2022

Reflecting on Recurring Failures in IoT Development

Dharun Anandayuvaraj
Purdue University, dananday@purdue.edu

James C. Davis
Purdue University, davisjam@purdue.edu

Reflecting on Recurring Failures in IoT Development

Dharun Anandayuvaraj
Purdue University, IN, USA
dananday@purdue.edu

James C. Davis
Purdue University, IN, USA
davisjam@purdue.edu

ABSTRACT

As IoT systems are given more responsibility and autonomy, they offer greater benefits, but also carry greater risks. We believe this trend invigorates an old challenge of software engineering: how to develop high-risk software-intensive systems safely and securely under market pressures? As a first step, we conducted a systematic analysis of recent IoT failures to identify engineering challenges. We collected and analyzed 22 news reports and studied the sources, impacts, and repair strategies of failures in IoT systems. We observed failure trends both within and across application domains. We also observed that failure themes have persisted over time. To alleviate these trends, we outline a research agenda toward a Failure-Aware Software Development Life Cycle for IoT development. We propose an encyclopedia of failures and an empirical basis for system postmortems, complemented by appropriate automated tools.

CCS CONCEPTS

• **Computer systems organization → Cyber-physical systems**.

KEYWORDS

Internet of Things, Cyber-Physical Systems, Embedded Systems, Safety-Critical Software, Software Engineering, Failure Analysis

ACM Reference Format:
Dharun Anandayuvaraj and James C. Davis. 2022. Reflecting on Recurring Failures in IoT Development. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3551349.3559545

1 INTRODUCTION

Internet of Things (IoT) realizes a modern world of smart devices interconnected by complex networks. Many IoT systems are software-intensive with hardware components off-the-shelf. IoT systems enable software to directly interact with the physical environment [20]. These interactions are divided into observations of the physical world using *sensors*, and effects on the physical world using *actuators* [20]. A plethora of technological breakthroughs, in batteries, hardware, wireless networking, mobile computing, Cloud services, and machine learning, has enabled widespread adoption of IoT systems [26]. These trends have enabled IoT systems to become pervasive [41] and increasingly interactive with the physical world. Their failures are often safety-critical [27]. Given the complexities of IoT systems, their diverse characteristics enable diverse failures.

ASE '22, October 10–14, 2022, Rochester, MI, USA

ACM ISBN 978-1-4503-9475-8/22/10.
https://doi.org/10.1145/3551349.3559545

In this paper, we studied IoT failures in the wild to identify engineering challenges. We conducted the first systematic study of IoT failures as reported in the media. Specifically, we studied the sources, impacts, and repair recommendations of IoT failures. We observed trends in failure both within and across application domains. We also observed that failure themes have persisted over time. To alleviate these trends, we outline a research agenda towards a Failure-Aware Software Development Life Cycle.

2 BACKGROUND AND RELATED WORK

The study of engineering failures is essential to successful design [32]. In the current state of software engineering research there is a gap to study and learn from IoT *failures*. Researchers have already studied *faults* in IoT systems (*e.g.,* bug studies [19, 28, 40, 44]). The distinction is that a fault is a defect within a system, whereas a failure occurs when one or more faults cause the system to fail in its required function [37]. Not all faults are created equal; the study of failures informs failure mode analysis [22, 34, 36] and helps researchers and engineers prioritize risks [17].

While the software engineering field has benefited from empirical and qualitative failure analysis research [13, 27], no such systematic studies have been conducted to learn from IoT failures. Such works require detailed information about the failures, which are often obtained through sources such as lawsuits and government investigations [27]. These sources of information are still limited for the emerging IoT field. Lacking detailed failure reports, we begin with news reports. We acknowledge that news reports are not ideal data sources, but in the absence of detailed failure reports, it is a starting point. In civil engineering research, news reports have been used as primary data sources to attain information for infrastructure failure interdependencies [43]. This methodology is also used in economics [23], public health research [18], and agriculture [16].

3 METHODOLOGY

Our goal is to systematically identify IoT failure events, their sources, and their impacts. Our approach is to conduct a qualitative analysis of failure reports covered by news reports. Specifically, we investigate two research questions:

RQ1: What are the common sources of IoT failures?
RQ2: What are the common impacts of IoT failures?

3.1 Data Collection

We collected IoT failures covered by reputable news sources. We selected *The New York Times* for its focus on general audience [5], and *WIRED* for its popular coverage of technology [10]. We used Google News to search both sources to maintain reproducibility. We used IoT related search terms [1] and limited the search from

[1]Search Terms: "iot, internet of thing, cyber physical system, autonomous, cyber security fail, monitor fail, device fail, software fail, computer fail"

January 2015 to October 2021 [2]. Through this search criteria we collected a sample of IoT failures.

For each search, we filtered out articles based on their titles, and subsequently their content. We included any article that described the *failure* of an *IoT system*, following the definitions given in sections 1 and 2. When multiple articles described the same event, the article with greater detail was selected.

3.2 Data Analysis

Content extraction: As a first-pass analysis we read each article, and identified the sources of failure and the impacts of failure of the IoT systems. If available in the text, repair recommendations were also identified. Sources referenced in the articles were also reviewed for supplementary understanding of the contents.

Organization of knowledge: We then performed a structured analysis of the qualitative data to code and categorize the data by sources of faults and taxonomy of faults. We followed the framework of Melo & Aquino [30], which provides definitions and characterizations for studying IoT failures. The framework divides IoT systems into four levels (*i.e.,* layers) in which failures can occur: perception (components for obtaining and processing data), communication (wired and wireless media), service (*e.g.,* data storing and processing services), and application (*e.g.,* user interface). The framework also taxonomizes the behavior and impacts of a failure: duration (transient, permanent, intermittent), fault origin location (internal: within system, external: from environment), semantics (crash, omission, timing, value, arbitrary), change in behavior (soft: altered, hard: inactive), and dimension (software, hardware).

From each news source, we identified 1–3 faults leading to failures. We report on one primary fault for each failure.

4 RESULTS AND ANALYSIS

The news articles that matched our search criteria are summarized in Table 1. Our search query yielded 570 results, from which we identified 22 articles that reported on IoT failures. Searching *The New York Times* yielded 14 articles, and *WIRED* yielded 8 articles. The 22 distinct failure events originated from 18 distinct organizations. The events represent 5 categories of IoT applications [8]. The automotive category was the most common (8/22), followed by critical infrastructure (6/22), consumer products (4/22), healthcare (2/22), and aerospace (2/22).

RQ1: Common sources of IoT failures. The sources of faults are presented in Figure 1. The most common source of faults originated at the application level (9/22), followed by connectivity at the communication level (7/22). Other sources of faults originated from embedded software and transducers at the perception level, links at the communication level, and the service level. Furthermore, Figure 2 shows that failure-triggering events occurred both within and outside of the system, and that *all faults were traceable to the behavior of software components.*

We observed failure trends both within and across application domains. In the automotive domain, *functional failures* were more common, because of a reliance on cutting-edge (and faulty) computer vision components (ID 7, 8, 10, 11, 12). In the healthcare domain, *the lack of a safe state* led to failures, specifically when network connectivity was lost (ID 19, 20). Across domains, *using a system outside of its intended specification* led to failures in autonomous cars (ID 7, 8, 10, 11, 12), consumer health monitors (ID 19, 20), and smart home products (ID 16). Another cross-domain cause of failure was *insecure remote access and authentication*, affecting critical infrastructure (ID 1, 3, 4, 5, 6), connected cars (ID 9, 14), and consumer products (ID 15, 18). These cybersecurity failures could often be attributed to misplaced trust in vendored components (*i.e.,* the software supply chain). *Improper isolation between safety-relevant vs. application software* led to failures in critical infrastructures (ID 2, 5) and connected cars (ID 9, 13, 14). *Incorrect software evolution* led to failures in a smart home product (ID 22) and an aircraft (ID 17). Additionally, we observed that 14 of the failure events were a result of multiple sources of failures (ID 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 22). This observation indicates opportunities to better exercise accident management techniques (*i.e.,* Swiss cheese model [33]) for IoT development.

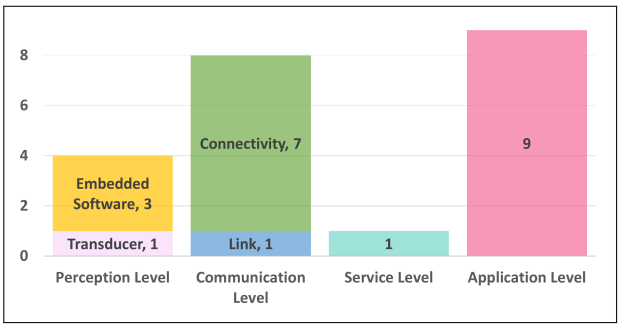

Figure 1: Sources of faults (*N=22*). Each failure is categorized into one source of fault at the system level.

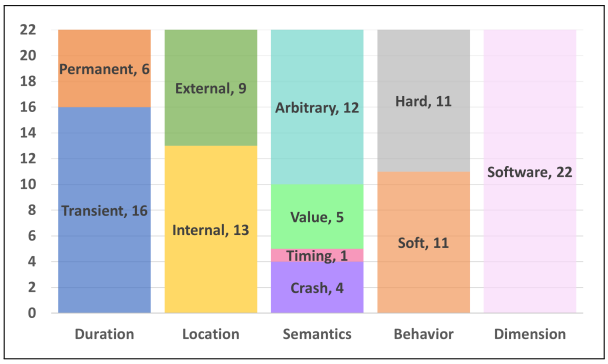

Figure 2: Taxonomy of faults (*N=22*). Each bar captures a different attribute of each failure.

RQ2: Common impacts of IoT failures. The taxonomy of faults is presented in Figure 2. Many of the systems behaved arbitrarily (12/22) when a fault was active, primarily due to security breaches. Most faults were temporary (16/22), some permanent (6/22), none intermittent. Furthermore, half of the systems exhibited an altered behavior, and half of the systems exhibited an inactive behavior when faults occurred.

General trends amongst the impacts of failures were primarily observed within the application domains (Table 1). Failures in critical infrastructures (ID 1, 2, 3, 4, 5, 6) led to unauthorized access to critical functions, and their impacts were extensive with respect to

[2]Google News Search Syntax: "[SearchTerm] site:[SourceWebsite] after:[StartDate: Year-Month-Day] before:[EndDate: Year-Month-Day]"

Table 1: IoT failures, organized by Cat(egory). ***'ID' denotes row in artifact (§8). Repair recommendations marked '+' are directly from articles.***

Cat.	ID	Date	System	Failure Source	Failure Impact	Repair Recommendation
Critical infrastructure	1	2021	Water works	Lack of seperation between IT & OT network	City's water poisoned	Remove remote access+
	2	2021	Oil pipeline	Hack of business network; lack of separation between business & operation networks	Shutdown of a major oil pipeline	Isolate critical infrastructure network+
	3	2021	Public transportation	Zero day attack in remote access software; malware web shells	Unauthorized access; $370,000 response cost	Scan periodically for backdoor web shells+
	4	2019	Water treatment facility	Un-updated discharged personnel authentication; unsecure remote access	Unauthorized access; altered cleaning & disinfecting process	
	5	2015	Power plant	Unsecure supply chain network; malware repositories	Unauthorized access; leak of critical information	
	6	2015	Power distribution center	Unsecure remote access; lack of 2FA for SCADA	Loss of power for 230,000 residents	Enable 2FA for safety-critical remote access+
Automotive	7	2019	Autonomous car	Failed to detect parked vehicle, stop sign, flashing red lights; ineffective driver engagement monitor	Fatal collision	Stringent user engagement monitor+
	8	2019	Autonomous car	Failed to detect merging vehicle; neglected radar data; ineffective driver engagement monitor	Fatal collision	Create redundancy in object detection system+; stringent user engagement monitor+
	9	2019	Connected car	Default authentication for user accounts; lack of isolation for safety-critical functions	Unauthorized access to safety-critical functions	Disable default authentication+
	10	2018	Autonomous car	Ineffective parked vehicle identification; ineffective driver engagement monitor	Fatal collision	Stringent user engagement monitor+
	11	2018	Autonomous car	Failed to detect road barrier; ineffective driver engagement monitor	Fatal collision	Stringent user engagement monitor+
	12	2016	Autonomous car	Failed to detect turning vehicle & stop at collision; ineffective driver engagement monitor	Fatal collision	
	13	2015	Connected car	Zero day exploit of entertainment system firmware	Unauthorized access to safety-critical functions	Isolate safety-critical functions+; auto monitor CAN bus+
	14	2015	Connected car	Unsecure remote access protocol (SMS, SSH w/ default key) in telematics dongle	Unauthorized access to safety-critical functions	Secure remote access+; isolate safety-critical functions+
Consumer products	15	2019	Connected real-time OS	Networking protocol bug in COTS; Lack of software bill of materials	Exploitable vulnerabilities	Maintain software bill of materials+
	16	2018	Smart home products	Lack of all users' consent; abuse enabling user experience	Domestic abuse; psychological stress	
	17	2016	Smart thermostat	Bug in software update	Battery depletion; Unprompted temperature decrease	
	18	2016	Old connected devices	Processors with default authentication	Distributed denial-of-service (DDoS) botnet; internet outage	Disable default authentication+
Healthcare	19	2019	Smart baby vital monitor	Device to server to phone app connection loss	False sense of safety	
	20	2019	Smart diabetes monitor	Device to server to phone app connection loss	False sense of safety	Alert when connection lost+
Aerospace	21	2019	Spacecraft	Vendored software with bug caused early communication query	Early state change depleted fuel	
	22	2019	Aircraft	Faulty sensor; lack of sensor redundancy; lack of updated system training	Catastrophic crashes	Create redundancy for critical systems+

human impact and monetary cost. Failures in autonomous cars (ID 7, 8, 10, 11, 12) led to fatal collisions, and failures in connected cars (ID 9, 13, 14) led to unauthorized access to safety-critical functions of the car. Failures in consumer products resulted in varying impacts including DDoS attacks (ID 18) and domestic abuse (ID 16). Failures in consumer health monitors (ID 19, 20) led to a false sense of safety. Failures in aerospace systems led to severe impacts such as ill-timed state changes (ID 21) and aircraft crashes (ID 22).

5 DISCUSSION

Since the unique challenges of IoT lie in the complexity of system design [9], it is beneficial to study failures from the system perspective. Previous works have largely examined *faults* present in IoT *software* rather than *failures* of IoT *systems* [19, 28, 40, 44]. Because of the focus on faults, they contain limited information about failure impacts (risks). Moreover, these works primarily study open-source software, which may not be representative of commercial systems [38]. In contrast, our data provides system-level information about the failures of IoT systems. The use of news reports documenting IoT failures enabled us to identify distinct faults along with their human impacts. This approach enables engineers to focus on faults that lead to catastrophic failures in IoT systems.

Our results indicate that half of the failures were due to cybersecurity issues (ID 1, 2, 3, 4, 5, 6, 9, 13, 14, 15, 18). This finding adds weight to government and industry calls to reinforce the security of IoT systems. Researchers could investigate whether these failures could have been prevented by following government guidelines [4], industry guidelines [6], and international standards [24].

Lastly, we observed echoes of past failures in modern IoT systems. For example, software evolution was a cause of the 1980s Therac-25 failure [27], and is still manifesting (ID 22). Within critical infrastructure, lack of isolation of safety-critical functions and insecure remote access led to unauthenticated access (ID 2, 5), a concern also raised by the National Research Council in 2007 based on failures in the 1990s-2000s [13].

6 RESEARCH AGENDA

Although we took a system-level perspective on IoT failures, it appears that many of these failures can be traced to problems in software and system engineering (Figure 2). The persistence of failure themes over time is unsurprising — these are some of the core challenges of software engineering — but we suggest that the software engineering discipline might benefit from development processes that place a greater emphasis on mitigating past failures. We therefore propose three research directions towards a Failure-Aware Software Development Life Cycle for IoT (Figure 3).

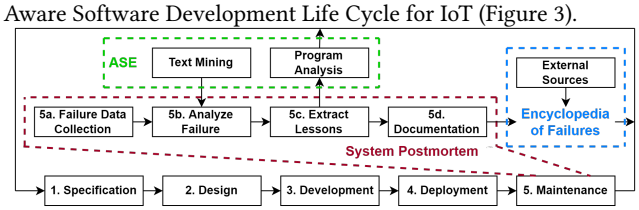

Figure 3: A Failure-Aware Software Development Life Cycle.

Infrastructure: A Failure Encyclopedia. To help IoT engineers anticipate failures, we believe they would benefit from an encyclopedia of previous IoT failures modeled on Table 1. The closest existing knowledge base is the CVE database of cybersecurity vulnerabilities [2], which deliberately omits root cause information — protecting affected users, but limiting the value for other engineering teams. A catalog of case studies could outline the failure, the underlying fault(s), the impact, and lessons learned from a system failure. Case studies could be built from within teams and organizations, as well as from external sources such as news reports or other organizations. These case studies could inform software engineering judgment [29, 37]. Additionally, they could help engineers determine the extent to which failures can be attributed to different stages of the engineering process. This would provide engineers with a heuristic for investing their time in different stages.

Process: An Empirical Basis for Postmortems. Our analysis of failures was restricted because we could only observe what was reported by journalists. IoT engineers working on the affected products could conduct a more detailed analysis to benefit their own and other teams, through a failure postmortem. Although postmortems are widely recommended [11, 12, 14], they are often omitted [15, 25, 39]. We know surprisingly little about postmortem practices in software engineering. We therefore recommend research to establish an empirical basis for software failure postmortems. For example, what are effective personal and team practices to collect, analyze, and document system failures? How can postmortem knowledge be integrated into the software development cycles, managing the tradeoff between agility and risk management? What mechanisms would support measuring the impact of postmortems in mitigating or recovering from failures?

Tools: Automation. There are many opportunities to automate elements of this research agenda. We outline one three-part sequence based on Figure 3. First, there are many computing failures in the news, including both IoT and IT software. Researchers could leverage text mining techniques (NLP) to extract system postmortem information from diverse representations, including news reports [42], user complaints [21], and open-source issue reports. This would facilitate the organization of a large encyclopedia of failures. Then, an engineering team would want to query this database for relevant failures to guide their system design or maintenance work; how can they filter thousands of cases down to the ones relevant in their context, or use machine learning to transfer lessons across contexts [35]? Finally, during validation, an engineering team might want to scan their system model or their codebase for known hazards, *e.g.*, using program analysis techniques [1, 3, 7] to identify anti-patterns. In the IoT context, an end-to-end scan might be difficult, suggesting a human-in-the-loop approach.

7 THREATS TO VALIDITY

We note two sources of methodological bias. First, the raw data: journalists might dramatize the impacts of IoT failures, and editors may sway which events are covered [31]. Second, only one researcher coded the data; a second author reviewed the process. News articles are designed for lay readers; coding was straightforward.

8 CONCLUSION

In this work, we studied real-world IoT systems to better understand the sources and impacts of IoT failures. We observed persistent failure trends both within and across application domains. We outlined a research agenda towards a Failure-Aware Software Development Life Cycle for IoT development.

Data: https://doi.org/10.5281/zenodo.7033016.

REFERENCES

[1] [n. d.]. CodeQL. https://codeql.github.com/
[2] [n. d.]. Common Vulnerabilities and Exposures. https://www.cve.org. Accessed: 2021-05-23.
[3] [n. d.]. Docs home | Semgrep. https://semgrep.dev/docs/
[4] 2017. *Baseline Security Recommendations for IoT.* Report/Study. ENISA.
[5] 2021. Credibility of Major News Organizations in the U.S. 2021. https://www.statista.com/statistics/239784/credibility-of-major-news-organizations-in-the-us/.
[6] 2021. *Internet of Things (IoT) Security Best Practices.* Technical Report. Microsoft.
[7] Mohannad Alhanahnah, Clay Stevens, and Hamid Bagheri. 2020. Scalable analysis of interaction threats in iot systems. In *Proceedings of the 29th ACM SIGSOFT international symposium on software testing and analysis*. 272–285.
[8] P. Augustine, P. Raj, and S. Munirathinam. 2022. *Enterprise Digital Transformation: Technology, Tools, and Use Cases.* CRC Press.
[9] Saurabh Bagchi, Tarek F. Abdelzaher, Ramesh Govindan, et al. 2020. New Frontiers in IoT: Networking, Systems, Reliability, and Security Challenges. *IEEE Internet of Things Journal* (2020).
[10] Bethany Baker. 2020. Wired Magazine. https://ischoolwikis.sjsu.edu/lispublications/wiki/civilian-publications/wired-magazine/.
[11] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016. *Site reliability engineering: How Google runs production systems.* " O'Reilly Media, Inc.".
[12] A. Birk, T. Dingsoyr, and T. Stalhane. 2002. Postmortem: Never Leave a Project without It. *IEEE Software* 19, 3 (May 2002), 43–45. https://doi.org/10.1109/MS.2002.1003452
[13] National Research Council. 2007. *Software for Dependable Systems: Sufficient Evidence?* The National Academies Press.
[14] Darren Dalcher. 1994. Falling down is part of growing Up; the study of failure and the Software Engineering community. In *Software Engineering Education*, Jorge L. Díaz-Herrera (Ed.). Vol. 750. Springer-Verlag, Berlin/Heidelberg, 489–496. https://doi.org/10.1007/BFb0017636 Series Title: Lecture Notes in Computer Science.
[15] Torgeir Dingsøyr. 2005. Postmortem reviews: purpose and approaches in software engineering. *Information and Software Technology* 47, 5 (March 2005), 293–303. https://doi.org/10.1016/j.infsof.2004.08.008
[16] Brett Drury and Mathieu Roche. 2019. A Survey of the Applications of Text Mining for Agriculture. *Computers and Electronics in Agriculture* (Aug. 2019).
[17] George Fairbanks. 2010. *Just Enough Software Architecture: A Risk-driven Approach.* Marshall & Brainerd. Google-Books-ID: ITsWdAAzVYMC.
[18] Vasiliki Foufi, Tatsawan Timakum, et al. 2019. Mining of Textual Health Information from Reddit: Analysis of Chronic Diseases With Extracted Entities and Their Relations. *Journal of Medical Internet Research* (2019).
[19] Joshua Garcia, Yang Feng, Junjie Shen, et al. 2020. A Comprehensive Study of Autonomous Vehicle Bugs. In *International Conference on Software Engineering*.
[20] Christopher Greer, Martin Burns, David Wollman, et al. 2019. Cyber-Physical Systems and Internet of Things. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1900-202.pdf
[21] Hui Guo and Munindar P. Singh. 2020. Caspar: Extracting and Synthesizing User Stories of Problems from App Reviews. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 628–640. ISSN: 1558-1225.
[22] Takuto Ishimatsu, Nancy G Leveson, John Thomas, Masa Katahira, Yuko Miyamoto, and Haruka Nakao. 2010. Modeling and hazard analysis using STPA. (2010). Publisher: International Association for the Advancement of Space Safety (IAASS).
[23] Mark Johnman, Bruce Vanstone, and Adrian Gepp. 2018. Predicting FTSE 100 Returns and Volatility Using Sentiment Analysis. *Accounting & Finance* (2018).
[24] JTC 1/SC 27. 2021. *Cybersecurity - IoT Security and Privacy - Guidelines.* Technical Report DIS 27400. ISO/IEC.
[25] Vijay Kasi, Mark Keil, Lars Mathiassen, and Keld Pedersen. 2008. The Post Mortem Paradox: A Delphi Study of IT Specialist Perceptions. *European Journal of Information Systems* 17, 1 (Feb. 2008), 62–78. https://doi.org/10.1057/palgrave.ejis.3000727
[26] In Lee and Kyoochun Lee. 2015. The Internet of Things (IoT): Applications, Investments, and Challenges for Enterprises. *Business Horizons* (2015).
[27] Nancy G. Leveson. 1995. *Safeware: System Safety and Computers.* ACM.
[28] Amir Makhshari and Ali Mesbah. 2021. IoT Bugs and Development Challenges. In *International Conference on Software Engineering*.
[29] Andrew McNamara, Justin Smith, and Emerson Murphy-Hill. 2018. Does ACM's Code of Ethics Change Ethical Decision Making in Software Development?. In *ESEC/FSE*.
[30] Mário Melo and Gibeon Aquino. 2021. The Pathology of Failures in IoT Systems. In *Computational Science and Its Applications – ICCSA*.
[31] Sendhil Mullainathan and Andrei Shleifer. 2002. Media Bias. (2002).
[32] Henry Petroski. 1994. *Design Paradigms: Case Histories of Error and Judgment in Engineering.* Cambridge University Press.
[33] J. Reason. 1990. The Contribution of Latent Human Failures to the Breakdown of Complex Systems. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 327, 1241 (1990), 475–484. http://www.jstor.org/stable/55319 Publisher: The Royal Society.
[34] Donald J. Reifer. 1979. Software Failure Modes and Effects Analysis. *IEEE Transactions on Reliability* R-28, 3 (Aug. 1979), 247–249. https://doi.org/10.1109/TR.1979.5220578
[35] Donald A Schon. 1984. *The reflective practitioner: How professionals think in action.* Vol. 5126. Basic books.
[36] Yao Song. 2012. *Applying system-theoretic accident model and processes (STAMP) to hazard analysis.* Ph. D. Dissertation.
[37] Neil R. Storey. 1996. *Safety Critical Computer Systems.* Addison-Wesley Longman Publishing Co., Inc.
[38] Lin Tan, Chen Liu, Zhenmin Li, et al. 2014. Bug Characteristics in Open Source Software. *Empirical Software Engineering* (2014).
[39] Felipe J. R. Vieira, Manoela R. Oliveira, Rogério P. C. do Nascimento, and Michel S. Soares. 2019. Technical and Managerial Difficulties in Postmortem Analysis in Software Projects. In *Computational Science and Its Applications – ICCSA 2019*, Sanjay Misra, Osvaldo Gervasi, Beniamino Murgante, Elena Stankova, Vladimir Korkhov, Carmelo Torre, Ana Maria A.C. Rocha, David Taniar, Bernady O. Apduhan, and Eufemia Tarantino (Eds.). Vol. 11623. Springer International Publishing, Cham, 59–69. https://doi.org/10.1007/978-3-030-24308-1_5
[40] Dinghua Wang, Shuqing Li, Guanping Xiao, et al. 2021. An Exploratory Study of Autopilot Software Bugs in Unmanned Aerial Vehicles. In *ESEC/FSE*.
[41] Irving Wladawsky-Berger. 2020. The Internet of Things Is Changing the World. https://www.wsj.com/articles/the-internet-of-things-is-changing-the-world-01578689806.
[42] Fan Zhang, Hasan Fleyeh, Xinru Wang, and Minghui Lu. 2019. Construction site accident analysis using text mining and natural language processing techniques. *Automation in Construction* 99 (March 2019), 238–248. https://doi.org/10.1016/j.autcon.2018.12.016
[43] Shenghua Zhou, S. Thomas Ng, Yifan Yang, et al. 2020. Delineating Infrastructure Failure Interdependencies and Associated Stakeholders through News Mining: The Case of Hong Kong's Water Pipe Bursts. *Journal of Management in Engineering* (2020).
[44] Wei Zhou, Chen Cao, Dongdong Huo, et al. 2021. Reviewing IoT Security via Logic Bugs in IoT Platforms and Systems. *IEEE Internet of Things Journal* (2021).