Machine Failing: System Acquisition, Software Development, and Military Accidents

Jeffrey Ding

June 2023

Abstract[1]

How does software contribute to military accidents? The stakes are high. In the present, software systems have been partly responsible for naval accidents, and such incidents could trigger larger conflicts. In the past, computerized early warning systems produced "near-miss" nuclear crises during the Cold War. In the future, military systems that incorporate new advances in artificial intelligence could fail with devastating consequences. Existing studies apply "normal accidents" and "high reliability organizations" theory to shed light on the causes of military accidents. While these approaches are helpful, they neglect the military's system acquisition process, which involves outsourcing software development to prime contractors such that input from military operators is limited to the late stages of development. By expanding the causal timeline of military accidents beyond decisions made on the battlefield to those made decades earlier in software design and development, this article presents an alternative way to explain how software contributes to military accidents. It tests the explanatory power of this theory against "normal accidents" and "high reliability organizations" theory across a range of case studies, including: i) the 1988 *Vincennes* incident, in which a U.S. naval ship accidentally shot down an Iran Air civilian airliner; ii) the Patriot fratricides at the beginning of the Second Gulf War; iii) the 2017 USS *McCain* collision; and iv) software upgrades in the 2021 Kabul airlift.

[1] This project has benefited greatly from conversations with Allan Dafoe, Sanne Verschuren, and Jacquelyn Schneider. For helpful feedback and suggestions, I thank Markus Anderljung, Noemi Dreksler, Ben Garfinkel, Julie George, Scott Sagan, Toby Shevlane, Eoghan Stafford, Baobao Zhang, and participants at an ISA panel. I am also grateful for support from Stanford's Center for International Security and Cooperation and Stanford's Institute for Human-Centered Artificial Intelligence.

I. Introduction

On August 21, 2017, the USS John McCain crashed into an oil tanker near the Strait of Malacca, resulting in the death of ten sailors and marking the Navy's worst accident in four decades. While the Navy initially blamed the incident on the McCain's crew, later investigations pointed out issues with the ship's navigation software.[2] In fact, the Navy's own review stated, "There is a tendency of designers to add automation…without considering the effect to operators who are trained and proficient in operating legacy equipment."[3]

Safety-critical software systems come with high stakes. In the present, navies rely on navigation software like the one used on the McCain, and an accident at sea is one of the most likely triggers for a conflict between the U.S. and China.[4] In the past, accidents involving computerized early warning systems produced many "near-miss" nuclear crises during the Cold War. In the future, military systems that incorporate new advances in artificial intelligence (AI) and other emerging technologies could also fail with devastating consequences.[5]

How does software contribute to the risk of military accidents? Existing scholarship on the safety risks of military technology systems draw on debates between normal accidents theory (NAT) and high reliability organizations (HRO) theory.[6] The normal accidents approach argues that the causes of accidents in highly complex and tightly coupled technological systems are deeply embedded in the systems themselves. Tightly coupled systems require centralized authority because small mishaps can rapidly escalate into major disasters, but problems in complex systems demand responses by local decision-makers who understand how the system works. The tension between

[2] Miller et al. 2019.
[3] U.S. Fleet Forces Command 2017.
[4] Yergin 2020.
[5] Scharre 2018; Horowitz et al. 2019.
[6] The seminal text on this subject is Sagan 1993. See Sagan 1991; Bezooijen and Kramer 2015; Scharre 2018. On the organizational politics of accidents in autonomous weapons, see Goldfarb and Lindsay 2022; Horowitz 2019; Horowitz et al. 2020; Scharre 2016; Schneider and Macdonald 2023. On other contextual factors that shape the risks of accidents, see Bode 2023; Owens 2003; Pauly and McDermott 2023; Slayton 2014.

these two imperatives results in unavoidable accidents.[7] Under NAT, since software-intensive military systems are very complex and tightly coupled, accidents are inevitable in these systems. For instance, one expert on AI governance predicts that normal accident problems will be "particularly acute in military AI applications."[8]

The HRO literature, in contrast, posits that certain organizations can effectively manage the risks of hazardous technologies. Studies in this tradition emphasize the importance of organizational culture, such as deference to expertise and dedication to learning from failures, as well as flexible organizational structures that permit authority to be centralized and decentralized depending on the situation.[9] As evidenced by studies of the U.S. Navy's nuclear aircraft carrier community and submarine community, certain military organizations have demonstrated excellent safety records with complex, interdependent technology systems.[10] Scholars have put forward the HRO model as a way to manage the risks of autonomous weapons.[11]

Without a doubt, these two approaches have produced valuable insights on the causes of military accidents. Yet, applications of NAT and HRO theory to software-intensive military systems share a drawback: their scope of analysis is confined to the actions of military organizations after software systems have been fielded. This means that they neglect the initial phase of software acquisition and development, when critical safety decisions are made. It also means that the existing literature does not account for the activities of private companies (defense contractors), which develop the vast majority of military software.

Taken together, these considerations point to the importance of the military's software acquisition process. If this process limits feedback from military operators to end-stage testing and

[7] Perrow 1984; Sagan 1993.
[8] Maas 2018; Borrie 2016; Scharre 2016; Horowitz 2019; Horowitz, Scharre, and Velez-Green 2019.
[9] Laporte and Consolini 1991; Hopkins 1999.
[10] Roberts et al. 1994; Scharre 2016.
[11] Dietterich 2019; Scharre 2016, 51.

evaluation (e.g., linear "waterfall" models), when it is too late to change fundamental system designs, accidents are more likely. These acquisition pathways often yield confusing human-machine interfaces and limit adaptability to hidden vulnerabilities that emerge from operators using the system in the field. This article proposes software development lifecycle (SDL) theory as an alternative way to explain how software contributes to military accidents.

Attention to the software development process helps account for system failures that HRO and NAT cannot explain. Consider, as an illustration, the failure of a Patriot missile battery to intercept a Scud missile which struck a U.S. barracks at Dhahran, Saudi Arabia, in the last days of the Gulf War. Army investigators attributed the breakdown to a timing error in the computer software designed by Raytheon. Long, continuous operation led to loss of precision in the range gate's tracking of missiles. Crucially, technical specialists were aware of the issue and had even developed a patch for the issue, but the upgrade was not prioritized because they discounted the possibility that operators would keep the computer running for long periods without a reboot.[12] The accident was not "normal"; despite tight coupling and complexity, the cause was well understood, and a fix was en route. Nor could it have been prevented by the Army improving its organizational culture and structure; the computer malfunction was a product of a considerable disconnect between Army users and software contractors in the Patriot's development process.

I test the relative strength of SDL theory against the NAT and HRO approaches with four case studies: i) the 1988 Vincennes incident, in which a U.S. naval ship accidentally shot down an Iran Air civilian airliner; ii) the Patriot fratricides at the beginning of the Second Gulf War; iii) the 2017 USS McCain collision; and iv) software performance in the 2021 Kabul airlift. Across each case, I evaluate the extent to which the empirical findings match with concrete expectations derived

[12] US General Accounting Office 1992. For more on Patriot malfunctions in the Gulf War, see Postol 1991 and correspondence by Raytheon officials.

from the three theoretical approaches, especially regarding the timing of the most salient decisions, the key precipitating events, and the principal actors. Selected and structured in a way that allows for a fair test of the three theories, the cases reveal how flaws in the military's system acquisition and development process were central to how software contributed to accident risks.

This article makes two main contributions. First, scholarship on military accidents has been preoccupied with the clash between the normal accidents and HRO frameworks. Since Scott Sagan's formative application of these frameworks to nuclear command and control accidents, this literature has seen very few theoretical innovations.[13] Departing from the NAT-HRO dichotomy, this article presents a novel approach to exploring the sources and limits of safety in military technology systems. In doing so, it connects political science scholarship to a shift in the systems engineering, human-computer interaction, and risk management fields, which increasingly emphasize the need for "moving beyond normal accidents and high reliability organizations."[14]

Second, this article has direct implications for the risks of emerging technologies such as artificial intelligence. In recent years, leading scholars and policymakers have likened the safety hazards of autonomous military systems to the Cold War's nuclear close calls, many of which were linked to false alarms produced by technological systems.[15] For example, Michael C. Horowitz, a University of Pennsylvania professor who serves as the director of the DOD's Office of Emerging Capabilities Policy, reexamined the Cuban Missile Crisis with autonomous naval ships in the mix, giving significant attention to the accident risks of uncontrollable systems.[16] These analyses appropriately highlight specific features of AI-enabled military applications, such as enhanced levels of autonomy, but they gloss over the simple fact that these applications will be implemented as

[13] Sagan 1993; Schlosser 2009.
[14] Leveson et al. 2009.
[15] In June 2023, comments made by a U.S. Air Force colonel — in which he detailed an exercise in which an AI-controlled drone attacked its operator — went viral. Later, the Air Force denied staging this simulation, and clarified that the colonel was describing a hypothetical scenario. Vanden Brook and Hjelmgaard 2023
[16] Horowitz 2019.

software programs.[17] While AI will bring novel risks, learning from past software-intensive military systems should serve as a foundation for comprehending the risks of military AI applications.

The article proceeds as follows. It first outlines my argument about how software development practices influence the risk of military accidents. Next, the article explains the empirical method by distilling the three theories' competing expectations about the impact of software technology on military safety. Evaluations of the Vincennes, Patriot, and McCain cases trace the sources of these accidents back to the initial software requirements phase and weak ties between software developers and military operators. The Kessel Run case further supports SDL theory by demonstrating that an iterative approach to software development can reduce safety risks. The article concludes with a summary of key findings, as well as their implications and limitations.

II. Theory: Software Development Lifecycle

How does software affect military accidents? Typically, a software failure is characterized as the inability of code to meet performance requirements.[18] When post-accident investigations assign blame to the crew by noting that software systems performed flawlessly, as was the case with the Navy's report on the McCain crash, they rely on this narrow definition of software failure. In contrast, NAT and HRO theory emphasize that software can contribute to accidents, even when it performs according to its requirements, by influencing proximate structures and organizations tasked with managing hazardous technologies. These prevailing organizational theories of safety in technological systems, therefore, present a broader conception of software failure in military accidents.

[17] One exception is Schneider and Macdonald 2023, which discusses the impact of different acquisition strategies on managing the risks of autonomous systems, including operational trade-offs between control/safety and cost.
[18] Foreman et al. 2015, 102.

The first approach contends that normal accidents occur in software-intensive military systems due to tightly coupled and highly complex structural elements. When problems — even seemingly trivial ones — arise in systems where many events occur simultaneously and interact with each other, it is difficult for managing organizations to identify fixes.[19] Even if the software works as coded, novel and unexpected interactions between battlefield conditions and such systems can snowball into unavoidable crises. For instance, in one 1979 false alert involving missile warning computers, the U.S. initiated retaliation measures based on mistaken reports of a major Soviet nuclear attack. This "near-miss" was produced by coincidences that would have been difficult to reasonably predict, including the insertion of training tape data at the same time as a momentary circuit failure in a ground station.[20]

The second approach, HRO theory, also posits that accidents in software-intensive military systems are rooted in the organizational structures that manage these systems. Expressing a more optimistic view, HRO scholars claim that organizations can reliably prevent system accidents if they maintain certain qualities, such as deference to experienced operators, devotion to learning from failures, and commitment to safety.[21] Studies of the U.S. nuclear navy, for instance, have highlighted the experience level of operators and cultural commitments to safety.[22] Notably, the autonomy of experienced front-line operators to "circumvent" certain bureaucratic procedures is one way HROs maintain system safety.[23]

While these two schools of thought are helpful for understanding how software affects military accidents, their focus is on how military organizations manage software systems after they

[19] Perrow 1984; Sagan 1993.
[20] Another coincidence that no one could have reasonably predicted was that the last block of sequential numbers on messages coming into the multiplexor of the system computer (the 427M) had been "001" and the first block of numbers in the mistakenly-inserted training data was "002." Aerospace Defense Command, History of ADCOM/ADC, 1 January-31 December 1979, n.d., Secret, excerpts, excised copy Jan 1, 1980. Newly declassified.
[21] LaPorte and Consolini 1991.
[22] Roberts et al. 2000; Winnefeld Jr., Kirchhoff, and Upton 2015; Scharre 2016.
[23] Leveson et al. 2009, 228.

are in operation. Less attention is paid to phases before militaries field these technologies: system acquisition and development. Yet, software safety specialists have identified the initial phase of software design and requirements specification as when the bulk of safety-critical decisions are made. Based on one study of military aviation mishaps, the concept development step accounts for 70 to 90 percent of safety-relevant decisions.[24]

Without accounting for military software acquisition and development, any explanation of accidents in software-intensive military systems is incomplete. In these early stages of system design, heavy reliance on contractors entails additional communication steps between people programming the software and those responsible for managing deployed systems. Drawing on Nancy Leveson's seminal work on accidents that involve software, one review of decades of research on this subject concludes, "The source of most serious problems with software relates to outsourcing software development."[25] In the context of military accidents, the strength of these feedback channels between operators and software developers is especially significant because they bridge the military and civilian domains.

This third approach, which I call software development lifecycle (SDL) theory, highlights the impact of software acquisition patterns on the development of accident-prone systems. Military software development typically follows a "waterfall model", which begins with system requirements specification and then progresses sequentially through system design, development, testing, and deployment. In this process, evaluation and feedback from operators is limited to the late stages of development, by which time it is difficult to rework system concepts.[26] As a 2010 National Research Council notes, the DOD's acquisition practices for information technology were hampered by a

[24] Leveson 2012, 51. Frola and Miller 1984.
[25] Dobbe 2022. Spanning engineering, risk assessment, and the social sciences, scholars from a wide range of disciplines have studied how to safeguard software-based systems over the past few decades.
[26] Hawley 2017.

"serial approach to development and testing (the waterfall model)," in which "end-user participation often is too little and too late."[27] Water does not travel back up a waterfall.

System safety can also be shaped by alternative models of software development, as well as other patterns of relationships among military procurers, field users, and external contractors. A more iterative software development process, for example, could limit the risk of accidents. For a start, incorporating more feedback from military operators in early stages of software system design could lead to human-centered interfaces, which aid military operators with rapidly interpreting data and making decisions in high-stress scenarios. Studies have found that confusing interface designs contributed to inadvertent military launches, fratricide, and other safety hazards.[28]

Another safety advantage of software-intensive military systems is their adaptability, in the sense that they can be patched to address problematic human-machine interactions and hidden vulnerabilities. This advantage, however, remains a theoretical one if technical specialists are not attentive to how operators are using the system, as demonstrated by the Dhahran example in the introduction.[29] Indeed, in recent years, the DOD has pushed for software engineering practices that involve rapid prototyping and early engagement with end operators.[30]

While SDL theory highlights the importance of software engineering practices, more communication between end-users and developers does not solve all safety risks with software-intensive military systems. Even if operators are sufficiently engaged in the software development

[27] National Research Council 2010. Related to the waterfall model, the DOD has historically relied on block development, in which each development phase is completed once according to unchanging software requirements established at the outset (cf. literature on evolutionary acquisition). Ford and Dillard 2009. I thank Jackie Schneider for her insights on this topic.
[28] Cummings 2006. For example, the lack of visual contrasts between button interfaces for different settings was implicated in a crash of an unmanned aerial vehicle during a routine flight in Nevada in 2000. Intending to press a key to pull up a submenu from the normal command menu, the pilot accidentally pressed a button in another menu button interface that severed the drone's data link connection with the ground control station. The two interfaces were visually identical apart from a different border color. Another term for this is a "mode error," which is when human operators lack awareness of the current mode of a particular system. Sheridan 2002.
[29] Schmitt 1991; Marshall 1992.
[30] Defense Innovation Board 2019.

process, they will not be able to anticipate all the contingencies under which the system will be used.[31] In the context of missile defense, Rebecca Slayton's work has demonstrated that differences between field tests and battlefield environments can demand significant shifts in software requirements.[32] Thus, if software updates stop with the testing stage, then iterative models of software development cannot handle such challenges.

The ramifications of SDL theory are present in other contexts where government agencies rely on external contractors to develop technological systems.[33] For example, the National Aeronautics and Space Administration (NASA) has struggled to ensure that contractors adequately attend to flight anomalies and safety issues. Like the DOD's acquisition process, NASA's contracting process has sometimes limited the ability of end-users, the Astronaut Office in NASA's case, to provide input on design problems.[34] As a result, contractors aggressively pursued short-term solutions that avoided significant delays in delivery, resulting in unaddressed safety issues.

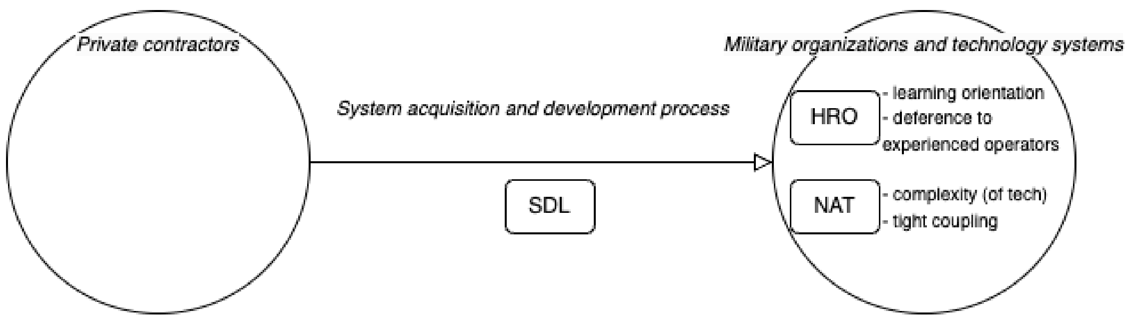
While this study does not investigate the deeper causes of the software development models employed by militaries, the organizational politics of software acquisition provides a useful starting point. To maintain their key programs, defense-specialized contractors, especially large prime contractors, leverage their political relationships with government officials as well as their fluency in the military's operational needs and idiosyncratic standards.[35] There is limited incentive for these defense-oriented firms to move away from the waterfall model, as evidenced by their resistance to past acquisition reforms.[36] In the U.S. case, personnel practices also reinforce certain software development patterns. Senior acquisition officials, often political appointees with limited operational

[31] Borning 1987.
[32] Slayton 2014.
[33] These theoretical claims also apply to other militaries. On different models of defense software development in France, the UK, Germany, and China, see Soare et al. 2023.
[34] Vaughan 1990, 241.
[35] Dombrowski and Gholz, 2006; Alic, 2014.
[36] Gholz and Sapolsky 1999, p. 34-35.

experience, are not particularly motivated to usher in more iterative software development approaches.[37]

SDL theory builds on the NAT and HRO approaches, which have broadened our perspective on software's role in military accidents, but also differs from them in crucial ways. Debates between the normal accidents and HRO camps tend to concentrate on whether accidents are inevitable or not in complex technological systems.[38] Likewise, this clash has been adjudicated in military domains by focusing on how military organizations operated and managed such systems after they had been fielded. SDL theory extends the causal timeline for military accidents back to the initial phases of software design and requirements specifications. In doing so, it also broadens the range of actors responsible for military accidents to include the defense contractors that actually build the software (Figure 1).

Figure 1: Expanded Causal Timelines and Actors in Military Accidents


Private contractors
System acquisition and development process
SDL
Military organizations and technology systems
HRO
- learning orientation
- deference to experienced operators
NAT
- complexity (of tech)
- tight coupling

[37] Fox 2012. Relatedly, boards of inquiry for military accidents are typically staffed by military operators, as opposed to professional investigators. These boards tend to avoid investigating the actions of contractors in designing systems, because they want to avoid implicating the more senior officials in charge of procurement decisions. Interview with C.W. Johnson, author of handbook on military accidents, March 13, 2023.

[38] In Sagan's words (1993, 14), these discussions center around "conflicting visions about what could be called the degree of perfectibility that is possible in complex organizations."

III. Empirics

Adopting a congruence analysis approach, I assess the relative strength of SDL theory against NAT and HRO theory across four historical case studies: the *Vincennes* accident (1969-1988); the Patriot fratricides (1985-2003); the USS *McCain* collision (2008-2017); and Kessel Run software's performance in the Afghanistan evacuation (2017-2021). In each of the cases, I evaluate the observable implications of SDL theory, and then compare them to those derived from the two more established theories (Table 1).[39] Though all three theoretical approaches agree that there is a causal relationship between the introduction of software and an accident, they offer substantially divergent interpretations of *how* this process occurs, in particular as it relates to principal actors, timing of most relevant decisions, precipitating events, and conceptions of software failure. The last case, in which the U.S. Air Force experimented with a different approach to software acquisition and development, further probes SDL theory by providing variation on the type of software acquisition pathway.

Table 1: Competing Perspectives on Software's Contribution to Military Accidents

	SDL	HRO	NAT
Principal actors	Software developers (contractors) and senior procurement officials	Military organizations responsible for operating software systems	Military organizations responsible for operating software systems
Timing of most relevant decisions	Initial contracting phase, software requirements specification, preceding decade(s)	After the system has been fielded (emphasis on continuous operations and training)	Combination of system design and organizational operations after system has been fielded
Conception of software failure	Software development process does not incorporate feedback from end-users	Operators not adequately trained to manage software	Software contributes to tight coupling and high complexity
Precipitating events	Predictable issues that remain unaddressed due to software dev. lifecycle	Mistakes that escalate because operators unable to maintain system safety	Novel and unexpected interactions between system components

[39] Blatter and Blume 2008.

To ensure a harder test of SDL theory, I selected cases that are held up as representative illustrations of the NAT and HRO models. Analyses of the McCain, Vincennes, and Patriot accidents often pinpoint either complex software or inadequate operator training as key causal factors.[40] Paul Scharre, for example, writes, "The causes behind the Patriot fratricides illustrate how normal accidents also can occur in military systems."[41] This test's generalizability is enhanced because these cases also differ in many ways, including their operational context, level of complexity, organizational culture, type of safety hazard, as well as the relevant military organizations.[42] Ideally, the cases would not be confined to the U.S. context, but this decision was shaped by practical considerations, including access to interviewees and archival records on a subject which the U.S. is relatively more open about than other countries.

The focus on software applications in navigation, operational planning, and weapons control systems — which aid platforms in tracking, targeting, and shooting their targets — provides two further advantages. First, these are reference classes that are similar in many relevant aspects to how AI could be incorporated into powerful military systems.[43] Second, compared to those linked to nuclear weapons systems, command and control issues in these military software applications are relatively understudied, and they provide a new universe of cases to explore the effects of technology on military accidents.[44] Thus, they provide fertile ground for theory development, so as to evaluate whether the potential validity of my argument is sufficient to warrant further empirical testing.

[40] Bode 2023; Scharre 2016; Miller et al. 2019.
[41] Scharre 2016.
[42] The three main branches of the U.S. military are represented: i) the Afghanistan evacuation (Air Force), ii) the Vincennes and the McCain (Navy), and iii) the Patriot fratricides (Army).
[43] Perrow, the pioneer of NAT, also cites software as a neglected area. Perrow 1999, 354.
[44] Rochlin 1991.

Lastly, these cases all feature systems in military settings where safety must be balanced against operational effectiveness. Lessons learned from software-related accidents in civilian domains where safety considerations supersede all else, such as airline transport or air traffic control, may not translate. Similarly, in many of the military operations studied by HRO researchers, such as peacetime aircraft operations, safety goals were protected from other competing priorities.[45] Insights from these cases, therefore, provide more leverage for understanding the risks of human-software interaction in the settings where the reduction of accident risk is especially challenging and geopolitically sensitive.

All of these accidents are the product of multiple, overlapping factors. The task of analyzing the congruence between observations and the predictions of three different theories is made more difficult because many of the official post-accident investigations do not discuss the system development process and the role of defense contractors. To fill in the gaps, I benefit from declassified documents, archived discussions in the Forum on Risks to the Public in Computers and Related Systems, and interviews with people who developed and tested these systems.

Vincennes (1969-1988)

On July 3, 1988, the USS Vincennes deployed to investigate reports of Iranian Revolutionary Guard Corps speedboats attacking neutral merchant ships.[46] Known as a "robo-cruiser," the Vincennes boasted an Aegis system, a highly sophisticated combat information center (CIC) that automated functions such as target classification and target-weapon pairing.[47] That morning, radar operators on the Vincennes misidentified an Iranian commercial plane for a military one, and the

[45] Leveson et al. 2009, 239.
[46] Much of this section draws on archived discussions in the Forum on Risks to the Public in Computers and Related Systems, a community of computer safety researchers who analyzed incidents such as the Vincennes. I also relied on materials about the Vincennes investigations held in the Black Vault, an online repository of declassified documents.
[47] Cox 2018.

U.S. naval ship fired surface-to-air missiles at Iran Air flight 655, killing all 290 civilians on board. Over three decades later, the tragic event still generates mistrust of the U.S. among the Iranian leadership and public.[48]

How does a billion-dollar warship, equipped with state-of-the-art software for tracking and classifying aircraft, end up shooting down a commercial plane? The official investigation revealed that the ship's Aegis system supplied accurate data to the Vincennes crew. Concretely, the system provided altitude information that the plane was ascending (like a commercial plane), not descending (like a hostile military plane). The crew, however, reported that the aircraft was descending as it approached the ship.[49] There is much to be mined from the high-stakes calls made in these critical minutes, and the ensuing investigations and Congressional hearings identified stress and psychological factors as primary causes.[50] Yet, it is equally, if not more, valuable to trace problems with the Aegis further back to the decisions made about software design in the initial procurement phase. This would also cast attention on the companies that built the system — most prominently RCA Corp., which was awarded the prime contract for the Aegis in 1969.[51]

If SDL theory holds, the case evidence should show that the system development process factored into the CIC operators' struggles with managing Aegis-related risks. One of the key issues is whether the Aegis's program managers and RCA gave sufficient attention to user interface considerations in tense, combat settings. Matt Jaffe, a systems engineer at RCA in the mid-1970s, pushed for the Aegis display to include a rate-of-change indicator for altitude. Having previously served in the Vietnam War on ships equipped with forerunners to the Aegis system, Jaffe was one of the few people involved in Aegis software development with experience operating similar systems in

[48] Danby 2021.
[49] Smith 1988.
[50] Dotterway 1992.
[51] Hilts 1988.

high-stress environments. On the issue of adding a rate-of-change indicator for altitude, Jaffe recalls, "I wound up literally screaming at my boss, 'You hired me for my technical experience and combat experience. If you're not going to listen to me, why don't you fire me?!'"[52]

These human-machine interface issues contributed to the *Vincennes* accident. Without this rate-of-change indicator, controllers had to "compare data taken at different times and make the calculation in their heads, on scratch pads, or on a calculator – and all this during combat."[53] This increased the likelihood of misreading whether a ship was descending or ascending. These issues could be traced back to the extent, timing, and influence of feedback from military operators to software contractors in the system development process. As Jaffe states, "When Aegis was being developed, it was being developed with a waterfall (model)…we drew up a software requirements specification document, sent it to the Navy for review, and then they would come back for one meeting of a few hours. It's hard to do human-machine interface that way."[54]

Furthermore, many actors had raised concerns that Aegis systems proceeded into production without adequate attention to testing and operator feedback. A Government Accountability Office (GAO) investigation reported a "long list of testing limitations" to the Aegis system at RCA's facility in Moorestown, New Jersey, such as the on-land location, lack of actual missile firing capability, and differences between the system being tested and the production version.[55] In 1983, according to the GAO report, a Navy Admiral and former director of the DOD's testing and evaluation office "termed the Moorestown facility not sufficiently realistic for tests prior to a production milestone."[56]

[52] Interview with Matt Jaffe, March 8, 2023. Jaffe also emphasized that RCA managers provided some valid pushback to this indicator, including: the limited display space, risks of information overload, and issues related to the vertical beam width of the radar (which could muddle rate-of-change calculations). Ultimately, Jaffe's supervisor claimed that the Navy never requested such an indicator. Thus, the root issue remains that RCA was not receiving input from military operators experienced with such systems.
[53] Lerner 1989.
[54] Interview with Matt Jaffe, March 8, 2023.
[55] I am very grateful to Shelby Oakley, at the Government Accountability Office, for helping me locate this report. U.S. Government Accountability Office 1988.
[56] U.S. Government Accountability Office 1988.

In 1983 and 1984, the Navy did conduct a few sea tests of Aegis combat systems equipped with a SPY 1-A, which was the radar used on the *Vincennes*; however, the realism of these tests has also been questioned. The test ranges were set up such that threats would only come from a predictable area. Moreover, certain events, such as aircraft leaving the immediate test area, tipped off crews that a test event would soon occur. Together, these conditions allowed "crews to deduce the general direction, timing, and type of the test threats."[57] Sources familiar with a classified version of the GAO report confirmed that these sea tests did not approximate a realistic, challenging combat environment.[58]

While this GAO investigation was published three weeks after the *Vincennes* accident, it drew on interviews conducted between September 1987 and March 1988. The report also studied testing and evaluation processes for not just the Aegis but also five other systems.[59] This means that, in contrast to many post-incident reviews, the GAO study had identified problems with the Aegis *before* the accident, limiting the risk of hindsight bias in shaping its conclusions.

As noted above, the earliest investigations of the *Vincennes* accident converged on psychological biases that affected the crew during those fateful minutes when the decision was made to strike, in particular a condition labeled "scenario fulfillment", under which groups fixate on a possible scenario (e.g., an Iranian air attack) and ignore evidence that goes against their expectations.[60] Later examinations of this case have undermined this account, arguing that the crew's misinterpretation of data was likely due to a combination of information overload and unwieldy user displays, rather than collective delusion.[61] Moreover, some psychologists stressed that mistakes made

[57] U.S. Government Accountability Office 1988.
[58] Ahern 1988. Regarding these sea trials, a detailed *Newsweek* investigation stated that "the navy could not afford to risk failure in the trials for fear that Congress would stop funding the Aegis program." Barry and Charles 1992.
[59] Chelimsky 1988.
[60] Another alternative factor is changes in the rules of engagement. See Sagan 1991. Sagan identifies "hair trigger" rules of engagement as a "permissive cause" in the Vincennes tragedy.
[61] For one of the strongest arguments against the "scenario fulfillment" explanation, see Dotterway 1992. Roberts and Dotterway 1995.

by operators were entrenched in the system design and development process. On this thread, Richard W. Pew, who testified to Congress on behalf of the American Psychological Association, stated, "Part of the problem is that automation decisions are made at the time the fundamental architecture of a system is being defined. We need more extensive methods of analysis to understand how to integrate human operator performance with system performance *during the conceptual design state of new weapon systems*."[62]

What can NAT and HRO tell us about the Vincennes case? At the time, the Aegis was one of the most complicated weapons systems in action.[63] It could also operate at high levels of automation from target tracking to missile firing sequences. This type of tight coupling was needed to respond to the types of threats that Aegis would face – the time between the appearance of the Iranian plane on the radar screen and the decision to fire spanned four minutes.[64] With a system like this, NAT tells us that a Vincennes-like disaster was inevitable.

If the Vincennes accident was caused by novel and unexpected interactions related to the Aegis system, that would provide further support to the NAT account. Based on evidence gathered by Kristen Dotterway, it is possible that the Aegis automatically re-assigned the Iranian plane's track number (TN 4474) to a different track number (TN 4131) entered by the USS *Sides*, which was part of the surface action group under the Vincennes. Then, when the Vincennes crew asked for an update on TN 4474, thinking it was still attached to the Iranian plane, the Aegis computer had already matched that number to a U.S. Navy jet that was descending to check in with a U.S. aircraft carrier in the Gulf of Oman. By some accounts, this was a "freak occurrence" that explains why multiple crew members reported that the aircraft was descending.[65]

[62] Pew 1988. Emphasis mine.
[63] Hilts 1988.
[64] Bode and Watts 2021, 44.
[65] Maclean 2017, 29.

On the other hand, aspects of this case suggest that the breakdowns in information-gathering within the Vincennes were much more predictable. Assuming the TN 4474 re-assignment account holds up, the software development process should have addressed the risks associated with automatically changing track numbers.[66] At the very least, the Aegis should have had an alert mechanism that notified the operator when track numbers had been automatically re-assigned.[67] Dotterway, who originally reconstructed the sequence of events in this account, also blames the "poor interface between the Aegis weapon system and the operator, especially the procedural complexity and a problematic presentation of information illustrated in the auto-correlation and subsequent confusion of track numbers."[68] Furthermore, another explanation for the "descending" call was that the crew had misinterpreted decreasing range values as altitude values, which would connect back to the system development issues related to rate-of-change indicator for altitude [69]

By highlighting the operators of the *Vincennes*, the HRO approach also offers some insights into this case. According to HRO theory, preventing Aegis-linked accidents comes down to whether organizations can cultivate environments in which groups can reliably perform in high-pressure situations, such as deference to experienced frontline operators who can independently circumvent rules to prevent accidents when issues arise.[70] In the *Vincennes* case, the CIC operators had limited experience managing Aegis systems in high-stress environments. Per a *Newsweek* investigation, the tactical officer for surface warfare "was uncomfortable with computers" and, according to one fellow officer, "used his screen as a surface for 'self-stick' notes."[71]

[66] In fact, these types of risks were well-established, not freak occurrences. Interview with Dr. Nancy Roberts, June 12, 2023.
[67] Dotterway 1992, 59.
[68] Dotterway 1992, 173.
[69] Dotterway 1992, 54.
[70] Scharre 2018.
[71] Barry and Charles 1992.

Yet, it is important to not overstate the implications of HRO in this case. Even the most well-trained and experienced crew would have struggled to process Aegis data in combat conditions.[72] As the evidence above demonstrates, the *Vincennes's* problems were less about the level of training or the ability of operators to learn from mistakes and more entrenched in the flaws of the software development model. By the time operators could train with and test the system, any issues they identified would not have been able to be fixed. Thus, the HRO approach disregards the role of the contractors that develop Aegis software and the robustness of feedback loops between military operators and system designers.

The Department of Defense's official investigation report on the *Vincennes* incident numbers 153 pages.[73] There are over 70 references to Captain Will Rogers III, who commanded the *Vincennes*; nearly 20 mentions of the experience level of operators; and 9 references to the compression of time or complexity of the technological system. RCA, the system developer, is not mentioned even once. Fully understanding the causes of the *Vincennes* tragedy requires looking beyond just the organization operating the ship to Moorestown, New Jersey, where the Missile and Surface Radar Division of RCA was located.

Patriot "Friendly-Fire" Incidents (1985-2003)

In 2003, during the Second Gulf War, the U.S. Army's Patriot air defense system was involved in three "friendly fire" incidents. In one episode, which occurred on March 23, a Patriot missile downed a British Tornado fighter-bomber, killing the two British soldiers on board. A week and a half later, a Patriot battery shot down a U.S. Navy fighter jet, killing the pilot in another fratricide. In a third incident, a Patriot radar locked on to a U.S. Air Force F-16; in response, the pilot destroyed the Patriot battery with a missile. Fortunately, this confrontation resulted in no

[72] Barry and Charles 1992.
[73] Fogarty 1988.

casualties. In total, friendly fire incidents comprised 25 percent of the Patriot's twelve total engagements in the conflict.[74]

Problems with how the Patriot system interpreted identification friend or foe (IFF) signals were central to these friendly fire incidents. In 2005, a Defense Science Board task force reviewed the Patriot's performance in Operation Iraqi Freedom (OIF) and concluded that Raytheon's IFF system performed very poorly – a problem that had been made apparent in training exercises.[75] This begs the question of why this issue was not addressed. The task force stated that it "remains puzzled as to why this deficiency never garner[ed] enough resolve and support to result in a robust fix."[76]

Detailed investigations suggest that the answer to this puzzle lies in the system development process, which extends back to at least 1985, the year that Raytheon was awarded a Patriot software modernization contract.[77] According to Dr. John K. Hawley, an engineering psychologist with the U.S. Army Research Laboratory with over 35 years of expertise on human-machine interactions in Patriot units, the DOD's systems acquisition process functioned as the most daunting obstacle to safer Patriot performance in OIF. Raytheon, the Patriot's prime contractor, followed the waterfall model, in which feedback and evaluation was left until system development was nearly complete.[78] Hawley comments, "[The system developers] still have this idea that they have this acquisition process, and when it's finished, it's finished. Oftentimes, the user is not equipped to use it that way."[79] The Government Accountability Office diagnosed similar issues in earlier cycles of Patriot development, in which "the Army believed it necessary to proceed (with production) even though test results identified major problems."[80]

[74] Scharre 2018.
[75] Defense Science Board Task Force 2005.
[76] Defense Science Board Task Force 2005.
[77] This contract initiated the first phase of software modifications (PAC-1). The PAC-2 software changes, which were the most relevant to OIF, began in 1986.
[78] Hawley 2017.
[79] Interview with John K. Hawley, May 9, 2022.
[80] Conahan 1983, 9.

This approach was especially ill-suited to Patriot upgrades in identification algorithms and automatic operating modes, which were brittle and demanded military users to intervene in extreme situations.[81] As they made upgrades to the Patriot's software, contractors and concept designers built technical components to meet certain efficiency and performance requirements, leaving operators to deal with the residual impacts (e.g., the Patriot's issues with track classification and identification). Whether operators could fulfill the associated demands was not tested until the tail of the system development process, when it was too late to change the system's fundamental design. Hawley and Anna L. Mares, another researcher at the U.S. Army Research Laboratory, conclude, "The roots of [the Patriot's] apparent human performance shortcomings can be traced back to systemic problems resulting from decisions made years earlier by concept developers, software engineers, procedures developers, testers, trainers, and unit commanders."[82]

Indeed, even efforts oriented toward identifying deficiencies in Patriot operators eventually turned to the system acquisition process. Following the 2003 fratricides, the Army Research Laboratory initiated "The Patriot Vigilance Project," which, as its name suggests, initially aimed to investigate the discipline and alertness of Patriot operators. Ultimately, after expanding its scope to cover twenty years of the Patriot's evolution, the review warned against laying too much blame on operators, instead emphasizing deeper system development problems like "faulty going-in concepts" that proved "difficult and expensive to correct" later in the process.[83]

How well do the NAT and HRO approaches account for the Patriot accidents? Certainly, the two system features of normal accidents were present in the Patriot case. To begin, it was difficult for operators to grasp the intricate connections between the Patriot system's moving parts. In the aftermath of the friendly-fire incidents, experts scrutinized the Patriot's "enormous

[81] Hawley 2017.
[82] Hawley and Mares 2018.
[83] Hawley 2007.

complexity," which had been enhanced by software upgrades to enable automated engagement.[84] Second, the Patriot system was also tightly coupled. There was very little slack between the initial detection of an incoming missile and the decision to respond.[85] As the British Ministry of Defence's inquiry into the Patriot-Tornado fratricide put it, "The crew had about one minute to decide whether to engage."[86]

While some aspects of this case bear out the expectations of NAT, it misses other key contributing factors to the Patriot accidents. The observable implications of NAT are most compatible with the second fratricide involving the F/A-18C Hornet fighter jet, in which the Patriot system could not adapt to novel and unexpected interactions.[87] In this instance, when Patriot radars operated in close proximity and followed the same aircraft, they would produce false ballistic missile trajectories, or "ghost tracks."[88] In line with NAT, this situation was relatively unpredictable, and it would have been difficult to uncover in the development process.

However, other elements of the Patriot accidents were much more preventable. The "ghost tracks" issue was not relevant to the March 23 fratricide or the U.S. F-16 engagement on the Patriot. Most of the IFF problems were well-known to both Patriot operators and aircraft pilots, who were afraid to fly in zones tracked by Patriot units due to the frequency with which the radar systems would lock on to their aircraft.[89] As Paul Scharre writes, "Other problems, such as the potential for the Patriot to misclassify an aircraft as an anti-radiation missile, had been identified during operational testing but had not been corrected and were not included in operator training."[90]

[84] Piller 2003.
[85] If the Patriot was operating in auto-fire mode, even less slack was present.
[86] Ministry of Defence (United Kingdom) 2003, 3; cited in Bode and Watts 2021.
[87] Scharre 2018, 144.
[88] Postol 2004. Lambeth 2013, 245.
[89] After the F-16 aircraft shot down the Patriot unit, one pilot remarked. "We had no idea where the Patriots were, and those guys were locking us up on a regular basis. No one was hurt when the Patriot was hit, thank God, but from our perspective they're now down one radar. That's one radar they can't target us with any more." Lambeth 2013, 115.
[90] Scharre 2018, 144. Unlike the ballistic missile misclassification, there was plenty of evidence to suggest that aircraft might be misidentified as anti-radiation missiles.

Developments in this case also illustrate the HRO approach's utility and drawbacks. The Patriot crew were relatively inexperienced and overly reliant on automated outputs.[91] In its investigation of the Patriot fratricides, the Defense Science Board task force suggested that operators gain more autonomy over firing decisions. "The solution here will be more operator involvement and control in the functioning of a Patriot battery," the task force report states.[92] Failing to satisfy critical features of an HRO, Patriot units did have experienced operators who could bypass established practices to maintain safety in crisis.[93]

Still, regarding the Patriot accidents, HRO theory's explanatory power is limited in two ways. First, it is unlikely that even front-line operators with more experience and autonomy would have been able to substantially reduce accident risks. The Patriot's IFF problems meant operators would have very little time and information to circumvent the system's targeting decisions.[94]

Second, the HRO approach neglects the need for operator feedback earlier on in the system development process. For instance, a more iterative development process could have alerted designers and contractors to the need for targeting algorithms that could be updated according to data specific to the missiles in a particular theater.[95] In contrast, the U.S. Army and Raytheon had "committed to a system concept that demonstrate[d] patterns of performance unreliability," leaving it to the operators to deal with the additional risks.[96] In this sense, the focus on Patriot safety culture and operator training points toward an "end-of-pipe" solution, whereas the SDL approach controls risk from the source.

---

[91] Automation bias refers to the phenomenon when operators have "too much" trust in autonomous systems. Bode and Watts 2021. Horowitz 2019.

[92] Defense Science Board Task Force 2005.

[93] An organization's learning orientation is another key characteristic of HROs. In this case, there is evidence that the Army did not learn from Patriot deficiencies in the First Gulf War. Postol 1991.

[94] Ministry of Defence (United Kingdom) 2003.

[95] Bode and Watts 2021, 55.

[96] Hawley 2007.

The above evidence suggests that the Patriot's safety issues are deeply embedded in the military's acquisition approach for software-intensive systems. In response to the Patriot fratricides, the military implemented technical fixes and adjusted human-machine interfaces to help operators adapt to new systems. Yet, these reforms do not address the underlying problems with how the military, in partnership with Raytheon and other private contractors, acquires and develops systems like the Patriot, in particular its reliance on the waterfall model of software development.

USS McCain Collision (2008-2017)

On August 21, 2017, the U.S. Navy destroyer *John S McCain* made a sudden turn to port and collided with a Liberian-registered oil tanker east of the Straits of Malacca. About an hour before, the destroyer's captain had switched the Integrated Bridge and Navigation System (IBNS) to backup mode, setting off a series of mistakes that caused the hard turn. The crash resulted in the death of ten Navy sailors, making it the Navy's deadliest accident in four decades.[97] More broadly, this mishap renewed fears that maritime accidents, especially ones that take place in strategically important waterways like the Straits of Malacca, could escalate into major military confrontations.[98]

As established in reviews by the National Transportation Safety Board (NTSB), the United States Fleet Forces Command, and the nonprofit newsroom ProPublica, design flaws in the IBNS played a major role in the accident. First, in its backup mode, the IBNS allowed crew from different parts of the ship to take charge of steering, which led to confusion over which station had thrust control for different propellers. Second, for steering commands, the IBNS only provided touch-screen controls, as opposed to mechanical throttles that provide more tactile feedback to operators.[99] As the NTSB's investigation concludes, "The design of the John S McCain's touch-

[97] Miller 2019.
[98] See, for example, The Straits Times 2017.
[99] These design decisions did not adhere to international standards for human engineering in marine systems, which have been approved by the U.S. DOD. Mallam 2020, 57.

screen steering and thrust control system increased the likelihood of the operator errors that led to the collision."[100]

To be sure, issues with the IBNS development process could have been partially mitigated by improved decision-making on the day of the collision or better training in the months before. Ultimately, however, these design flaws were rooted in the proclivity of system designers to automate and digitalize navigation functions without sufficient attention to the needs of operators. According to IBNS operators interviewed in the Navy's investigation, they regularly disabled the system's touch screen to avoid accidental rudder changes and ignored fault notifications due to the difficulty of interpreting indicators on the display areas.[101] Notably, the IBNS modernization efforts did not consult the Navy's own experts on human factors engineering.[102]

It was only after the McCain collision that the Navy recognized the need for stronger feedback loops between operators and software developers. Prompted by an internal review, the Navy surveyed surface ship crews on the IBNS system. One striking finding — "the number-one feedback from the fleet" according to Rear Adm. Bill Galinis — was that operators "overwhelmingly" preferred mechanical controls over touch-screen systems.[103] This type of participatory input, in which the introduction of new technology integrates end-user perspectives, was missing from the initial development of the IBNS.[104] In line with SDL theory, these problems extend back across almost a decade of software acquisition and development to 2008, when the Navy first announced a contract with Northrop Grumman to build the IBNS.[105]

[100] National Transportation Safety Board 2019, 33
[101] U.S. Fleet Forces Command 2017.
[102] U.S. Fleet Forces Command 2017.
[103] Eckstein 2019.
[104] Mallam et al. 2020. As Eric Lofgren, an expert on defense acquisition, states, "Such an advanced bridge/navigation system should probably first been tried on smaller ships with continuous user feedback, tested extensively, iterated, then progressively scaled up to larger and more complex ships." Lofgren 2019.
[105] Miller et al. 2019.

Insights from NAT and HRO theory also bear on this case. Regarding the former, some of the IBNS's issues certainly stemmed from adding unnecessary complexity, including the feature to transfer thrust control for each propeller.[106] Moreover, the accident happened in one of the world's busiest seaways, an environment ripe for unexpected interactions in which one misstep could easily cascade.[107] Applied to this case, NAT posits that accidents like the McCain collision are inevitable as long as the Navy relies on navigation systems that are highly complex and tightly coupled.

It is important, however, to not overemphasize NAT's explanatory power with respect to the McCain collision. To begin, not all the design issues with the IBNS can be reduced to unexpected interactions and the limited amount of slack between various components. For instance, automated bridge systems equipped with improved indicators and alarms should encourage looser coupling by giving the crew more time to correct issues that arise. Yet, the McCain crew ignored these alert systems because the display area was densely packed and difficult to interpret — the product of not incorporating operator needs into the design process.[108] Furthermore, many of the IBNS's complications were neither novel nor unexpected, as presumed by NAT, but rather predictable. The replacement of mechanical throttle with touch-screen controls, for example, went against principles held by the Navy's own human factors engineering team and DOD standards.[109]

Some of the lessons from HROs also pertain to the McCain case. Forward-deployed forces in the Western Pacific faced high operational tempo and staffing shortages, which resulted in long shifts and inadequate rest.[110] This undermined the crew's ability to take safety measures, as evidenced by the fact that bridge watchstanders were "acutely fatigued at the time of the accident."[111] Another critical aspect of HROs is the high experience level of operators. In this case, post-accident

[106] Mallam et al. 2020, 57.
[107] Miller et al. 2019; Mallam 2020.
[108] U.S. Fleet Forces Command 2017.
[109] Mallam 2020, 57; U.S. Fleet Forces Command 2017.
[110] Aucoin 2018; Rowen et al. 2022.
[111] NTSB 2019.

investigations highlighted that the crew had not received sufficient training on the new IBNS system.[112]

Yet, absent changes in the software development process, it is unlikely that the McCain could have avoided a major accident even if it had adhered to HRO principles. Having trained with the IBNS system in sea trials, the McCain's captain grew accustomed to operating the IBNS in backup mode, which removes built-in safeguards, in part because he could not understand some of the system's automated functions.[113] Hypothetically, if the entire crew had more time to train with the navigation system, they would have adopted similar ways to bypass unwieldy features of the IBNS. Contrary to the expectations of HRO theory, which regard the ability of operators to sidestep certain procedures as a benefit to system safety, in this case, such circumventions would have enhanced the risk of accidents.

Following the accident, the Navy sanctioned crew members for failing to properly manage the IBNS in the critical minutes before the collision, while the organizations and officials responsible for the software development evaded blame. In August 2019, the Navy did announce that it would contract with Northrop Grumman to remove touchscreen controls across the fleet; however, there has been no evidence that this redesign process will incorporate input from surface warfare operators.[114] As the SDL approach suggests, without this fundamental change in the connections between the organizations that procure and develop the software, accidents like the McCain collision will continue to occur.

[112] U.S. Fleet Forces Command 2017; NTSB 2019.
[113] Advice from two other captains helped him make this decision. Miller et al. 2019.
[114] Malachowski 2020.

Kessel Run Software and the Afghanistan Evacuation (2017-2021)

In the last weeks of August 2021, the U.S. and its allies scrambled to evacuate as many people from Afghanistan as possible before the August 31 withdrawal deadline. The Kabul International Airport was not used to managing this level of traffic — in all, more than 120,000 people were flown out — so mission leaders relied on a software tool to plan out the exact time slots for arrivals and departures. Developed by Kessel Run, the U.S. Air Force's own software factory, this planning software played a major role in the largest non-combatant evacuation in U.S. military history. Lt. Gen. Greg Guillot, who led the mission as the Air Forces Central commander, stated that Kessel Run's software "served as a reliable, adaptable tool as we planned and executed this complex, historic operation."[115]

What accounts for the safe and reliable performance of Kessel Run's software in the Afghanistan evacuation? This outcome diverges from expectations linked to the HRO and NAT framework. The operation, a 17-day sprint that required a substantial surge of Air Force aircraft and crew, could not benefit from the continuous learning existent in HROs.[116] In addition, the evacuation's complexity had increased the mission count past the point which the software was designed to accommodate. The compressed timeline meant that on-the-fly patches were required to deal with software outages.[117]

In line with the expectations of SDL theory, Kessel Run's software development model was critical to the airlift planning system's ability to limit the risk of accidents. Accounting for this outcome necessitates an understanding of Kessel Run's iterative software engineering process, under which tools like the ones used in the evacuation were developed around cycles of experimentation

[115] Blumenstein 2021.
[116] At one point, half of the Air Force's C-17 transport planes were dedicated to the mission. Horton and Lamothe 2021.
[117] Beachkofski 2022.

with significant user feedback and testing.[118] This approach starkly differs from the software modernization effort that Kessel Run replaced, the "AOC 10.2" program initially awarded to Lockheed Martin, which had limited engagement with operators in the field even a decade after software requirements were established.[119]

The importance of this adaptable approach was validated by Kessel Run's response to a software outage, as the system struggled to accommodate the exponential growth in mission counts demanded by the evacuation effort. In the span of 12 hours, the Kessel Run team implemented a fix, which included adding a new feature that was approved by users in the 609th Air Operations Center.[120] According to one Kessel Run engineer, this adaptation was successful because the team could "dry-run the changes they were making in a sort of digital staging area, while liaison officers — with literally the boots-on-the-ground, in some cases — could communicate with end users to fully realize their immediate needs."[121] As Brian Beachkofski, who led Kessel Run during the evacuation, notes, if this system was developed with a "big bang" release, in which end-users do not engage with the software until after the big product delivery event, this adaptive response would have been impossible.[122]

While the first three case studies serve as the main test ground for competing theories of software's role in military accidents, evidence from this case demonstrates how iterative software development practices can reduce the risk of mishaps. Some qualifying factors should be taken into account. In this scenario, the magnitude of a software mishap was not on the same level as accidents in the other three cases.[123] For systems where the stakes are magnified, such as nuclear command

[118] Budden et al. 2021.
[119] Beachkofski 2022.
[120] Beachkofski 2022.
[121] Coleman 2021.
[122] Interview with Brian Beachkofski, April 10, 2023. This "big bang" release approach is a common byproduct of the waterfall model.
[123] For instance, a software failure could have caused missions to be released without aerial refueling support, which could result in a transport plane having to land in a place unable to accept refugees.

and control, a continuous delivery approach that eschews software requirements specification may not be the most appropriate approach.[124] In addition, Kessel Run's safety benefits were limited to one specific mission, not proven across years of operations. Fortunately, this type of continuous analysis may be more feasible in the future, given that Kessel Run now assists the F-22 and F-35 program offices, which govern some of the military's largest procurement programs.[125]

IV. Conclusion

International relations as a discipline has largely spurned the study of accidents, possibly because they tend to be seen as random events that cannot be explained by general causes.[126] In this line of thinking, civilian casualties are tragic but inevitable costs of warfare; friendly fire incidents are unfortunate, but it would be pointless to try and identify their common determinants. In contrast, this article maintains that a science of accidents is possible.

By centering the system acquisition and development process, I have provided an explanation and evidence for how software technologies contribute to military accidents. When military software development follows a "waterfall" approach, in which input from operators is limited to end stages of testing and deployment, many safety hazards will only be revealed after it is too late to rework system designs. Under this model of software development, the risk of accidents is elevated. Conversely, when the software development process allows for early feedback from military operators during system design and requirements specification, confusing interface designs and other human-machine interaction problems can be mitigated.

[124] Leveson 2019.
[125] The Air Force has mandated one command-and-control monitoring tool used in the Afghanistan evacuation to be used across all Air Combat Command installations. Budden et al. 2021. Pomerleau 2021.
[126] Owens 2003.

This article makes two main interventions in the literature on complex technological systems and military accidents. First, studies of military accidents have primarily tapped into two wells of organizational scholarship, the HRO and NAT approaches, on managing hazardous technologies. By focusing on breakdowns in complex technologies and military organizations in the critical seconds and minutes of a crisis, applications of these two theories neglect the process that occurs before militaries field technological systems: acquisition and development. In extending the causal timeline of military accidents beyond decisions made on the battlefield to those made decades earlier in software design and development, SDL theory presents an alternative way to explain how software contributes to military accidents.

My aim is not to devalue the contributions of NAT and HRO theory to the study of safety in military systems. These two approaches have introduced important concepts like coupling and complexity, and they have underlined the impact of safety culture and organizational structure on accidents.[127] They have also broadened the notion of software failure, beyond the inability of code to meet performance specifications, to encompass problems linked to interactions between software systems and surrounding structures and organizations. Instead, by illustrating some limitations of these two prominent theories as it pertains to military accidents, this article seeks to open new directions for assessing safety in military systems, which highlight the organizational politics of system acquisition.

SDL theory also runs up against limitations, which can serve as jumping-off points for future research. The impact of software acquisition and development pathways on military accidents is constrained by other factors, such as the capacity of military end-users to imagine realistic risks associated with software systems and the extent to which field tests simulate battlefield environments. Furthermore, due to space constraints, this article does not further explore why

[127] Leveson et al. 2009.

militaries adopt software development models like the waterfall approach. On this topic, fruitful inquiries can draw from the literature on the political economy of military expenditures.[128]

Second, this article provides a distinct perspective on the burgeoning discussions of AI-enabled military systems and their safety risks. Revisiting near nuclear confrontations in the Cold War, a *Bulletin of the Atomic Scientists* essay warned, "Today, artificial intelligence, and other new technologies, if thoughtlessly deployed could increase the risks of accidents and miscalculation even further."[129] In these discussions, scholars and policymakers underline "novel" risks, such as those linked to increased speed of decision-making. To be sure, there are many ways that AI systems today differ from the software of old.[130] And, certainly, investigating those unique features will uncover useful insights into understanding how AI will affect military accidents.

At the same time, there is a lot to learn from historical cases. This article uncovers lessons from the development of automation software in older military systems that translate to recent AI advances. In addition to following the latest technical advances in the AI field, debates over the risks of military AI applications must also integrate knowledge from studying past software-intensive military systems. After all, new technologies will not escape old problems.

[128] Alic 2014; Dombrowski and Gholz 2006.
[129] Ruhl 2022.
[130] For an overview of unique safety issues with AI systems built using machine learning and reinforcement learning techniques, see Amodei et al. 2016.

References

Ahern, Tim. “Aegis System Got Poor Marks in GAO Report.” *Associated Press*, July 9, 1988.

Alic, John A. “The Origin and Nature of the US ‘Military-Industrial Complex.’” *Vulcan* 2, no. 1 (2014): 63–97.

Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. “Concrete Problems in AI Safety.” *ArXiv Preprint ArXiv:1606.06565*, 2016.

Aucoin, Joseph. “It’s Not Just the Forward Deployed.” *U.S. Naval Institute Proceedings*, April 2018. https://www.usni.org/magazines/proceedings/2018/april/its-not-just-forward-deployed.

Barry, John, and Roger Charles. “Sea Of Lies.” *Newsweek*, July 12, 1992. https://www.newsweek.com/sea-lies-200118.

Beachkofski, Brian. “Making the Kessel Run.” *Air & Space Forces Magazine* (blog), March 23, 2022. https://www.airandspaceforces.com/article/making-the-kessel-run/.

Beachkofski, Brian, and Dan Patt. “Kessel Run Shows How to Bridge the Gap Between Development and Operations.” *War on the Rocks* (blog), July 28, 2022. https://warontherocks.com/2022/07/kessel-run-shows-how-to-bridge-the-gap-between-development-and-operations/.

Bezooijen, Bart Van, and Eric-Hans Kramer. “Mission Command in the Information Age: A Normal Accidents Perspective on Networked Military Operations.” *Journal of Strategic Studies* 38, no. 4 (June 7, 2015): 445–66. https://doi.org/10.1080/01402390.2013.844127.

Blair, Bruce G. *The Logic of Accidental Nuclear War*. Brookings Institution Press, 1993.

Blatter, Joachim, and Till Blume. “In Search of Co-Variance, Causal Mechanisms or Congruence? Towards a Plural Understanding of Case Studies.” *Swiss Political Science Review* 14, no. 2 (2008): 315–56. https://doi.org/10.1002/j.1662-6370.2008.tb00105.x.

Blumenstein, Richard. “Kessel Run’s C2IMERA Used during Afghan Evacuation.” Air Combat Command, September 23, 2021. https://www.acc.af.mil/News/Article/2786357/kessel-runs-c2imera-used-during-afghan-evacuation/https%3A%2F%2Fwww.acc.af.mil%2FNews%2FArticle-Display%2FArticle%2F2786357%2Fkessel-runs-c2imera-used-during-afghan-evacuation%2F.

Bode, Ingvild. “Practice-Based and Public-Deliberative Normativity: Retaining Human Control over the Use of Force.” *European Journal of International Relations*, April 10, 2023, 13540661231163392. https://doi.org/10.1177/13540661231163392.

Bode, Ingvild, and Thomas Watts. “Meaning-Less Human Control: Lessons from Air Defence Systems on Meaningful Human Control for the Debate on AWS.” Centre for War Studies (University of South Denmark) and Drone Wars UK, 2021. https://dronewars.net/wp-content/uploads/2021/02/DW-Control-WEB.pdf.

Borning, Alan. “Computer System Reliability and Nuclear War.” *Communications of the ACM* 30, no. 2 (1987): 112–31.

Borrie, John. “Safety, Unintentional Risk and Accidents in the Weaponization of Increasingly Autonomous Technologies.” *UNIDIR Resources* 5 (2016).

Budden, Phil, Fiona Murray, Isaac Rahamim, Dylan Brown, and Nick Setterberg. "Kessel Run: An Innovation Opportunity for the US Air Force." MIT Mission Innovation Working Paper, May 2021, https://innovation. mit. edu …, 2021.

Coleman. "Kessel Run's SlapShot Saves Lives." Air Force Life Cycle Management Center, September 28, 2021. https://www.aflcmc.af.mil/NEWS/Article-Display/Article/2791602/kessel-runs-slapshot-saves-lives/https%3A%2F%2Fwww.aflcmc.af.mil%2FNEWS%2FArticle-Display%2FArticle%2F2791602%2Fkessel-runs-slapshot-saves-lives%2F.

Cox, Samuel. "H-020-1: USS Vincennes Tragedy," July 2018. http://public1.nhhcaws.local/content/history/nhhc/about-us/leadership/director/directors-corner/h-grams/h-gram-020/h-020-1-uss-vincennes-tragedy--.html.

Cummings, Mary L. "Automation and Accountability in Decision Support System Interface Design." *Journal of Technology Studies* 32, no. 1 (2006).

Danby, Nick. "How the Downing of Iran Air Flight 655 Still Sparks US-Iran Enmity." *Responsible Statecraft* (blog), July 2, 2021. https://responsiblestatecraft.org/2021/07/02/how-the-downing-of-iran-air-flight-655-still-influences-us-iran-enmity/.

Defense Innovation Board. "Software Acquisition and Practices (SWAP) Study," May 3, 2019. https://innovation.defense.gov/software/.

Dietterich, Thomas G. "Robust Artificial Intelligence and Robust Human Organizations." *Frontiers of Computer Science: Selected Publications from Chinese Universities* 13, no. 1 (February 1, 2019): 1–3. https://doi.org/10.1007/s11704-018-8900-4.

Dobbe, Roel I. J. "System Safety and Artificial Intelligence." In *The Oxford Handbook of AI Governance*, edited by Justin B. Bullock, Yu-Che Chen, Johannes Himmelreich, Valerie M. Hudson, Anton Korinek, Matthew M. Young, and Baobao Zhang, 0. Oxford University Press, 2022. https://doi.org/10.1093/oxfordhb/9780197579329.013.67.

"DOD's Management of Government Property Furnished to Defense Contractors," June 23, 1983.

Dombrowski, Peter, and Eugene Gholz. *Buying Military Transformation: Technological Innovation and the Defense Industry*. F First Edition. New York: Columbia University Press, 2006.

Dotterway, Kristen Ann. "Systematic Analysis of Complex Dynamic Systems: The Case of the USS Vincennes." Master's thesis, Citeseer, 1992.

Eckstein, Megan. "Navy Reverting DDGs Back to Physical Throttles, After Fleet Rejects Touchscreen Controls." *USNI News* (blog), August 9, 2019. https://news.usni.org/2019/08/09/navy-reverting-ddgs-back-to-physical-throttles-after-fleet-rejects-touchscreen-controls.

Edwards, Paul N. *The Closed World: Computers and the Politics of Discourse in Cold War America*. MIT Press, 1996.

Fogarty, William M. *Investigation Report: Formal Investigation into the Circumstances Surrounding the Downing of Iran Air Flight 655 on 3 July 1988*. Department of Defense, 1988.

Ford, David N., and John Dillard. "Modeling the Performance and Risks of Evolutionary Acquisition." *Defense AR Journal* 16, no. 2 (2009): 143.

Foreman, Veronica L., Francesca M. Favaró, Joseph H. Saleh, and Christopher W. Johnson. "Software in Military Aviation and Drone Mishaps: Analysis and Recommendations for the Investigation

Process." *Reliability Engineering & System Safety* 137 (May 1, 2015): 101–11. https://doi.org/10.1016/j.ress.2015.01.006.

Fox, John Ronald. *Defense Acquisition Reform, 1960-2009: An Elusive Goal.* Government Printing Office, 2012.

Frola, F.R., and C.O. Miller. "System Safety in Aircraft Acquisition." Washington D.C.: Logistics Management Institute, January 1984.

Gholz, Eugene, and Harvey M. Sapolsky. "Restructuring the US Defense Industry." *International Security* 24, no. 3 (1999): 5–51.

Goldfarb, Avi, and Jon R. Lindsay. "Prediction and Judgment: Why Artificial Intelligence Increases the Importance of Humans in War." *International Security* 46, no. 3 (February 25, 2022): 7–50. https://doi.org/10.1162/isec_a_00425.

Hawley, John K. "Looking Back at 20 Years of MANPRINT on Patriot: Observations and Lessons." ARMY RESEARCH LAB ADELPHI MD, 2007.

———. "Patriot Wars: Automation and the Patriot Air and Missile Defense System." Center for a New American Security, January 25, 2017. https://www.cnas.org/publications/reports/patriot-wars.

Hawley, John K., and Anna L. Mares. "Human Performance Challenges for the Future Force: Lessons from Patriot after the Second Gulf War." In *Designing Soldier Systems*, 3–34. CRC Press, 2018.

Hilts, Philip J. "AEGIS SYSTEM HAS BEEN CONTROVERSIAL FROM THE START." *Washington Post*, July 7, 1988. https://www.washingtonpost.com/archive/politics/1988/07/07/aegis-system-has-been-controversial-from-the-start/eb4d2ab1-b3a4-40d0-8496-82b1541d924f/.

Hopkins, Andrew. "The Limits of Normal Accident Theory." *Safety Science* 32, no. 2 (1999): 93–102.

Horowitz, Michael C. "When Speed Kills: Lethal Autonomous Weapon Systems, Deterrence and Stability." *Journal of Strategic Studies* 42, no. 6 (September 19, 2019): 764–88. https://doi.org/10.1080/01402390.2019.1621174.

Horowitz, Michael C., Lauren Kahn, Christian Ruhl, Mary Cummings, Erik Lin-Greenberg, Paul Scharre, and Rebecca Slayton, "Policy Roundtable: Artificial Intelligence and International Security," *Texas National Security Review*, June 2020, https://tnsr.org/roundtable/policy-roundtable-artificial-intelligence-and-international-security/.

Horowitz, Michael C., Paul Scharre, and Alexander Velez-Green. "A Stable Nuclear Future? The Impact of Autonomous Systems and Artificial Intelligence." *ArXiv:1912.05291 [Cs]*, December 13, 2019. http://arxiv.org/abs/1912.05291.

Horton, Alex, and Dan Lamothe. "Inside the Afghanistan Airlift: Split-Second Decisions, Relentless Chaos Drove Historic Military Mission." *Washington Post*, September 28, 2021. https://www.washingtonpost.com/national-security/2021/09/27/afghanistan-airlift-inside-military-mission/.

Lambeth, Benjamin S. *The Unseen War: Allied Air Power and the Takedown of Saddam Hussein*. Naval Institute Press, 2013.

LaPorte, Todd R., and Paula M. Consolini. "Working in Practice But Not in Theory: Theoretical Challenges of 'High-Reliability Organizations.'" *Journal of Public Administration Research and Theory* 1, no. 1 (January 1, 1991): 19–48. https://doi.org/10.1093/oxfordjournals.jpart.a037070.

Lerner, Eric J. "Lessons of Flight 655." *Aerospace America* 27, no. 4 (1989): 18.

Leveson, Nancy. "An Engineering Perspective on Avoiding Inadvertent Nuclear War." NAPSNet Special Reports, July 26, 2019. https://nautilus.org/napsnet/napsnet-special-reports/an-engineering-perspective-on-avoiding-inadvertent-nuclear-war/.

Leveson, Nancy, Nicolas Dulac, Karen Marais, and John Carroll. "Moving Beyond Normal Accidents and High Reliability Organizations: A Systems Approach to Safety in Complex Systems." *Organization Studies* 30, no. 2–3 (February 1, 2009): 227–49. https://doi.org/10.1177/0170840608101478.

Leveson, Nancy G. *Engineering a Safer World*. The MIT Press, 2012.

Lofgren, Eric. "Is Poor Software Design/Testing the REAL Cause of the USS McCain Crash?" *Acquisition Talk* (blog), December 25, 2019. https://acquisitiontalk.com/2019/12/is-poor-software-design-testing-the-real-cause-of-the-uss-mccain-crash/.

Maas, Matthijs M. "Regulating for'Normal AI Accidents' Operational Lessons for the Responsible Governance of Artificial Intelligence Deployment." In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 223–28, 2018.

Maclean, Dirk. *Shoot, Don't Shoot: Minimising Risk of Catastrophic Error Through High Consequence Decision-Making*. Air Power Development Centre, 2017.

Malachowski, James A. "Robots and Rogue Thinkers: Leveraging Organizational Learning Theory to Prevent Catastrophic Failure After Rapid Fielding of Disruptive Technology." NAVAL WAR COLLEGE NEWPORT RI NEWPORT United States, 2020.

Mallam, S. C., K. Nordby, S. O. Johnsen, and F. B. Bjørneseth. "The Digitalization of Navigation: Examining the Accident and Aftermath of US Navy Destroyer John S. McCain." *Proceedings of the Royal Institution of Naval Architects Damaged Ship V*, 2020, 55–63.

Marshall, Eliot. "Fatal Error: How Patriot Overlooked a Scud." *Science* 255, no. 5050 (March 13, 1992): 1347–1347. https://doi.org/10.1126/science.255.5050.1347.

McDonald, Christopher. "From Art Form to Engineering Discipline? A History of US Military Software Development Standards, 1974–1998." *IEEE Annals of the History of Computing* 32, no. 4 (October 2010): 32–47. https://doi.org/10.1109/MAHC.2009.58.

Miller, T. Christian, Megan Rose, Robert Faturechi, and Agnes Chang. "The Navy Installed Touch-Screen Steering Systems To Save Money. Ten Sailors Paid With Their Lives." *ProPublica*, December 20, 2019. https://features.propublica.org/navy-uss-mccain-crash/navy-installed-touch-screen-steering-ten-sailors-paid-with-their-lives/.

Mindell, David A. *Between Human and Machine: Feedback, Control, and Computing before Cybernetics*. Johns Hopkins University Press, 2002.

Ministry of Defence (United Kingdom). "Aircraft Accident to Royal Air Force Tornado GR MK4A ZG710," March 23, 2003. https://www.gov.uk/government/publications/military-aircraft-accident-summary-aircraft-accident-to-raf-tornado-gr-mk4a-zg710.

National Research Council. *Achieving Effective Acquisition of Information Technology in the Department of Defense*. Washington, D.C.: National Academies Press, 2010. https://doi.org/10.17226/12823.

National Transportation Safety Board. "Collision between US Navy Destroyer John S McCain and Tanker Alnic MC Singapore Strait, 5 Miles Northeast of Horsburgh Lighthouse August 21, 2017," 2019. https://www.ntsb.gov/investigations/AccidentReports/Reports/MAR1901.pdf.

"On the Subject of the USS Vincennes Downing of Iran Air Flight 655," September 14, 1988.

Owens, Patricia. "Accidents Don't Just Happen: The Liberal Politics of High-Technology `Humanitarian' War." *Millennium* 32, no. 3 (December 1, 2003): 595–616. https://doi.org/10.1177/03058298030320031101.

"Patirot System Performance." Washington, D.C.: United States Dept. of Defense, Office of the Under Secretary of Defense For Acquisition, Technology, and Logistics, 2005.

Perrow, Charles. *Normal Accidents: Living with High Risk Technologies*. New York: Basic Books, 1984.

Piller, Charles. "Vaunted Patriot Missile Has a 'Friendly Fire' Failing." *Los Angeles Times*, April 21, 2003, sec. World & Nation. https://www.latimes.com/archives/la-xpm-2003-apr-21-war-patriot21-story.html.

Pomerleau, Mark. "Air Force Software Tool Helped Coordinate Afghanistan Evacuation of Civilians." C4ISRNet, September 23, 2021. https://www.c4isrnet.com/battlefield-tech/c2-comms/2021/09/23/air-force-software-tool-helped-coordinate-afghanistan-evacuation-of-civilians/.

Postol, Theodore A. "An Informed Guess about Why Patriot Fired upon Friendly Aircraft and Saw Numerous False Missile Targets during Operation Iraqi Freedom." Security Studies Program, Massachusetts Institute of Technology, April 20, 2004.

———. "Lessons of the Gulf War Experience with Patriot." *International Security* 16, no. 3 (1991): 119–71. https://doi.org/10.2307/2539090.

"Review of the Office of Operational Test and Evaluation: Hearing before the Acquisition Policy Panel Committee on Armed Services," September 14, 1988.

Roberts, Karlene H. "Some Characteristics of One Type of High Reliability Organization." *Organization Science* 1, no. 2 (1990): 160–76.

Roberts, Karlene H., Denise M. Rousseau, and Todd R. La Porte. "The Culture of High Reliability: Quantitative and Qualitative Assessment Aboard Nuclear-Powered Aircraft Carriers." *The Journal of High Technology Management Research* 5, no. 1 (1994): 141–61.

Roberts, Nancy C., and Kristen Ann Dotterway. "The Vincennes Incident: Another Player on the Stage?" *Defense Analysis* 11, no. 1 (1995): 31–45.

Rochlin, Gene I. "Iran Air Flight 655: Complex, Large-Scale Military Systems and the Failure of Control." *Responding to Large Technical Systems: Control or Anticipation*, 1991, 95–121.

———. *Trapped in the Net: The Unanticipated Consequences of Computerization*. Princeton University Press, 1997. https://www.jstor.org/stable/j.ctt7rq0c.

Rowen, Aaron, Martha Grabowski, and Dale Russell. "The Impact of Work Demands and Operational Tempo on Safety Culture, Motivation and Perceived Performance in Safety Critical Systems." *Safety Science* 155 (November 1, 2022): 105861. https://doi.org/10.1016/j.ssci.2022.105861.

Ruhl, Christian. "Sixty Years after the Cuban Missile Crisis, How to Face a New Era of Global Catastrophic Risks." *Bulletin of the Atomic Scientists* (blog), October 13, 2022. https://thebulletin.org/2022/10/sixty-years-after-the-cuban-missile-crisis-how-to-face-a-new-era-of-global-catastrophic-risks/.

Sagan, Scott D. "Rules of Engagement." *Security Studies* 1, no. 1 (September 1, 1991): 78–108. https://doi.org/10.1080/09636419109347458.

———. *The Limits of Safety*. Princeton University Press, 1993. https://press.princeton.edu/books/paperback/9780691021010/the-limits-of-safety.

Scharre, Paul. *Army of None: Autonomous Weapons and the Future of War*. W. W. Norton & Company, 2018.

———. "Autonomous Weapons and Operational Risk." Center for a New American Security, 2016. https://s3.us-east-1.amazonaws.com/files.cnas.org/documents/CNAS_Autonomous-weapons-operational-risk.pdf?mtime=20160906080515&focal=none.

Schlosser, Eric. *Command and Control: Nuclear Weapons, the Damascus Accident, and the Illusion of Safety*. Penguin, 2009.

Schmitt, Eric. "AFTER THE WAR; Army Is Blaming Patriot's Computer For Failure to Stop the Dhahran Scud." *The New York Times*, May 20, 1991, sec. World. https://www.nytimes.com/1991/05/20/world/after-war-army-blaming-patriot-s-computer-for-failure-stop-dhahran-scud.html.

Schneider, Jacquelyn, and Julia Macdonald. "Looking Back to Look Forward: Autonomous Systems, Military Revolutions, and the Importance of Cost." *Journal of Strategic Studies* 0, no. 0 (January 24, 2023): 1–23. https://doi.org/10.1080/01402390.2022.2164570.

Sheridan, Thomas B. *Humans and Automation: System Design and Research Issues*. Wiley, 2002. https://www.wiley.com/en-us/Humans+and+Automation%3A+System+Design+and+Research+Issues-p-9780471234289.

Slayton, Rebecca. "The Fallacy Of Proven And Adaptable Defenses." Public Interest Report. Federation of American Scientists, 2014.

Smith, R. Jeffrey. "DECISION ON VINCENNES ECHOES PRECEDENT." *Washington Post*, August 20, 1988. https://www.washingtonpost.com/archive/politics/1988/08/20/decision-on-vincennes-echoes-precedent/069c5094-670b-42ed-b2ae-d10b904354cb/.

Soare, Simona R., Pavneet Singh, and Meia Nouwens. "Software-Defined Defence: Algorithms at War." The International Institute for Strategic Studies, February 2023. https://www.iiss.org/research-paper//2023/02/software-defined-defence.

The Straits Times. "After USS John S. McCain Collision, Chinese Press Say US Navy a Hazard," August 22, 2017. https://www.straitstimes.com/asia/east-asia/navy-collisions-show-us-combat-readiness-and-military-management-has-declined-chinese.

U.S. Fleet Forces Command. "Comprehensive Review of Recent Surface Forces Incidents." Department of the Navy, October 26, 2017. https://news.usni.org/2017/11/02/document-navy-comprehensive-review-surface-forces.

U.S. General Accounting Office. "Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia," February 1992. https://apps.dtic.mil/sti/citations/ADA344865.

U.S. Government Accountability Office. "Weapons Testing: Quality of DOD Operational Testing and Reporting," July 26, 1988. https://www.gao.gov/products/pemd-88-32br.

Vanden Brook, Tom, and Kim Hjelmgaard. "US Air Force Says It Did Not Run Simulation in Which AI Drone 'Killed Its Operator.'" USA TODAY. Accessed June 6, 2023. https://www.usatoday.com/story/news/politics/2023/06/02/us-air-force-denies-ai-drone-simulation-killed-its-operator/70280780007/.

Vaughan, Diane. "Autonomy, Interdependence, and Social Control: NASA and the Space Shuttle Challenger." *Administrative Science Quarterly* 35, no. 2 (1990): 225–57. https://doi.org/10.2307/2393390.

Yergin, Daniel. "The World's Most Important Body of Water." *The Atlantic*, December 15, 2020. https://www.theatlantic.com/international/archive/2020/12/south-china-sea-us-ghosts-strategic-tensions/617380/.