-18.98	-52.82	89.77	19.61	12.41	-34.75	54.93	-64.67	5.04	-60.19	57.82	22.50	-77.15	-50.88
37	20.88	10.24	-42.63	54.18	92.39	-17.84	36.09	81.90	23.68	-58.46	-16.10	1.98	24.60	-16.92	-31.68	49.27	95.03	67.81	-6.39	44.32
38	-92.00	98.73	-25.64	61.05	63.13	64.39	-70.18	-31.24	-29.85	5.74	-59.83	71.40	6.38	-99.36	59.86	88.45	-14.45	-15.09	-83.41	-71.87
39	-19.22	12.32	90.56	-85.91	-78.03	9.63	-21.03	-35.12	-8.06	82.30	-10.05	-74.93	83.70	85.87	-51.62	-69.60	33.16	47.28	-20.43	-50.82
40	91.47	-34.16	49.94	62.58	82.85	17.01	56.18	93.59	93.69	-65.26	-41.00	58.52	-77.39	87.86	11.29	52.77	-38.69	-26.56	42.72	89.94
41	-39.50	60.29	-26.42	-50.73	47.31	87.01	46.80	-2.46	-8.25	4.72	-23.63	-17.73	-60.20	83.89	84.06	1.80	46.73	62.84	93.51	-8.29
42	-14.06	1.42	-6.98	-45.01	76.52	-24.89	-86.34	58.66	-40.13	84.75	27.96	69.29	-84.94	-69.70	-56.41	-25.70	-1.71	-32.00	6.82	-55.41
43	56.14	-53.30	86.25	13.15	-77.65	75.65	16.91	-96.23	-79.04	-54.69	87.50	83.74	-52.58	-97.88	57.42	73.67	-99.06	98.81	61.83	88.15
44	22.90	-93.81	79.38	15.20	-46.69	-52.87	-68.08	16.70	-89.28	63.59	32.46	-84.24	-72.70	-36.29	-70.91	-86.67	-17.93	-81.63	-38.17	-67.14
45	-23.07	-85.37	-60.93	-23.18	-84.48	-99.11	53.95	-22.86	9.64	8.75	89.01	34.43	-4.66	-46.58	47.37	-87.06	-38.02	-91.44	44.76	31.83
46	-8.77	65.35	-52.70	-81.24	82.28	-83.06	-93.58	87.65	-79.40	-96.33	-44.85	-32.50	7.58	3.91	6.74	-60.74	-0.58	95.49	-22.38	91.25
47	-45.03	-3.02	14.74	42.22	-79.13	23.88	72.00	-70.22	-73.76	2.34	2.08	-27.69	-61.85	35.79	34.52	-37.78	92.61	-43.17	-38.12	99.66
48	49.23	-45.90	36.96	3.95	-93.30	52.59	-88.12	7.92	-69.44	-22.04	-23.30	68.77	-30.71	91.32	86.08	-82.68	-17.83	-9.15	-79.87	-97.18
49	-11.12	-62.41	-52.05	78.93	83.16	45.58	99.18	-79.74	95.80	-49.78	8.20	-12.04	-60.86	88.91	39.06	-48.89	-70.88	-59.79	8.99	-42.11
50	-78.84	17.97	-22.30	82.36	94.01	-23.96	83.07	-99.29	-91.32	32.63	53.10	52.39	-99.59	-32.22	-15.77	31.83	-54.07	78.14	-24.59	43.57

The table is continued on the next page.

This table is continued from the previous page.

	POI_01		POI_02		POI_03		POI_04		POI_05		POI_06		POI_07		POI_08		POI_09		POI_10	
Tour	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
51	-60.67	83.91	-23.38	-20.32	-88.56	-72.47	6.04	-73.62	-81.72	90.74	92.76	66.38	23.00	32.25	14.05	47.66	-9.00	58.73	2.23	54.57
52	60.47	-94.79	27.64	-95.46	24.59	19.38	-53.49	-58.03	-35.25	45.35	45.93	3.96	-48.32	6.31	-36.84	59.18	-50.03	41.02	-18.66	-83.77
53	58.46	-13.83	-96.97	6.89	-3.92	-90.09	59.14	-47.75	-9.65	-19.56	76.55	24.30	11.94	68.55	46.14	-88.90	50.33	81.78	-64.91	-76.01
54	90.51	-8.01	18.19	8.22	97.37	5.39	-53.46	38.30	27.69	-77.69	23.21	84.90	89.88	67.58	71.04	86.13	-23.14	9.27	39.64	53.50
55	-89.49	-73.82	69.84	64.90	-67.30	-93.48	95.54	-69.35	19.62	13.10	-54.09	-84.73	-97.97	-11.08	27.87	82.21	-71.59	46.46	-76.13	-58.34
56	54.49	-3.40	66.08	63.80	-14.08	89.31	-3.87	32.31	-74.35	-63.66	-9.97	57.70	-10.85	-47.18	98.73	-58.97	24.91	-27.90	-23.67	-64.69
57	-52.65	18.48	18.34	54.51	-94.80	-13.29	96.11	-58.40	57.81	-28.89	-75.37	83.02	-56.59	72.30	16.20	33.17	56.73	84.43	-58.69	8.12
58	3.80	2.43	-84.10	17.56	-95.98	1.57	99.02	-18.54	70.32	-31.24	-43.10	75.43	-61.34	69.90	-94.29	-69.72	5.89	35.99	33.66	3.39
59	-63.76	-17.30	-19.03	-21.00	21.29	-61.40	-66.82	54.17	-93.92	67.47	-1.12	44.69	-78.03	-45.51	-47.77	7.52	21.75	-32.73	8.20	-99.32
60	99.45	-72.30	-1.79	86.68	-99.09	73.21	-30.53	-17.22	-1.04	25.74	25.96	-56.80	72.79	-52.94	77.12	33.93	-56.08	96.86	12.08	78.14
61	69.96	-18.49	64.57	72.26	-63.25	55.24	-82.25	-54.51	-93.59	-48.83	-15.37	-60.85	-28.97	-80.13	-41.86	-81.01	-0.75	-20.61	57.01	38.02
62	-98.08	54.21	72.78	-85.12	21.97	67.75	65.03	50.15	66.74	98.98	-11.89	-27.06	89.11	71.64	43.35	-29.57	3.69	32.04	-53.97	75.60
63	-24.73	-74.54	76.96	56.13	38.84	13.39	-13.86	29.99	8.46	95.06	8.70	78.70	16.92	21.85	76.92	98.22	-49.73	28.40	90.08	91.86
64	-93.30	-85.87	-17.94	-70.59	-39.80	-53.93	-45.29	-74.70	-79.50	74.42	75.38	50.09	-19.89	5.67	-50.83	-0.92	33.85	-71.82	-48.98	51.94
65	29.55	-85.01	18.09	93.36	61.64	46.66	85.69	92.33	9.95	63.28	94.67	-97.65	5.10	41.81	-19.41	-21.75	-23.77	34.40	14.66	-63.58
66	-10.34	-6.71	83.36	12.86	-91.23	15.48	76.26	-36.59	97.79	-17.69	4.44	-58.96	-73.99	43.70	-99.86	59.09	-2.35	53.94	-52.44	-11.54
67	9.91	62.46	61.42	-90.82	-9.98	27.54	98.60	89.43	-54.69	17.75	-57.14	-46.57	-90.13	-7.89	38.97	-14.44	-77.85	30.04	-2.33	-87.89
68	-9.76	-96.93	-11.84	-30.32	-9.65	-12.71	-18.91	-88.59	-84.98	27.72	86.89	75.48	-84.48	-12.21	82.95	-92.52	-92.37	19.83	-93.95	62.49
69	10.50	-57.86	-96.23	-29.74	-40.62	-82.76	-27.85	-98.11	-62.99	-80.22	6.83	4.95	-45.24	-65.01	19.93	-85.01	60.73	25.74	71.67	56.69
70	95.85	-11.40	68.62	-49.75	-25.20	86.20	-70.88	78.87	-17.77	-3.98	-74.98	-53.85	71.47	-24.53	-7.36	-53.50	-55.57	68.96	-86.23	67.26
71	60.75	-8.84	-82.06	93.88	-22.20	86.63	0.96	7.07	63.12	76.48	-29.00	63.92	-84.46	-60.95	64.85	-99.06	24.54	-14.49	52.98	52.04
72	62.75	-55.98	-89.59	-43.28	-64.43	91.55	-46.84	96.43	-30.72	-22.27	65.96	69.53	35.31	-42.41	37.00	67.47	-66.41	76.00	31.89	-76.62
73	43.40	-27.23	17.86	76.37	26.16	-46.60	-6.09	17.53	40.03	-13.36	-99.28	-16.81	88.75	2.12	-8.01	-91.19	70.64	-31.47	-49.51	-58.32
74	81.97	-6.22	-84.14	80.67	98.11	4.33	-23.05	-3.72	-22.68	72.98	-50.25	53.46	-90.78	-63.76	92.12	-61.34	-55.42	-91.66	24.27	49.15
75	-55.62	-67.62	14.34	37.15	-40.26	-72.64	-23.99	38.85	-31.67	-59.03	-15.27	45.87	-51.27	-92.24	-73.34	-19.21	7.61	-61.62	80.84	31.76
76	-33.15	-48.21	87.85	37.82	-54.40	93.80	20.80	82.97	-43.57	-37.38	68.75	85.78	-15.52	-78.14	10.20	24.42	-57.60	-79.46	-34.65	40.91
77	76.91	9.15	-62.51	82.96	89.49	-19.65	-56.80	-39.77	66.93	-13.40	10.13	-50.08	-27.75	85.65	-34.77	-31.89	53.27	71.63	-12.84	-80.95
78	40.62	79.95	-88.77	-92.04	18.61	38.65	12.31	4.36	85.93	-52.71	-72.35	23.27	76.57	29.28	-14.15	62.56	55.82	-73.46	-97.35	48.82
79	89.62	66.31	89.53	-95.47	9.84	43.53	18.97	14.45	47.25	-96.27	45.02	-69.43	-56.23	-59.95	-65.78	-96.34	17.46	-22.57	8.13	4.48
80	-80.87	-33.12	-6.52	71.87	-5.27	-72.14	-2.35	25.76	95.73	67.87	-23.54	-69.21	-61.28	68.17	-49.71	-80.50	17.17	48.99	37.16	17.66
81	-2.01	36.59	-21.05	64.75	94.01	-42.57	-40.69	-5.45	-91.01	51.56	87.95	-6.59	39.07	87.51	59.17	-34.22	-80.99	-68.50	-50.80	83.42
82	-68.80	25.05	-17.98	26.50	15.82	90.76	-73.82	-0.33	51.59	60.82	-29.23	71.09	-93.89	-54.71	26.99	55.89	-54.34	-99.62	42.15	-49.96
83	49.88	37.15	93.45	-94.74	-64.92	-2.07	-30.09	-35.34	-84.27	17.80	39.89	-24.75	-14.97	67.21	8.90	-66.34	14.32	47.11	-98.64	67.70
84	-36.80	63.09	99.52	-59.52	-3.73	33.16	-11.21	-51.68	12.43	79.26	-73.81	77.87	-40.18	-19.44	-0.30	21.82	-14.90	-95.45	-77.37	0.79
85	71.69	-45.46	-50.29	-98.50	66.76	12.23	-81.72	-83.22	29.78	-82.44	-95.98	87.23	-54.92	-72.47	55.18	67.94	84.60	57.08	-81.65	-49.59
86	-24.85	-29.40	-59.43	7.83	62.52	-8.07	45.92	-61.91	15.08	-76.84	55.93	17.85	-15.64	-38.80	-45.43	36.71	-7.82	-69.02	-98.95	-35.67
87	-72.89	-48.85	-45.76	88.10	-18.79	-69.94	-18.12	26.44	-2.82	84.73	94.29	-32.15	61.05	76.32	-97.41	34.73	-26.00	-2.33	-37.91	27.35
88	-36.79	69.93	-51.42	87.01	45.96	76.03	-5.58	7.39	19.54	43.51	-50.21	42.38	-13.62	42.86	19.00	76.62	-68.83	-88.13	54.17	-77.74
89	32.87	-93.92	90.68	48.80	42.21	36.13	-74.57	-23.38	95.87	-40.25	74.66	-1.94	34.98	-24.75	97.82	-0.22	-13.89	-89.13	-42.25	57.97
90	72.27	-79.67	-2.60	-30.67	67.17	46.85	-34.02	96.65	-89.91	-36.76	-15.38	-12.03	84.25	21.02	-91.09	20.93	84.17	-36.85	-2.82	76.23
91	71.50	-91.26	-31.97	-31.57	46.95	38.21	37.76	-30.66	57.70	-80.50	-81.50	49.15	31.53	12.03	-43.89	6.95	-14.40	73.55	53.46	-53.48
92	46.16	-60.19	-60.85	54.97	-94.38	65.81	99.63	-55.34	-87.39	-53.20	81.37	36.71	15.85	-30.93	89.21	-47.50	-13.57	-82.64	46.56	-5.93
93	2.24	71.50	35.56	-70.80	25.40	53.89	3.33	-25.86	32.18	78.28	62.33	-11.80	-43.98	-22.27	-7.13	-95.33	16.59	-61.12	-36.93	-41.59
94	63.46	-97.35	57.45	46.33	87.32	84.67	61.58	-84.74	-97.53	-82.21	-72.34	-87.59	-25.06	-42.85	67.26	73.26	-19.75	20.96	-91.73	-64.99
95	86.34	-99.98	61.52	61.56	-5.73	-5.74	-82.74	-44.31	-22.41	83.33	-41.00	-96.68	49.84	66.51	23.13	19.81	50.14	83.62	-71.17	9.00
96	54.14	-73.27	-94.01	-28.08	41.12	14.40	0.56	-71.34	40.52	-73.88	30.57	-7.87	3.30	-2.71	68.17	-23.95	-6.57	96.13	-75.39	48.55
97	-80.55	-95.29	71.71	-19.80	-4.93	-9.64	-68.20	51.59	30.40	-59.95	54.25	-97.34	-60.22	99.72	-36.05	-38.71	99.51	99.78	10.46	-50.81
98	1.97	-7.49	0.86	8.26	-62.16	45.33	34.41	-73.84	-18.25	36.40	88.38	62.23	-1.01	62.57	-0.83	36.93	-47.17	-9.75	-37.51	25.55
99	-13.11	54.34	69.71	20.51	10.85	56.50	76.38	-44.13	-7.83	35.66	1.48	-54.38	26.75	91.09	-16.77	-62.39	18.37	27.27	76.27	38.66
100	51.90	-12.14	-85.03	70.75	-53.66	58.47	96.08	-74.67	7.78	49.31	-10.21	-35.53	-72.84	-23.23	91.34	-28.45	21.67	-55.09	22.07	82.63

Appendix C. Acronyms

ACC	Adaptive Cruise Control
ACPS	Autonomous Cyber-Physical System
AE	Autonomy Engine
AI/ML	Artificial Intelligence / Machine Learning
AR	Assurance Refinement
ATC	Air Traffic Control
AUV	Autonomous Underwater Vehicle
AV	Autonomous Vehicle
CACC	Cooperative Adaptive Cruise Control
CGI	Computer-Generated Imagery
COLREGS	Collision Regulations
CONOPS	Concept of Operations
CPS	Cyber-Physical System
DESCRETE	Digital Environment for Simulated Cyber Resilience Engineering, Test and Experimentation
DTP	Departure Tour Plan
DAC	Dynamic Assurance Case
ETE	Estimated Time Enroute
ETP	Enroute Tour Plan
FV	Formal Verification
GPS	Global-Positioning System
HC-N	Harbor Control Navigation Service
HC-T	Harbor Control Touring Service
HCS	Harbor Control Services
IFR	Instrument Flight Rules
IMARA	Iterative Mission-Assurance Refinement Analysis
INS	Inertial Navigation System
IPDRR	Identify, Protect, Detect, Respond, and Recover
JHU	Johns Hopkins University
JHU/APL	Johns Hopkins University Applied Physics Laboratory
LOI	Level of Interest
LOS	Loss of Separation

MAAUV	Mission Assured Autonomous Underwater Vehicle
MBSE	Model-Based Systems Engineering
MO	Mission Objective
MRMO	Minimum-Required Mission Objective
PB	Physical Boundary
PC	Personal Computer
POI	Point of Interest
PRNG	Pseudo-Random Number Generator
RB	Response Boundary
RM	Resilience Module
RPY	Role, Pitch, Yaw
RSZ	Reduced-Separation Zone
SAS	Self-Adaptive System
SB	Separation Boundary
SDF	Simulation Description Format
SOE	Safe Operating Envelope
T-SUB	Tourism Submersible
TD-Max	Maximum-allowed tour duration
TD-Min	Minimum-required tour duration
TPM	Trusted Platform Module
UAS	Uncrewed / Unmanned Aerial System
UUV	Uncrewed / Unmanned Underwater Vehicle
WB	Warning Boundary
WZ	Warning Zone
XML	Extensible Markup Language
YAML	YAML Ain’t Markup Language

Appendix D. Curriculum Vitae

Karl A. Siil received his B.E. degree in Electrical Engineering from the Cooper Union in 1984. He received his M.E. degree in Electrical Engineering from Manhattan College in 1987. Since 1987, he has worked for AT&T Bell Laboratories, SRI International, and the Johns Hopkins University Applied Physics Laboratory (JHU/APL), in various engineering capacities. In addition, from 2000 to 2007, he was a founding member of Lumeta Corporation, serving as chief architect.

Over his career, he has received a variety of awards and forms of recognition including four patents, appointments as a senior member of the ACM and IEEE, and Principal Professional Staff member at JHU/APL.

He enrolled in the Doctor of Engineering program in the JHU Whiting School of Engineering in 2019. His research interests include Model-Based Systems Engineering, Modeling and Simulation, Trusted Systems and the application of these technologies in embedded environments. He enjoys family vacations, travel, general aviation, puttering with computers and software, and professional football and hockey.