2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA) | 979-8-3503-4588-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SERA57763.2023.10197809

Obsolescence in Operating Systems and Microprocessors

Dheeraj Naraharisetti
Marriot's Ridge HS
Marriotsville, MD

Ramesh Karne
Computer and Information Sciences
Towson University
Towson, MD
rkarne@towson.edu

Joel Weymouth
Information Science and Systems
Morgan State University
Baltimore, MD

Alex Wijesinha
Computer and Information Sciences
Towson University Towson, MD
awijesinha@towson.edu

***Abstract*—Obsolescence and its impacts on software and systems continue to be of interest. Reducing obsolescence in operating systems and microprocessors will help to reduce software obsolescence. We examine obsolescence in Intel microprocessors and Windows operating systems. We first present data that illustrates the extent of the problem. We then consider extensible designs to reduce obsolescence in operating systems and microprocessors. This approach can be adapted to design software and hardware that are resilient to obsolescence.**

Keywords—software obsolescence, software evolution, operating systems, microprocessors, extensible designs

I. Introduction

Software evolution is an important part of the software life cycle and includes approaches for designing software that lasts and is easier to maintain. However, software and hardware products are often designed to have a limited lifetime. When this lifetime is reached, current software and hardware are replaced with new or upgraded versions. This cycle is a special case of obsolescence, which comes from the Latin term "obsolēscere," meaning to fall into disuse and decay. Although viewed as being an outcome of technological advancement and/or essential for business reasons [9], obsolescence results in a waste of hardware and software components. It also promotes frequent and unnecessary software and hardware releases. The environmental consequences of software and hardware obsolescence are an increased use of energy and resources, and e-waste due to the dumping of hardware, devices, and computer equipment. Reducing software and hardware obsolescence can help green computing efforts.

Software typically requires the support of an operating system or kernel to run, which in turn depends on the underlying hardware. Obsolescence in operating systems and microprocessors impacts software obsolescence. In this paper, we examine obsolescence in operating systems and microprocessors.

Operating systems include general purpose, centralized, distributed, real-time, embedded, mobile, and library operating systems. Their design and implementation are closely related to advancements in processor technology [4]. To limit the scope of the discussion, we only consider obsolescence in Microsoft desktop/laptop operating systems and Intel microprocessors. We first present data on the frequency of releases for these products, which shows that operating system and microprocessor obsolescence are closely related. We then discuss the use of extensible designs to reduce obsolescence.

The rest of the paper is organized as follows. Section II deals with obsolescence in operating systems and microprocessors and includes relevant data. Section III describes designs for microprocessors and operating systems to reduce obsolescence. Section IV discusses related work. Section V concludes the paper.

II. Obsolescence in Operating Systems and Microprocessors

Fig. 1 shows 16 editions of Windows 10, dominant among Microsoft operating systems for six years, and supported by 815 Intel processor models. A total of 12,337 Windows editions were designed from 2009 to 2021 including Starter, Home Basic, Home Premium, Professional, Enterprise, and Ultimate, and associated with Intel processors such as Celeron, Core, Pentium, Xeon, and Atom as shown in Fig. 2 (based on data from [5], [6]). Operating system size also increased, peaking at 55 million lines of code, as shown in Fig. 3 (based on data from [7]). To support new user requirements, more architectural features were added to CPUs, while technological improvements resulted in new CPU designs.

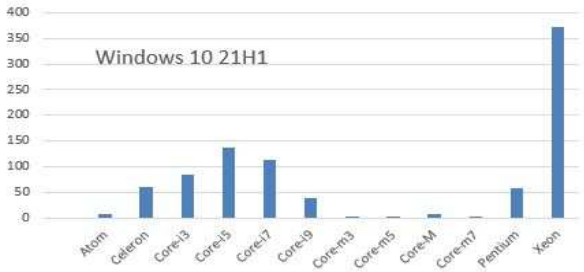


Fig. 1. Intel processors for Windows 10.

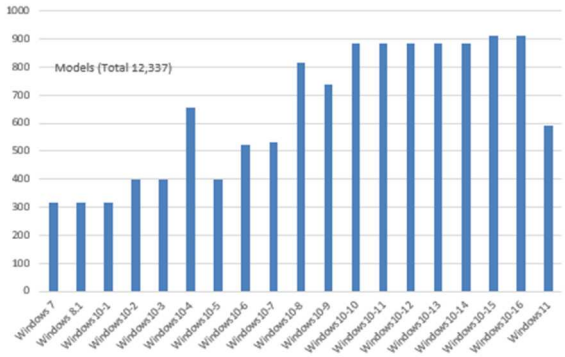


Fig. 2. Windows editions (2009-2021).

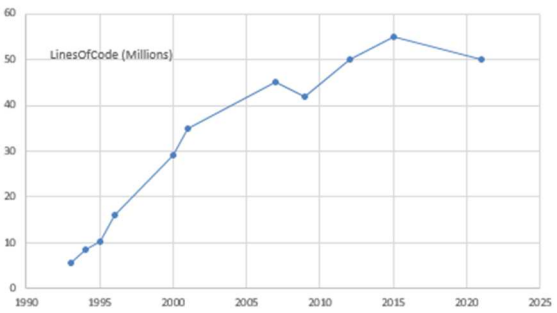


Fig. 3. Number of lines of code.

Obsolescence in Intel microprocessors used in desktops and laptops is evident in the data given in [1], [2]. Microprocessors are characterized by the number of transistors, process technology, and speed (frequency), while the number of bits (16, 32 or 64) defines their computer architecture and organization. As semiconductor chip technology improved, the number of transistors and clock rates increased, and performance improved. New chips were produced as soon as an improvement in technology became available, replacing the older chips within their computing environments and platforms. Every variant production required a separate chip, testing, maintenance, and replacement overhead, and a different programming environment to exploit its features. In turn, this impacted operating systems, compilers, applications, and other related environments and tools, and resulted in obsolescence of people skills related to hardware and software.

Due to the 1 MB memory limit on 16-bit processors (address 16+4=20 bits), 32-bit processors emerged. These processors provided ample address space (4 GB), virtual memory, and supported even more computer applications. Improvements in technology resulted in the advent of 64-bit processors. Table I shows the number of variants and chip types for 16, 32, and 64-bit Intel microprocessors. For 32 and 64-bit processors, variants outnumber chip types by almost a factor of ten. Over 43 years (1978-2021) a chip was released every 7.8 months, and one variant was released every 0.84 months.

TABLE I. MICROPROCESSOR RELEASES

No of Bits	Chip Types	No of Variants
16	5	11
32	27	274
64	34	330
Total	66	615
Avg. release in 43 years	7.8 months	0.84 month

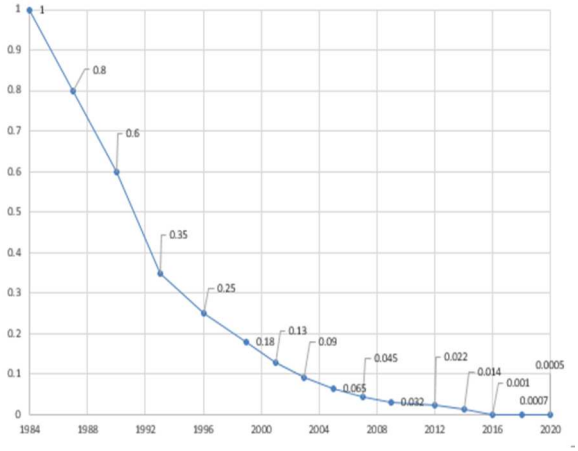


Fig. 4. Intel process technology trends.

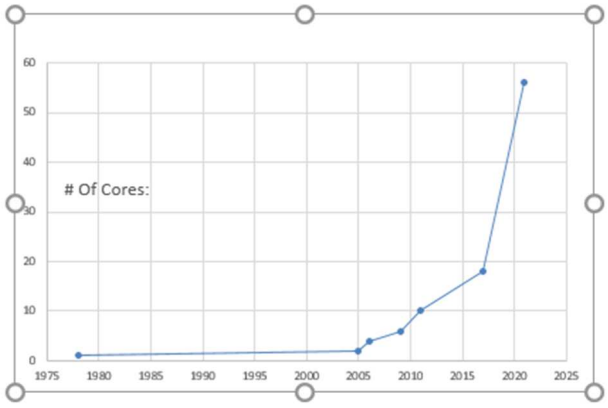


Fig. 5. Intel multicore trends.

Fig. 4 illustrates 64-bit Intel process technology trends over 36 years. Note that process technology shrunk from 1 to 0.5 nm (2000 times, which averages about 55 times in a year). Fig. 5 shows that over 16 years, the number of cores in the chip increased from 1 to 56, a trend that continues. Fig. 6 shows an improvement in clock speed from 0.8 to 5.33 GHz over 21 years. Fig. 7 shows that the number of transistors in 64-bit processors increased from 376 to 2270 million on a single chip.

As density increased, engineers added more functionality to the chip. More functionality was also part of the GPUs (general-purpose units for graphics, servers, and specialized functions), hyper-threads, pipelining, branch predictions, multiple cores, multilevel caches, varying cache sizes, variation of models, variation of clock frequencies, network protocols, security algorithms, customization for a given application, increased instructions (Fig. 8 based on data from [3]), specialized registers, specialized bus widths and I/O buses, and so on. These features were not envisioned in the original CPU architecture, so new designs were added as needed.

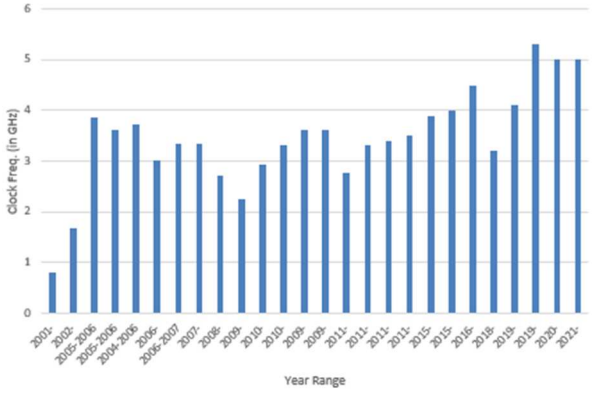


Fig. 6. Clock speeds.

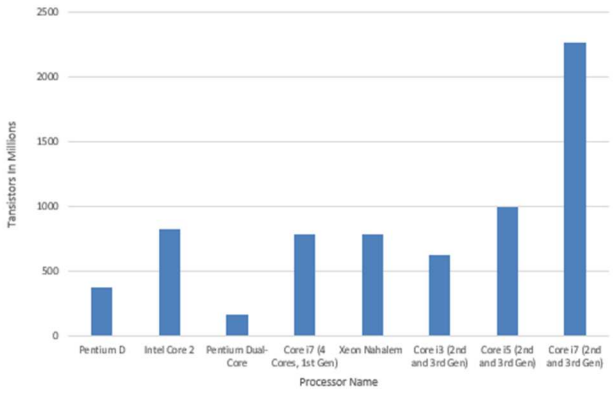


Fig. 7. Number of transistors.

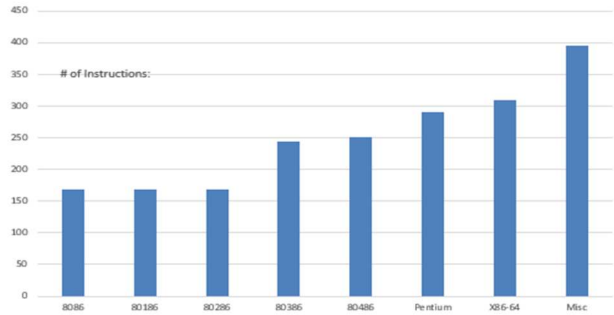


Fig. 8. Number of CPU instructions.

III. Design Strategies to Reduce Obsolescence

A. Reducing Microprocessor Obsolescence

The number of bits (16, 32, or 64), variants (clock frequencies, models, etc.), additional logic (cache, hyper-threads, system registers, etc.), the number of cores, instructions, and technology (brands) define six attributes characterizing Intel multiprocessors as shown in Fig. 9. The reduction of microprocessor obsolescence can be achieved with respect to four categories: variants, technology, additional functions, and bus widths. For example, chip designers produced eight variants of microprocessors from October 1999 to March 2000 (5 months), improving the clock frequency from 650 to 866 MHz. If microprocessor designs are extensible, a new variant will extend an earlier design and only one variant (866 MHz in this case) will be introduced only once the current technology is stable and a higher clock rate becomes feasible. This approach would have reduced microprocessor obsolescence. For customers with lower speed requirements, it would also have been possible to offer high-speed clock rates at the same price, thus reducing the number of variants.

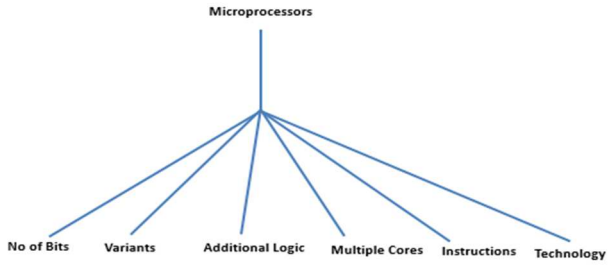


Fig. 9. Intel micrprocessor attributes.

Similarly, Intel released seven different Pentium microprocessor types from 1992 to 2000 or approximately one chip per year. Process technology improved from 0.8 to 0.25 microns. It could be argued that there was no need for so many releases. Pentium 2 and Celeron also overlap in their time of release. Waiting for at least a year would have reduced the number of chip releases and consequently reduced obsolescence. Furthermore, designing the Pentium processor model to be upward compatible would avoid the need to have other Pentium derivatives.

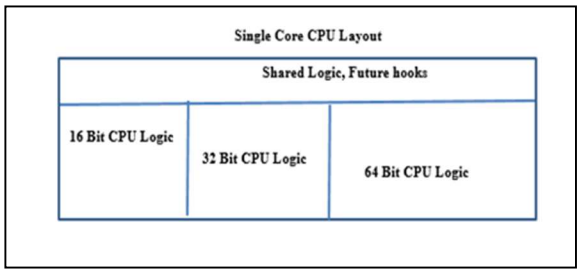
More logic (cache levels and varying sizes, hyper-threads, GPU etc.) can be architected a priori in the CPU and have hooks placed to enable them when released. Thus, a previous design is logically extended to support new functions. The instruction set in the original architecture should be robust enough to allow adding more instructions as needed. The CPU architecture should be designed with hooks enabling extension when older chips are replaced by newer chips. This approach is object-oriented and transparent to the user. It will reduce the number of chip releases and the corresponding software and maintenance costs, thus reducing obsolescence at many levels.

A simple example will illustrate the above concept. In the past, chip engineers located cache outside a CPU chip. As the CPU chip sizes grew, designers changed chip design to place a small L1 cache onto the chip. Eventually, chip designers

located L1, L2, and L3 caches on the chip. It may even be possible in future to push large amounts of main memory onto the chip and avoid a cache. Cache and main memory can be viewed as memory for the CPU. Putting a few hooks in the CPU, indicating where that memory location is, would solve the location problem and allow the extension of CPUs without obsoleting them. A similar concept can be used throughout CPU architectures and designs to reduce obsolescence.

Multicore processors have also increased complexity and caused rapid obsolescence in CPUs, chips, tools, supporting software, and applications. Multicore designs require memory to be shared and cause cache coherency problems. They also make load balancing, concurrency control mechanisms, and writing applications more complex. Instead of multiple cores, designers should consider adding other logic to microprocessors, providing upward compatibility, and preserving previous product versions. Furthermore, the bit sizes indicate the capacity of arithmetic and logic operations in a CPU. For example, in a 64-bit CPU, all operations are based on 64 bits. CPU architectures do not provide hooks to extend the bus width. In principle, CPU obsolescence can be reduced by adding hooks to the CPU for 16, 32, or 64-bit configurations as shown in Table II. Designing extensible chips for CPUs to enable enhanced functions and features will reduce obsolescence in manufacturing, production, and maintenance, and reduce complexity in supporting software for the chips.

TABLE II. PROPOSED SINGLE CORE LAYOUT



B. Reducing Operating System Obsolescence

Microsoft operating systems can also be characterized by six attributes as shown in Fig. 10: number of bits, editions, brands, additional applications, multiple cores, and additional features. Each release of the operating system impacts browsers, databases, editors, graphical interfaces, multi-core support, I/O, security, RAID, cache variations, system registers and special registers etc. A new OS release may be due to changes in computer architecture such as hyper-threading, multiple cores, or CPU bits. These upgrades increase the number of attributes and operating system complexity.

Design obsolescence and planned obsolescence are common with operating system releases. For example, Windows 11 replacing Windows 10 may be viewed as design or planned obsolescence [41]. While new releases address important security issues, they are likely to have a larger attack surface and be harder to secure due to increased complexity and larger code sizes [13]. Although not discussed here, such security issues are not limited to Microsoft operating systems.

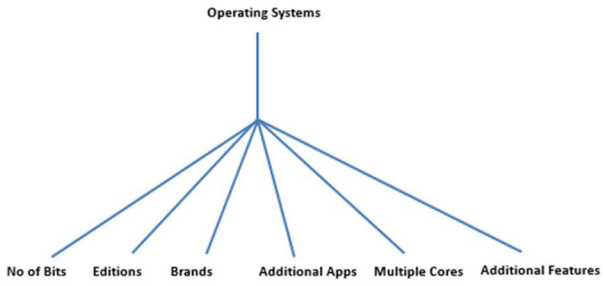


Fig. 10. Microsoft operating system attributes.

Reduction of operating system obsolescence is addressed with respect to five categories: processors, applications, additional features, direct interfaces, and user-mode functions. Adopting methodologies to extend the useful life of microprocessors as discussed above will reduce operating system obsolescence. The number of models will be considerably reduced, inducing savings in code size, production, maintenance, support, and human resources. Upward compatibility in operating systems will also help minimize obsolescence regardless of the size of the bit logic in Table II. Editors, presentation software, spreadsheets, tools, compilers, debuggers, user interfaces, networking protocol software, databases etc. are continually updated with new versions. Since these applications have many features in common, designers should create more abstractions in operating system facilities and design applications with generic drivers that target a variety of devices to reduce efforts and obsolescence.

Operating system designers should anticipate new features and place appropriate hooks in the original architecture. Operating systems can be logically extended by simply enabling a new feature and adding new code. This promotes software reuse and reduces obsolescence. Designers can also provide direct hardware interfaces to bypass the operating system. These interfaces can be implemented at the CPU level by writing system-level code to make these hardware interfaces available to application programmers thus reducing or even bypassing the operating system interface to applications. This approach will reduce operating system obsolescence while not requiring additional features to enhance the operating system.

As in Exokernel [11], some non-essential kernel functions can be moved to applications. Kernel functions are increasingly done in user space with current operating systems. An extreme approach is to run applications on bare machines [12] without any operating system or kernel. This approach can be facilitated in the future if vendors design hardware with direct interfaces for applications. Designing extensible operating systems with security in mind avoids the need for new operating system releases and patches based on security issues. More research is needed on how to integrate the design of microprocessors, operating systems, and software to achieve the goals of security and extensibility while reducing obsolescence.

IV. Related Work

Obsolescence and its impact on society, information technology and systems, software, hardware, business, manufacturing, engineering, education, and management have been (and continue to be) investigated. Mellal [23] provides an obsolescence typology and discusses the impact of obsolescence based on a literature review. Smit [25] used a triangulation process to study information obsolescence. Weerasuriya and Wijayanayake [26] study obsolescence in information systems and provide guidelines to manage it. Earlier studies such as Sandborn [27], Whelen [28], Gravier and Swartz [29], Bulow [30] and Legge [31] also deal with various aspects of obsolescence.

Bisschop, Hendlin, and Jaspers [14], and Malinauskaite and Erdem [15] discuss legal aspects of planned obsolescence. Green Matters [16] reports on French regulations that force industry to report a repair-ability score on many products. Hudomiet and Willis [20] examine the impacts of computerization on older workers in the labor market whose skills became obsolete.

Ma [21] defines a measure of technological obsolescence based on patent data and discusses results covering many aspects of product markets including innovation, growth, productivity, and profits. Vergara et al. [42] study the technological obsolescence of two virtual reality learning environments in engineering classes from 2013-2020 based on surveys using 135 students. Trabelsi et al. [22] discuss the use of machine learning and features selection for obsolescence forecasting. Parvin and Beruvides [24] review the literature on forecasting technology obsolescence. They also study abandonment of a technological innovation as an optimization problem [44].

Studies have also addressed software and hardware obsolescence. Jang et al. [32] investigate software and hardware degradation and propose strategies to prevent them. Bowlds, Fossaceca, and Iammartino [43] use multicriteria decision making for software obsolescence risk assessment. Gerasimou et al. [46] propose a method to address software library obsolescence. An opinion on Docker obsolescence is given in [10].

Zallio and Berry [17] conduct a literature review on design strategies that may help to address planned obsolescence in the IoT. Borning, Friedman, and Logler [18] consider the negative impacts of IT including device proliferation in the IoT and discuss material consumption, energy usage, and waste. Paul [19] notes some cases of obsolescence in the IoT. Bol et al. [33] target device obsolescence and design a reprogrammable microcontroller unit for the IoT targeting low power and long life. Abate and Violante [38] deal with obsolescence of digital components and propose an approach for customizing processor cores prior to FPGA implementation. A comprehensive study by Shalf [35] on the general future of computing and technology without Moore's Law makes the interesting observation that algorithm-driven hardware design could make the hardware obsolete.

Obsolescence management in aviation including system, software and hardware obsolescence is discussed in a detailed FAA study [34]. As part of their device security guidance initiative in the UK, NCSC [36] discuss secure configuration of platforms and reducing risks from obsolete devices and applications. The government of Victoria in Australia provides a guide that addresses risk from technology obsolescence [8]. Heagney and Walker [37] propose using application virtualization to address software obsolescence and security risks due to operating systems and chipsets. In a survey done by Mattord and Bandyopadhyay [39], a high percentage of respondents indicated that operating system obsolescence is not important in the development and operation of client/server and distributed systems. Nield [40] and Hand [45] discuss operating system obsolescence.

V. Conclusion

Obsolescence is a common problem in microprocessors and operating systems that impacts many areas of software engineering. The data on Intel microprocessors and Microsoft operating systems presented in this paper shows that there are frequent unnecessary releases resulting in obsolescence. Similar studies on obsolescence could be done using other microprocessors and operating systems. Extensible designs for microprocessors and operating systems will reduce their rate of obsolescence while making it easier to add new features and functions in the future. This approach is also applicable to other microprocessor technologies, CPUs, and operating systems. Thus, non-Intel chips, smartphones based on the ARM architecture and operating systems such as Linux and iOS could also be designed to be extensible as discussed above to reduce obsolescence. When microprocessors and operating systems have a long life, it will reduce technological obsolescence and extend the life of software that runs on these platforms.

The extensibility concept can be used in other areas of computing, and to design systems and system components. Hardware and software could also be designed with a view towards adding security and other features and functionality without continually releasing new hardware and software. Designs based on extensibility can be used in servers, routers, switches, IoT devices, application software, and systems software. More research is needed on approaches to allow for technological improvements while minimizing the impact of business and industry-driven strategies that create obsolescence and waste.

Future studies could investigate how obsolescence in operating systems and microprocessors impact software evolution and development. Such studies could also consider obsolescence in other operating systems and microprocessors: for example, in iOS and Apple silicon, and in Linux distributions.

References

[1] List of Intel Processors: https://en.wikipedia.org/wiki/List_of_Intel_processors#Core_i3_(2nd_and_3rd_Generation).

[2] List of Processors: https://en.wikipedia.org/wiki/List_of_Intel_processors#Pentium_III.

[3] Number of Instructions: https://en.wikipedia.org/wiki/X86_instruction_listings#Added_with_80186/80188.

[4] Windows Processor Requirements: https://docs.microsoft.com/en-us/windows-hardware/design/minimum/windows-processor-requirements

[5] https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_versions#Personal_computer_versions

[6] https://docs.microsoft.com/en-us/windows-hardware/design/minimum/supported/windows-10-21h1-supported-intel-processors

[7] Lines of code: https://www.quora.com/How-many-lines-of-code-does-Microsoft-Windows-have.

[8] https://www.vic.gov.aus, A guideline for managing risk from technology obsolescence, 2019.

[9] Built Not to Last: https://www.sierraclub.org/sierra/2021-4-fall/material-world/built-not-last-how-overcome-planned-obsolescence

[10] Why Docker is getting obsolete, https://www.aurigait.com/blog/why-docker-is-getting-obsolete/, 2022.

[11] D. R. Engler, The Exokernel Operating System Architecture, Ph.D. thesis, MIT, October 1998.

[12] R. K. Karne, Bare Machine Computing, Relevant Publications, http://orion.towson.edu/~karne/dosc/pubs.htm, accessed 12/10/22.

[13] D. Gens, OS-level Attacks and Defenses: from Software to Hardware-based Exploits, Thesis, Darmsstadt, Germany, Dec 2018.

[14] L. Bisschop, Y. Hendlin, and J. Jaspers, Designed to break: planned obsolescence as corporate environmental crime, Crime, Law and Social Change 78, 2022.

[15] J. Malinauskaite, F. B. Erdem, Planned Obsolescence in the Context of a Holistic Legal Sphere and the Circular Economy, Oxford Journal of Legal Studies, Vol. 41, Issue 3, Autumn 2021, 719–749.

[16] Planned Obsolence Exposed at Apple and Microsoft, in Light of New French Regulations, April 7, 2021, https://www.greenmatters.com/p/planned-obsolescence, Accessed 2-4-2022.

[17] M. Zallio and D. Berry, Design and Planned Obsolescence. Theories and Approaches for Designing Enabling Technologies., The Design Journal, 2017, sup1, S3749-S3761, DOI: 10.1080/14606925.2017.1352879.

[18] A. Borning, B. Friedman, and N. Logler, The 'Invisible' Materiality of Information Technology, Communications of the ACM, June 2020, Vol. 63 No. 6, 57-64, 10.1145/3360647

[19] F. Paul, IoT has an obsolescence problem, Network World, June 11, 2018.

[20] P. Hudomiet and R. J. Willis, Computerization, Obsolescence, and the Length of Working Life, NBER Working Paper No. w28701 Last revised: December 4, 2021, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3834141

[21] S. Ma, Technological Obsolescence, Nov 15, 2021, https://ssrn.com/abstract=3964128

[22]] I. Trabelsi, B. Zeddini, M. Zolghadri, M. Barkallah, and M. Haddar, Obsolescence Prediction based on Joint Feature Selection and Machine Learning Techniques, Proc. 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) – Vol. 2, 787-794, 10.5220/0010241407870794.

[23] M. A. Mellai, Obsolescence – A review of the literature, Technology in Society- Elsevier, 1-6, August 2020

[24] A. J. Parvin Jr. and M. G. Beruvides, Forecasting Technology Obsolescence: Assessing The Existing Literature, A Systematic Review, Proceedings of the American Society for Engineering Management, 2017.

[25] E. S. Smit, Obsolescence and its impact on reliability: further development of Internet triangulation, 9th IBA Bachelor Thesis Conference, July 5, 2017, University of Twente, Enschede, Netherlands.

[26] G. T. Weerasuriya and W. M. Wijayanayake, An Evaluation of Factors Affecting Information Systems Obsolescence, Journal of Emerging Trends in Computing and Information Sciences, Vol. 5, No. 3, March 2014 ISSN 2079-8407

[27] P. Sandborn, Designing for Technology Obsolescence Management, Proc. Industrial Engineering Research Conference, 2007, http://escml.umd.edu/Papers/Sandborn_IERC Paper_2007-revised.pdf

[28] K. Whelan, Computers, Obsolescence, and Productivity. The Review of Economics and Statistics, 84(3), 445–461, 2002.

[29] M. Gravier and S. Swartz, The dark side of innovation: Exploring obsolescence and supply chain evolution for sustainment-dominated systems. The Journal of High Technology Management Research, 20. 87-102. 2009, 10.1016/j.hitech.2009.09.001.

[30] J. Bulow, An Economic Theory of Planned Obsolescence. The Quarterly Journal of Economics, 101(4), 729–750. 1986, https://doi.org/10.2307/1884176

[31] K. Legge, Obsolescence Of People—In Context, Management Decision, Vol. 11 No. 1, 27-49. 2002, https://doi.org/10.1108/eb001008

[32] E. Jang et al., Unplanned Obsolescence: Hardware and Software After Collapse, Workshop on Computing Within Limits, June 2017.

[33] D. Bol et al., SleepRunner: A 28-nm FDSOI ULP Cortex-M0 MCU With ULL SRAM and UFBR PVT Compensation for 2.6–3.6-µW/DMIPS 40–80-MHz Active Mode and 131-nW/kB Fully Retentive Deep-Sleep Mode, IEEE Journal of Solid-Stae Circuits, 56:7, July 2021.

[34] Obsolescence and Life Cycle Management for Avionics, FAA, Nov 2015, https://www.tc.faa.gov/its/worldpac/techrpt/tc15-33.pdf

[35] J. Shalf, The Future of Computing beyond Moore's Law, Philosophical Transactions Of The Royal Society A, Jan 2020.

[36] Device Security Guidance, NCSC 2021, https://www.ncsc.gov.uk/collection/device-security-guidance

[37] C. P. Heagney and L. J. Walker, Virtual Applications Reduce Cyber Attack Surface for Test Program Sets and Station Software, IEEE AUTOTESTCON, Washington DC, 2018.

[38] F. Abate and M. Violante, Coping with Obsolescence of Processor Cores in Critical Applications, IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, 2008, 24-32, doi: 10.1109/DFT.2008.28.

[39] H. J. Mattord and T. Bandyopadhyay, The Impact of Operating System Obsolescence on the Life Cycle of Distributed Teams. 14 Americas Conference on Information Systems (AMCIS), 2008.

[40] D. Nield, What to do when your OS becomes obsolete—and how to save money in the process, Popular Science, Jan 14, 2020.

[41] B. Dipert, Microsoft embraces obsolescence by design with Windows 11, EDN, Sept. 7, 2021.

[42] D. Vergara, J. Extremera, M. P. Rubio, and L. P. Davila, The Technological Obsolescence of Virtual Reality Learning Environments, Applied Sciences, 10(3), 2020.

[43] T. F. Bowlds, J. M. Fossaceca, and R. Iammartino, Software obsolescence risk assessment approach using multicriteria decision-making, Systems Engineering, 21(5), 2018.

[44] A. J. Parvin Jr. and M. G. Beruvides, Optimizing the Abandonment of a Technological Innovation. Systems, 9 (2), 2021.

[45] A. Hand, Intention is at the root of overcoming OS Obsolescence, Automation World, May 24, 2018.

[46] S. Gerasimou et al., On software modernisation due to library obsolescence, ACM/IEEE 2nd International Workshop on API Usage and Evolution (WAPI), 2018.