Interoperability for Autonomy

Kara Combs
Air Force Research Laboratory
Kara.Combs.1@us.af.mil

Trevor J. Bihl
Air Force Research Laboratory
Trevor.Bihl.2@us.af.mil

James Pennington
Ohio University
Jp013718@ohio.edu

Abstract

Autonomous systems aim to augment human capabilities with machine-based decision-making in the absence of a user. Ideally, autonomy hardware and software would be modular, having the ability to swap components in and out as needed based on necessary capabilities. However, many legacy systems in use utilize proprietary software with specific standards and components, reducing the system's ability to be interoperable. Currently, the literature's definition of interoperability is vague and often mistaken for other similar terms. We distinguish the uniqueness of interoperability and codify it through a taxonomy. Next, we extend this framework to understand autonomy and its hardware/software components through a proposed unified autonomy stack. We then evaluate the similarity between four autonomy architectures based on 29 stack components that are later presented in the "interchangeability matrix." Thus, we demonstrate the necessity to unify autonomy hardware/software under the proposed taxonomy in the development of future autonomous systems.

1. Introduction

Autonomy is the ability to make decisions or act on one's own (Trivellato, Spiesses, & Zannone, 2009). Autonomous systems such as robots, self-driving cars, and unmanned aerial vehicles have been incorporated into daily life to assist humans (Lum & Heer, 1989). The ability to connect these individual autonomous systems to work has been the goal of the Internet of Things (IoT) (Li, Xu, & Zhao, 2015). However, before individual autonomous systems can cohesively work together, they must be interoperable (Trivellato, Spiesses, & Zannone, 2009).

There are many similar, but technically different definitions of interoperability; however, essential to all of them is the ability for individual systems to communicate with one another (Ford, Colombi, Graham, & Jacques, 2007; Kasunic, 2001). In the broadest terms, there are two types of interoperability: operational and technical (Kasunic, 2001). Operational interoperability is focused on enterprise/organization-based interoperability often involving systems of systems and humans. Whereas technical interoperability focuses on the individual systems at the technical level regarding data and information exchange between systems. Since then, types of interoperability have been further split into more levels namely such as business, processes, services, and data (Chen, 2006) and organizational, operational, systems, and technical (Moon, Fewell, & Reynolds, 2008). At each level, there are varying degrees to which these systems are considered interoperable. This has been the subject of research aimed at the quantification interoperability and metrics therein (see (Mensh, 1989)).

Though systems cognitively connected, dubbed "smart objects" in (Kaisler, Money, & Cohen, Smart objects: An active big data approach, 2018) are applicable and important to interoperability, this paper is focused on the technical interoperability of autonomous systems. As such, the focus of this paper concerns systems that physically or otherwise transmit data and information to one another. The primary research questions are:

RQ1) Can a systems-level understanding of autonomy be developed and conceptualized into a stack?
RQ2) What are ways to conceptualize and describe the interoperability of systems and issues therein?
RQ3) Can a high-level taxonomy of interoperability, separate from autonomy be developed?
RQ4) Can this taxonomy and system stack of autonomy be used to influence further collaboration and integration?

The contributions of our paper are thus, each corresponding to the respective research questions: (1) we have produced a unified autonomy stack to guide the development of future autonomy systems, (2) provide definitions of interoperability-related concepts and current barriers to interoperability, (3) creation of an interoperability taxonomy for autonomous systems, and (4) evaluate the interoperability of four popular autonomy architectures. First, in Section 2, this paper provides an overview of autonomy systems and stacks before describing our proposed autonomy stack. In Sections 3 & 4, we survey the literature to understand

URI: https://hdl.handle.net/10125/106486
978-0-9981331-7-1

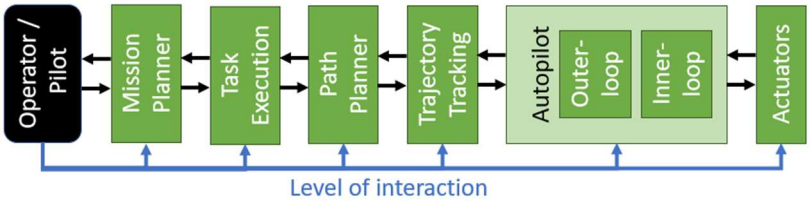

Figure 1(a). Planning stack, for autonomy, rotated and modified for space, from (Kingston, 2017)

Figure 1(b). Autonomy stack from (Gulzar, 2022)

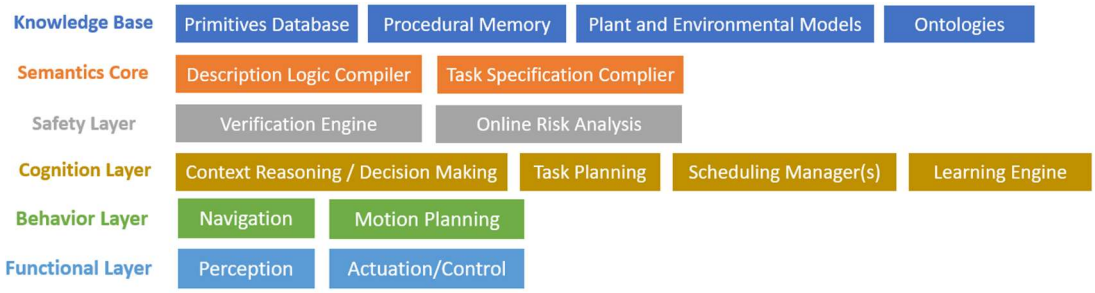

Figure 1(c). Autonomy stack from (Barad, Martinez Luna, Dentler, & Olivares Mendez, 2021)

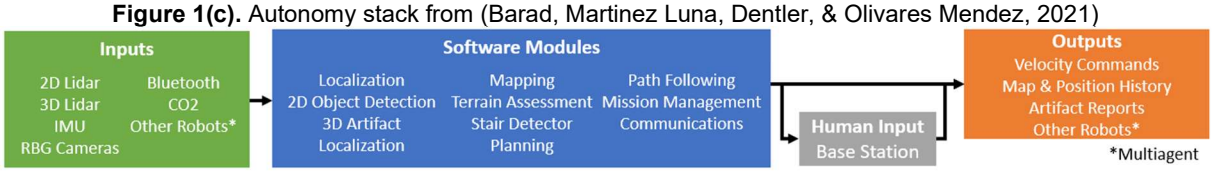

Figure 1(d). Autonomy stack, modified for space, from (Biggie, et al., 2023)

more about interoperability and the current limitations of interoperable autonomous systems, which guide the development of our taxonomy. Then, in Section 5, we compare the interoperability of four autonomy architectures based on the stack in Section 2. Finally, in Section 6, we discuss future work and conclusions.

2. Autonomous Systems

As described by (Bihl & Talbert, 2020), autonomy describes a system with intelligence-based capabilities, allowing it to respond to situations that were not preprogrammed or anticipated in the design. Such systems include biological intelligence and potentially artificial intelligence applications. To be autonomous, a system needs perception, decision-making (including control), and actuation (including physical/localization or data creation) (Gan, et al., 2022) (Bihl & Talbert, 2020). Notionally, this is divided into "autonomy in motion," i.e., a robot, and "autonomy at rest," i.e., a system that processes data and develops its interpretation (Zacharias, 2019). Such autonomous systems must have a degree of self-government, and self-directed behavior and have three general characteristics of cognition (Bihl & Talbert, 2020).

2.1. Autonomy Stacks

Autonomy, in all its embodiments, involves a complex interaction of systems and subsystems. These can include visual stimuli being converted to signals in a neuron, which go to a visual cortex for further processing in the brain and so on, as well as a sensor on a robot that converts sensed data to digital information which is then converted to a data standard for further processing by a computer processor. Notably, in all cases, data is converted to different formats, and/or processed, by different systems and subsystems of an autonomous system. Technology ("tech") stacks allow for the ability to have data consistently flow between hardware and/or software "layers" (MongoDB, 2023). The advantages of using a tech stack are ease of scalability, focused areas of expertise, initial understanding of requirements, ability to customize, large support community, and utilization of best security practices (MongoDB, 2023). There are several different types of tech stacks including:

- Software: Characterized by exclusively containing software components (e.g., MEAN, LAMP, etc.)
- Planning: Specific type of software stack utilized in robotics (e.g., Figure 1(a)) that includes behavior aspects, local plans, and sometimes mission plans (Kingston, 2017; O'Kelly, et al., 2016)
- Protocol/Network: Used to guide communication between network entities (OSI, TCP/IP, etc.)

One of the most widely used technology stacks is the OSI model, which expands upon the earlier TCP/IP model (see the comparison in Table 1) (Zimmermann, 1980; Meyer & Zobrist, 1990). The OSI model is a protocol stack that focuses on the communication between hardware and software components. OSI has been investigated in terms of interoperability but it was limited to specific hardware/software systems (see (Wood, Harvey, Linderman, Gardener, & Capraro, 2012)).

Table 1. OSI vs TCP/IP Models

OSI Layers	TCP/IP Layers	Description
Application	Application	Receives data and presents to end-user
Presentation		Curation of the presentation of data
Session		Com. between two devices/computers
Transport	Transport	Data transmission
Network	Internet	Com. between two indirectly connected hardware elements
Data Link	Network Access	Com. between two directly connected hardware elements
Physical		Hardware elements

Recently, a 7-layer AI technology stack model has been proposed in (Tsaih, Chang, Hsu, & Yen, 2023) consisting of the following layers:

1. Infrastructure – hardware components
2. Platform – operating system (OS), environment, programming language, etc.
3. Framework – programming libraries, pre-built packages/software, etc.
4. Algorithm – customized algorithms for task(s)
5. Data Pipeline – data pre-processing
6. Service – application programming interface (API)
7. Solution – specific AI tool for a given task(s) depending on the need.

The AI technology stack focuses on providing "AI-as-a-service" rather than on the development of autonomous systems, which requires other aspects such as communications, behaviors, and plans. Though these foundational software, planning, and protocol stacks exist, there is still a need for a go-to autonomy-specific one, e.g., an "autonomy stack." Autonomy stacks generally aim to standardize components of hardware and software of autonomous systems. The literature presents a few examples as shown in Figure 1(a-d). The stack developed by (Kingston, 2017), shows the information that is passed between layers and considers the overall mission objective. The limitation of this stack is that many layers in series are required to achieve commands being sent to the Autopilot, which is the layer that ultimately makes control decisions. Ideally, these layers would be unified under a single algorithms layer with which the operator/pilot does not directly interact as much as what is currently proposed. In (Gulzar, 2022) (visualized in Figure 1 (b)), a stack is presented to demonstrate the flow of information from the real world and how it contributes to the eventual planning and control of an agent. This stack, however, considers only that information flows in a single direction and does not consider interoperability between autonomous systems. Figure 1(c) presents a stack created by (Barad, Martinez Luna, Dentler, & Olivares Mendez, 2021), which outlines the different capabilities that an autonomous agent must possess to truly be autonomous. The limitation of this stack is that it does not consider the flow of information between the layers. Finally, the stack developed by (Biggie, et al., 2023) in Figure 1(d), shows a clear transition of information as it is processed by the agents to plan and act. However, this stack is not generalized and relies on software and sensors specific to the application therein.

2.2. Developing a Unified Autonomy Stack

A new autonomy stack must address the limitations of those currently available in the literature. First, it must apply to truly autonomous agents. To be truly autonomous, an agent must be capable of responding to unexpected situations without human input (Bihl, Cox, & Jenkins, 2018). Consequently, human input should be minimal, only occurring to provide mission requirements. As shown by the developed stack in Figure 2, the human only interacts with the environment layer to provide information about mission requirements. The next limitation of many stacks, including those shown in Figure 1(a,b,d), is that information related to the perception of the real world cannot be shared between agents but only directly to the actuation phase.

To address these limitations, the authors developed the autonomy stack in Figure 2. Here, we consider that

information flows in both directions through each layer. At the bottom of the stack, information is sent and received by interoperable systems. Another advantage to our stack is its ability to apply to "autonomy in motion" as well as "autonomy at rest," with the difference between the two in the context of actuation and sensing (a physical environment for the former and a more virtual space for the latter). Finally, the stack must be generalized. As a given system may have any subset of sensors and/or actuators available, the stack must not limit its utilization of them. This allows for robustness and wide application of the stack.

Going up the stack in Figure 2, we begin with the hardware layer. This layer encompasses all physical

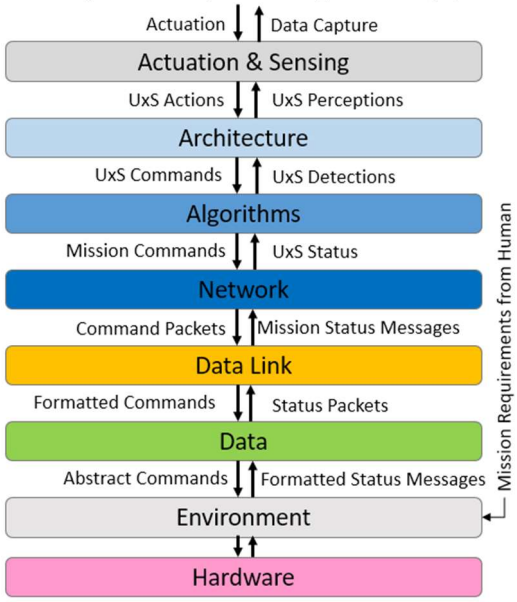

Figure 2. Developed Unified Autonomy Stack

components that make up the computer, i.e., hard drive, motherboard (CPU/GPUs), and RAM. Next up the stack is the environment layer, which handles the operating system (OS)/container system and interaction with the human operator. This layer specifically considers proprietary OS, open-source Linux OS, and/or containerization software. This layer sends messages generated by a human (AKA "mission requirements") in the form of abstract commands to the data layer.

The data layer's purpose is to convert the abstract commands to formatted message types. Several data formats/standards are relevant to autonomy such as geographic data, general structure/protocol, and NATO standards. The data layer takes a command that would be human-readable, extracts the important information, and stores that in a computer-readable file. That file is then passed to the data link layer, where the physical link between systems is established. The data link layer considers communication standards such as SATCOM and relevant recommended standards (RS). The formatted command from the data layer is then broken into packets and sent to the network layer. At the network layer, the protocol by which the command will be sent is determined before the command is sent to the algorithms layer. These communication standards include widely used file sharing and communication ones such as TCP/IP and HTTP(S) as well as some lesser-known ones.

At the algorithms layer, mission commands are assigned to a particular agent, referred to as an unmanned system or UxS. The algorithms layer involves heavy computation as it handles navigation and task planning of the UxS as well as building a model of the surrounding environment to reliably assign tasks to the optimal UxS. There are several subsections of the algorithms layer (from the bottom up): program (open vs proprietary), control (actuator and waypoint commands), behavioral (navigation; motion planning), cognition (learning engine; scheduling manager(s); task planning; context reasoning/decision making), and knowledge base (ontologies; plant and environment models; procedural memory; primitives database).

UxS commands from the algorithms layer are passed to the architecture layer. This layer ensures the cooperation of agents through interoperability. At this layer, UxS commands become actions and are sent to the UxS (i.e., a command such as "Move to this waypoint," becomes "Orient this direction and increase throttle"). Finally, the actuation and sensing layer represents the UxS, where actions are translated to motor controls.

Moving down the stack acts much the same as moving up the stack, except rather than considering actions to be performed, we consider information regarding the environment. Data is first captured by various sensors on the UxS. This data becomes perceptions of the environment at the architecture layer, which then interprets those perceptions for detection. From these, the algorithms layer builds the status of an individual UxS. Over the network layer, the status of each UxS is collected and summed to develop the status of the mission. Continuing down the stack, those are broken into packets and formatted messages so that commands can be developed to send back up the stack.

3. Interoperability

An autonomy stack provides an understanding of the data flow and general needs of an architecture and is a

first step to understanding how a system works. However, while general architecture or embodiment of architecture is useful, it lacks a specific indication of the degree of interoperability between technologies in the stacks. While an autonomy stack provides an understanding of how a system works, more fidelity is needed to create an autonomous system able to collaborate with external systems. Beyond the physical or digital connections providing interoperability, there are non-technological interoperability types that can be considered as well (Kaisler, 2005).

Although used synonymously, it is a misconception that portable, interchangeable, compatible, integrated, and interoperable are equivalent to one another (see (Kolb & Wirtz, 2014)). Within Figure 3, a "system" is used regarding computer software, hardware, component, service, or entity. Whereas, an environment is a more encompassing term referring to a platform that houses or hosts multiple systems, or more simply a "system of systems."

Portable systems can migrate between two environments (Lewis, 2013). One series of standards defines portability as the "degree of effectiveness and efficiency with which a system ... can be transferred from one hardware, software, or other operational or usage environment to another" (ISO/IEC 25000, 2022). These systems ideally have a plug-and-play-like ability to make these exchanges seamless with little to no adaptations (Al Ridhawi, Otoum, Aloquaily, & Boukerche, 2020). Portability is a characteristic of one individual system and should always be an underlying characteristic of interchangeable systems.

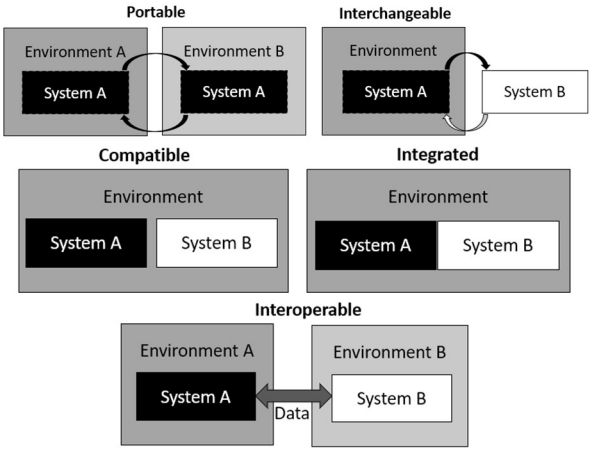

Figure 3. Conceptualization of Related Interoperability Concepts

Merriam-Webster defines interchangeable as the "[capability] of being substituted in place of one another." Interchangeability has been equated to "cloning" wherein all the abilities of one system are reproduced in another separate system (Buono, 2005). Interchangeable systems ideally have a plug-and-play ability like portable systems; however, what differentiates the two is that the interchangeable systems may not perform the same tasks/processes as portable systems are expected to. Since interchangeable systems are expected to perform the same tasks with the same inputs/outputs, usually there is no need to consider whether the systems are compatible, integrated, or interoperable even though they might be.

Compatibility can have multiple meanings. Systems, in general, may be deemed compatible based on the consistency between their executed actions during a process (Taberko, Ivaniuk, Shunkevich, & Pupena, 2020). According to Dictionary.com's definition of compatibility regarding computer systems, it is the "ability to be used … without the need for special adaptations." The key characteristic of two compatible systems is the ability to perform their respective tasks/processes without disrupting the other one (Hajim, 2021). The most basic level of compatibility does not require two systems to interact with one another in any way. Though this definition seems broad and encompassing, higher degrees of compatibility can be better described by being integrated or interoperable.

Integrated systems are those designed or adapted to operate within the same environment, within which they are contained (Hasselbring, 2000; Mohamed, Mahadi, Miskon, Haghshenas, & Adnan, 2013; Testing Standards Working Party, 2005). This allows the individual systems to operate as one within the environment, which is the distinguishing characteristic of integrated systems (Wilder, 2019; Smith, 2018). This requires compatibility between the two systems which all functional integrated systems have. However, the converse is not always true. The primary benefit of integrated systems over most interoperable systems is minimal adaptation in the upgrading or development of the individual systems; however, the integration process can be costly and time-consuming.

According to ISO 25964-2:2013, interoperability is defined as "the ability of two or more systems or components to exchange information and to use that information that has been exchanged" (ISO, 2013). IEEE defines interoperability as "the capability of two or more networks, systems, devices, applications, or components to externally exchange and readily use information securely and effectively" (IEEE, 2018). Regardless of specifics, at the heart of these definitions are two systems that can effectively communicate with

one another (Testing Standards Working Party, 2005). The key characteristic of interoperable systems is the ability to effectively communicate through the exchanging of data typically through a "translating" middleware despite being different systems often in different environments (Mohamed, Mahadi, Miskon, Haghshenas, & Adnan, 2013; Asif & Webb, 2015). Many integrated systems are interoperable in the sense they frequently receive and send data to one another because the integrated systems act as a single unified system. This has confused the literature where integrated used to mean interoperable (as done in (Mohamed, Mahadi, Miskon, Haghshenas, & Adnan, 2013; Asif & Webb, 2015)).

4. Developing Measures of Interoperability

There has been some research on the quantification of interoperability. This research was first focused on addressing common barriers to interoperability as it pertains to computer systems. Popular interoperability measures are categorical with the ability to classify a system into different tiers of interoperability.

4.1 Barriers to Interoperability

Despite interoperability being a straightforward concept, its implementation has proven difficult. Barriers to interoperability have been broken into three categories (Chen, 2006; Chen, 2017):

- Conceptual – concerned with the consistency of the presentation of data/information
- Technological – issues with the ability for systems to communicate with other systems;
- Organizational – difficulties with the delegation of implementation tasks among people/groups.

Issues with obtaining conceptual interoperability largely include discrepancies between consistent definitions (semantics) and the representations (syntax, structure, etc.) across the various levels of interoperability (Cuenca, Boza, Ortiz, & Trienekens, 2015; Chen, 2017). Conceptual interoperability is the foundation of technological and organizational interoperability since if the data are ambiguous, it is impossible to ensure the data keeps a consistent and accurate meaning.

With an understanding of how interoperability is unique from related but different terms, it is important to understand why and how interoperability may be implemented. In 2019, the US Department of Defense (DoD) required the use of the Modular Open Systems Approach (MOSA) for the design and implementation of open systems where feasible (US DoD, 2019). This has led to several "open architectures" where the focus is to reuse previously created solutions to prevent re-inventing the wheel. Some researchers distinguish "open systems" from "open architecture" (Kramer, 2016) (Kovach, Natarian, & Littlejohn, 2021)); however, in this paper, they are used interchangeably with the latter used most frequently for consistency. An open architecture has its standards published and available and these standards discuss design modularity and the interfacing between systems (Grovak, 2021; Kovach, Natarian, & Littlejohn, 2021). Historically, systems were mostly proprietary, or "closed," meaning that their internal properties were kept hidden from anyone not associated with the owning vendor of the system (Dantoni, 2022). Thus, the owning vendors were the only ones with the ability to make modifications and adjustments to the systems. With open architectures, there is no longer a "vendor lock" on these systems as there are no longer restrictions on who can be knowledgeable about the system (Lyke, 2014). This allows the goal of interoperability to be more feasible since the barrier-to-entry has been lowered with the usage of open architectures. This does not mean "plug-and-play," as stated above, but it opens the door for various systems to be able to "communicate" with one another without the need for integration or application-specific middleware. Of course, there are varying degrees of interoperability.

4.2 Current Measures of Interoperability

Since there are varying levels of which systems can be interoperable, several metrics to quantify this relationship have been developed over the years. These evaluation models have been the subject of several literature reviews (Ford, Colombi, Graham, & Jacques, 2007; Rezaei, Chiew, Lee, & Aliee, 2014; Moon, Fewell, & Reynolds, 2008; Leal, Guedria, & Panetto, 2019). Most models are categorical; however, there has been some research involving continuous interoperability values as seen with i-Score (Ford, Colombi, Graham, & Jacques, 2007). The two most famous are the levels of information system interoperability (LISI) and the levels of conceptual interoperability (LCIM). The LISI assessment identifies five levels of interoperability:

0. Isolated - standalone system(s) with only manual data transfer
1. Connected - simple data transfers: text, email, etc.
2. Distributed/functional - data transfer through local area networks (LANs)

3. Integrated/domain - data transfer through wide area networks (WANs)
4. Enterprise - data is freely accessible and distributed throughout the environment.

LISI was further expanded into sub-levels for four attributes that can be evaluated through the assessment: procedures, applications, infrastructure, and data (C4ISR Architecture Working Group, 1998; Morris, Levine, Meyers, Place, & Plakosh, 2004).

The LCIM approach more narrowly focuses on the data exchange between systems into 5 levels (Tolk & Muguira, 2003; Tolk, Diallo, & Turnitsa, 2007):

0. No interoperability/data sharing
1. Technical - common comm. protocol for bits/bytes
2. Syntactical - common data reference model/format
3. Semantic - clear definitions of data and meanings
4. Pragmatic – in-context data and processes are understood by all systems within
5. Dynamic - ability for a system to understand and adapt based on changes in its environment
6. Conceptual - systems are fully aware of one another such that the model has "meaningful abstraction."

However, there are several other categorical models such as the NATO C3 Technical Architecture Reference Model for Interoperability (which now aligns closely with LISI) (NATO, 2021), System of Systems Interoperability (Morris, Levine, Meyers, Place, & Plakosh, 2004) and the Informational Systems Interoperability Maturity Model (Van Staden & Mbale, 2012). Despite being able to think of these classifications as a taxonomy, a formal taxonomy for interoperability has yet to be defined in the literature.

4.3. Interoperability Taxonomy

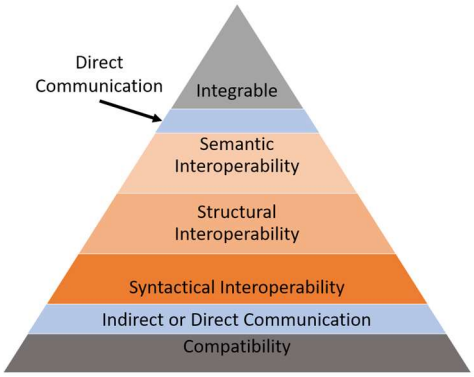

Figure 4. Interoperability Taxonomy

From our understanding of the terms discussed in Section 4, we create a taxonomy combining those terms focused on communication and data information exchanged as shown in Figure 4. As discussed in Section 3, compatibility is a necessity for interoperability (shown as the base of Figure 4). Two systems are not necessarily expected to communicate to be compatible, but their ability to communicate leads to potential interoperability. Communication is usually done through an interface, that is through a translator (indirect); however, it is possible to communicate directly. Interoperability is split into 3 sub-types, each dependent on the previous one: syntactical, structural, and semantic. Syntactical interoperability is based on using common syntax for the communicated data such as common data types. Structural interoperability is the use of a common data structure or format. Together syntactical and structural interoperability is part of the LCIM syntactic level. The final sub-section of interoperability is its semantic form, which requires an understanding of what the data means when passed from one system to another. Finally, through direct communication between systems where they act together as one is when two systems are integrable.

5. Autonomy Stack Case Study

To create an appropriate evaluation of interoperability in the current environment, architectures, where interoperability is crucial to their function, must be chosen to be evaluated. Such a category where interoperability is crucial is that of ground control station software. A ground control station is a computer application that communicates with one or many UAS via wireless telemetry to control them during flight through uploading mission commands (ArduPilot Dev Team, 2023) Many of these applications are commercially available to be purchased or free to install by any user.

Of interest were four autonomy architectures. First, Mission Planner's stable version (released last updated March 2023) is a free open-source ground control station developed for use with UAS running ArduPilot firmware (Oborne, 2023). Second, QGroundControl (QGC) v4.2.6 is another free, open-source application, but it is intended to operate with UAS that uses the MAVLink serial protocol (Dronecode Foundation, 2019). Third, VCSi is a modular suite of proprietary software developed by Lockheed Martin for commercial and military use (Lockheed Martin, 2023). Finally, the fourth architecture was OpenUxAS, or simply UxAS, which is an open-source ground control station developed by AFRL with military applications as the target case (AFRL, 2023). These were selected based on having the most information availability and the ability to run locally rather than through a hosted server.

	A & S		Algorithms		Network		Data Link				Data								Environment								Hardware			Interoperability Score
	Perception	Actuation & Controls	Open Code	Open Data	TCP/IP	UDP/IP	USB	WiFi	Bluetooth	SiK Telemetry Radio	NMEA	SiRF Binary Protocol	STANAG 4586	CoT	JSON	JREAP-C	LMCP	KML	Windows 7	Winows 10	Windows 11	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04	Kubernetes	Docker	GPU Required/ Reccomended	Compatible with 8 GB of RAM	Compatible with 16 GB of RAM	
Mission Planner			X	X	X	X	X	X	X	X	X	X							X	X	X	X	X	X						16
QGC			X	X		X	X	X		X					X				X	X	X	X	X	X		X	X	X	X	17
VCSi				X									X					X	X	X	X						X		X	8
UxAS			X	X	X	X											X					X	X	X						8

Figure 5. Interoperability Analysis Based on Developed Unified Autonomy Stack

5.1. Autonomy Architecture Comparison

Utilizing the autonomy stack proposed in Figure 2 the four architectures mentioned above were evaluated in terms of their interoperability and similarity to one another. An interoperability scoring system was created based on elements of the 7 remaining layers of the proposed stack. This interoperability score was later used to calculate the interchangeability scores.

Starting with the bottom layer, hardware, the architectures were evaluated based on requirements for whether the software required or recommended to run on a GPU and if there was a recommended RAM amount. Needing a GPU awarded one point, as well if it was compatible with 8 or 16 GB of RAM for a total of up to three points for the hardware layer. Moving up to the environment layer, its focus was operating systems (OS) (either proprietary or open source) and the containerized software (Kubernetes; Docker) the autonomy architecture was able to run on. Each of the six OS and the two container software were worth one point each if the architecture was able to be hosted and run on it; therefore, there is a total of eight points that an architecture could earn in the environment layer. Next, the data layer was evaluated for each architecture on its ability to send and receive ten common and popular file types with each file type worth one point if the architecture supported it. After the data layer is the data link layer related to whether the architecture supported three methods of data transfer, which was worth one point each if the architecture did support it. For the network layer, the architecture was evaluated on the data transfer protocols they utilize with a total of ten being considered with each being worth one point each if supported. The algorithms layer was focused on the openness of the architectures related to its code and data. A point was earned if the architecture had open code and used open data (as opposed to proprietary code and data) for a total of two potential points gained in the algorithms layer. The architecture layer was skipped since it was being evaluated. Finally, in the actuation & sensing layer, the architectures were evaluated to determine if there was a perception and/or an actuation & control module, which were one point each. Thus, the architectures could gain up to two points in this layer. The interoperability score was the sum of points across all the layers with a maximum of 29 points an architecture could receive for its interoperability score. Mission Planner tied with QGC with the highest score of 16, followed by another tie with VCSi and UxAS at 8. Figure 5 displays the results of this analysis with the specific criteria shown for each layer. Note that the architectures may be interoperable with more components; however, Figure 5 is as complete as possible based on publicly available data.

Table 2. Interchangeability Matrix

	Mission Planner	QGC	VSCi	UxAS
Mission Planner		0.71	0.25	0.44
QGC	0.71		0.35	0.35
VSCi	0.25	0.29		0.25
UxAS	0.44	0.35	0.25	

Now that each system has been evaluated, of interest is how similar they are to one another. An interchangeability analysis was conducted that compared the interoperability score between each architecture. The interchangeability metric for each pairing was determined by observing whether the architecture on the row paired was compatible with the architecture on the column given their layer components (if the selected architecture both had an “X” in the same column of Figure 5) This number was then divided by the larger of the two interoperability scores since, to have interchangeability, the two architectures should be interoperable with the same components in each layer. These results are presented in Table 2 For example, Mission Planner and VCSi have a total of 4 common components, this number was divided by Mission

Planner's interoperability score, 16, since VCSi only had a score of 8. Hence, the interchangeability score between Mission Planner and VCSi is 4/16 = 0.25.

6. Conclusions

With the rise in autonomous systems, their ability to communicate effectively with one another is of importance to their future development. Being interoperable allows one autonomous system to leverage the abilities of another, which is important to ensure the wheel is not constantly being reinvented. Thus, to facilitate this idea of an open modular autonomous system, we proposed a unified autonomy stack and interoperability taxonomy. The unified autonomy stack consists of eight layers (bottom-up): hardware, environment, data, data link, network, algorithms, architecture, and actuation & sensing. This discussion also necessitates the need for interoperability of autonomous systems. First, we define the related but different concepts of portable, interchangeable, compatible, and integrated systems regarding interoperability, which focuses on communication between two systems. Utilizing these definitions and current interoperability metrics, we propose an interoperability taxonomy to guide the future development of interoperable systems, especially autonomous ones. This taxonomy classifies systems as compatible, interoperable (at the syntactic, structural, or semantic levels), or integrated based on whether direct or indirect communication is utilized.

The autonomy stack was utilized to evaluate 4 architectures: Mission Planner, QGroundControl (QGC), VCSi, and OpenUxAS (UxAS)**.** Based on the architecture's ability to support a variety of different operating systems/containers, communication protocols, data formats, and data transfer standards were used in addition to whether they had proprietary or open code and data to determine their interoperability per the autonomy stack. Across the eight layers, 44 components were identified for the architectures to be evaluated. Through the analysis, the most to least interoperable architectures were: QGC (17), Mission Planner (16), UxAS (8), and then, VSCi (8). Furthermore, looking at the similarity (with 0 being not similar at all and 1 being the same) between architectures for interchangeability, Mission Planner and QGC received the highest score (0.71) with a significant decline with the remaining architecture pairs. Related future work concerning the interoperability taxonomy, might include a qualitative way of determining where a system lies would be of benefit to the evaluation of the interoperability of autonomous systems. Additionally, would be to better refine the criterion in the autonomy stack and expand our analysis to other autonomous architecture.

7. Acknowledgements

The views expressed in this paper are those of the authors and do not necessarily represent any views of the U.S. Government or Air Force. This work was cleared for unlimited release under SAF/PA-2023-0588.

8. References

AFRL. (2023, May). *OpenUxAS*. Retrieved from GitHub.

Al Ridhawi, I., et al. (2020). Generalizing AI: Challenges and opportunities for plug and play AI solutions. *IEEE Network, 35*(1), 372-379.

ArduPilot Dev Team. (2023). *Choosing a Ground Station*. Retrieved from ArduPilot Documentation.

Asif, S., & Webb, P. (2015). Software system integration - Middleware - an overview. *Int. J. of Computer Applications, 121*(5), 27-29.

Barad, K., et al. (2021). Towards incremental autonomy framework for on-orbit vision-based grasping. *Proc. of the IAC*.

Biggie, H., et al. (2023). Flexible supervised autonomy for exploration in subterranean environments. *arXiv preprint arXiv:2301.007*.

Bihl, T. J., & Talbert, M. (2020). Analytics for Autonomous C4ISR within e-Government: a Research Agenda. *Proc. 52nd HICSS*, 2218-2227.

Bihl, T., et al. (2018). Finding common ground by unifying autonomy indices to understand needed capabilities. *SPIE Proc*.

Buono, F. M. (2005, January 1). "Interoperability," not "interchangeability". *Corporate Counsel Business J*.

C4ISR Architecture Working Group. (1998). *Levels of information systems interoperability (LISI).* U.S. DoD.

Chen, D. (2006). Enterprise interoperability framework. *Proc. of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability.* Luxembourg: Enterprise Modelling and Ontologies for Interoperability.

Chen, D. (2017). Framework for enterprise interoperability. In B. Archimede, & B. Vallespir, *Enterprise Interoperability: INTEROP-PGSO Vision* (Vol. 1, pp. 1-18). Hoboken, NJ, USA: John Wiley & Sons, Inc.

Cuenca, L., Boza, A., Ortiz, A., & Trienekens, J. J. (2015). Conceptual interoperability barriers framework (CIBF): A case study of multi-organizational software development. *Proc. 7th Int'l Conf Enterprise Information Systems*, 521-531. Barcelona: SCITEPRESS.

Dantoni, J. (2022, May 18). *Open architecture defined: Advantages & when to consider*. Oracle Netsuite.

Dronecode Foundation. (2019). *QGroundControl*. Retrieved from QGroundControl.

Ford, T. C., Colombi, J. M., Graham, S. R., & Jacques, D. R. (2007). A survey on interoperability measurement. *12th Int. Command and Control Research and Tech. Sym.*

Ford, T. C., Colombi, J. M., Graham, S. R., & Jacques, D. R. (2007). The interoperability score. *5th Annual Conf. on Systems Eng. Research*, 1-10. Hoboken: SERC.

Gan, Y., et al. (2022). Braum: Analyzing and protecting autonomous machine software stack. *2022 IEEE 33rd ISSRE*, 85-96. Charlotte: IEEE.

Grovak, M. (2021, August 24). Embracing open architectures. *New Electronics*, 14-15.

Gulzar, M. (2022). Integrating a motion prediction baseline into an autonomy stack. *Final Report, Uni. of Tartu*.

Hajim, M. (2021). *Interoperability*. RingCentral.

Hasselbring, W. (2000). Information system integration. *Communications of the ACM, 43*(6), 33-38.

IEEE. (2018). *IEEE Std 1547-2018.* Piscataway: IEEE.

ISO. (2013). ISO 25964-2:2013. International Standard Organization.

ISO/IEC 25000. (2022). System and software quality requirements and evaluation (SQuaRE). *Portability*.

Kaisler, S. H. (2005). *Software Paradigms.* Hoboken: John Wiley & Sons, Inc.

Kaisler, S. H., et al. (2018). Smart objects: An active big data approach. *Proc. of the 51st HICSS* (pp. 809-818).

Kasunic, M. (2001). *Measuring systems interoperability version 1.0.* McLean: Software Engineering Institute.

Kingston, D. (2017, Dec. 7). *UxAS Overview*. Retrieved from https://www.youtube.com/watch?v=B6xIIcAoiwU

Kolb, S., & Wirtz, G. (2014). Towards application portability in platform as a service. *2014 IEEE 8th Int. Symposium on Service Oriented System Engineering*, 218-229.

Kovach, N. S., et al. (2021). The rise of open architectures in the U.S. Department of Defense. *Proc. SPIE 11753, Open Architectures/Open Business Model net-Centric Systems and Defense Transformation 2021*, 12-23. SPIE.

Kramer, R. (2016, February 24). Open architecture v. open systems: Distinctions that impact innovations from government contractors. *FEI Daily*.

Leal, G. S., et al. (2019). Interoperability assessment: a systematic literature review. *Computers in Industry, 106*, 111-132.

Lewis, G. A. (2013). *The Role of standards in cloud-computing interoperability.* Pittsburgh: Software Engineering Institute.

Li, S., et al. (2015). The internet of things: A survey. *Information Systems Frontiers, 17*, 243-259.

Lockheed Martin. (2023). *VCSi*. Retrieved from Lockheed Martin: https://www.lockheedmartin.com/en-us/products/cdl-systems/vcsi.html

Lum, H., & Heer, E. (1989). Toward Intelligence Robot Systems in Aerospace. In H. Lum, & E. Heer, *Machine Intelligence and Autonomy or Aerospace Systems* (pp. 1-13). Washington DC: AIAA.

Lyke, J. C. (2014). Empowering open systems through cross-platform interoperability. *Proc. SPIE 9096, Open Architecture/Open Business model net-Centric Systems and Defense Transformation 2014*, 1-16. SPIE.

Mensh, D. R. (1989). The quantification of interoperability. *Naval Engineers J., 101*(3), 251-259.

Meyer, D., & Zobrist, G. (1990, February). TCP/IP versus OSI. *IEEE Potentials, 9*(1), 16-19.

Mohamed, N., Mahadi, B., Miskon, S., Haghshenas, H., & Adnan, H. M. (2013). Information system integration: A review of literature and a case analysis. *WSEAS Math. and Computers in Contemporary Science*, 68-77.

MongoDB. (2023). *What is a tech stack and how do they work?* Retrieved from MongoDB.

Moon, T., et al. (2008). The what, why, when and how of interoperability. *Defense and Security Analysis, 24*(1), 5-17.

Morris, E., et al. (2004). *System of systems interoperability (SOSI): Final report.* Pittsburgh: Carnegie Mellon Software Engineering Institute.

NATO. (2021). *C3 Taxonomy Baseline 5.0.* Norfolk: NATO.

Oborne, M. (2023, May 12). *Mission Planner Home*. Retrieved from ArduPilot: https://ardupilot.org/planner/

O'Kelly, M., et al. (2016). APEX: Autonomous vehicle plan verification and execution. *SAE World Congress*, 1-12.

Rezaei, R., et al. (2014). Interoperability evaluation models: A systematic review. *Computers in Industry, 65*, 1-23.

Smith, G. (2018, February 15). *Interface, interoperability, integration - Quick Guide for Distributor*. Retrieved from TIMS Software: Industrial.

Taberko, V., et al. (2020). Principles for enhancing the development and use of standards within Industry 4.0. *Open Semantic Intelligence Systems Design Technologies*, 167-174.

Testing Standards Working Party. (2005). *Testing Standards*. Retrieved from Integration, interoperability, compatibility and portability.

Tolk, A., & Muguira, J. A. (2003). The levels of conceptual interoperability model. *Proc. 2003 Fall Simulation Interoperability Workshop*, 1-11.

Tolk, A., Diallo, S. Y., & Turnitsa, C. D. (2007). Applying the levels of conceptual interoperability model in support of integrability, interoperability, and composability for system-of-systems engineering. *J. of Systematics, Cybernetics, and Informatics, 5*(5), 65-74.

Trivellato, D., Spiesses, F., & Zannone, N. E. (2009). POLIPO: Policies & ontologies for interoperability, portability, and autonomy. *IEEE Int. Symposium Policies for Distributed Systems and Networks*, 110-113.

Tsaih, R.-H., et al. (2023, March). The AI Tech-Stack Model. *Communications of the ACM, 66*(3), 69-77.

US DoD. (2019). *Modular Open Systems Approach (MOSA)*. Retrieved from Defense Standardization Program.

Van Staden, S., & Mbale, J. (2012). The information systems interoperability maturity model (ISIMM): Toward standardizing technical interoperability and assessment within government. *Int. J. Information Eng. Eletronic Business, 5*, 36-41.

Wilder, R. (2019, January 22). *Integration vs. interoperability: What's the difference?* Retrieved from Spok.

Wood, R. J., et al. (2012). OSI for hardware/software interoperability. *Coupling Tech. to Nat. Need*, 231-239.

Zacharias, G. (2019). *Autonomous Horizons: The Way Forward.* Maxwell AFB, AL: Air University Press.

Zimmermann, H. (1980). OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Trans. on Communications, 28*(4), 425-432.