Designing for Dystopia

Software Engineering Research for the Post-apocalypse

Titus Barik, Rahul Pandita, Justin Middleton, and Emerson Murphy-Hill
North Carolina State University, USA
{tbarik, rpandit, jamiddl2}@ncsu.edu, emerson@csc.ncsu.edu

ABSTRACT

Software engineering researchers have a tendency to be optimistic about the future. Though useful, optimism bias bolsters unrealistic expectations towards desirable outcomes. We argue that explicitly framing software engineering research through pessimistic futures, or dystopias, will mitigate optimism bias and engender more diverse and thought-provoking research directions. We demonstrate through three pop culture dystopias, *Battlestar Galactica*, *Fallout 3*, and *Children of Men*, how reflecting on dystopian scenarios provides research opportunities as well as implications, such as making research accessible to non-experts, that are relevant to our present.

CCS Concepts

•Human-centered computing → HCI theory, concepts and models; •Social and professional topics → Computing / technology policy;

Keywords

culture; design fiction; dystopia; ideation; post-apocalypse; software engineering

1. MOTIVATION

Literary theorists have long recognized the trade-offs in optimistic and pessimistic thinking through *utopias*, uncritical visions of an idealized future, and *dystopias*, scenarios which embody the fears and concerns originating from our own hesitancies regarding the future [3]. Unfortunately, research suggests that scientists are overwhelmingly optimistic, and subject to the effect of *optimism bias* — a robust and pervasive finding for peoples' tendencies to assign higher probabilities to desirable outcomes even in the face of contradictory evidence that challenges their beliefs [2].

For software engineering researchers, dystopias may serve as a means to mitigate optimism bias and to engender more diverse and thought-provoking directions in research. To advance our argument, we explore three dystopian scenarios from modern pop culture, and investigate the relevancy of software engineering research when designing for dystopia. We also demonstrate that, despite the fact that dystopias are dark fantasies that test the boundaries of reality [10], they nevertheless offer implications and insights that are relevant to our present. In other words, dystopian thinking "allows us to apprehend the present as history" [21].

Figure 1: Reluctant to rely on computers to assist in planning, Captain Adama and Executive Officer Tigh calculate and revise tactics offline using printed navigation charts and pencil-and-paper.

The rest of the paper is structured as follows: Section 2 adopts three dystopian scenarios from pop culture as a channel into potential software engineering research visions; Section 3 considers ways in which dystopian thinking gives us insights into the present; finally, Section 4 proposes the use of *design fiction* as a means to further explore dystopia.

2. DESIGN SCENARIOS

Using *scenario paradigms* [18], a literary technique used to construct alternative futures to facilitate reasoning, we investigate three dystopian design scenarios from pop culture media. For each scenario, we describe the story and setting of the selected dystopia, contextualize the role of software engineering, and use the scenario as a source of inspiration through which we postulate research directions in software engineering.

2.1 Battlestar Galactica

Story and setting. Heavily informed by the events of 9/11, Moore and Eick's *Battlestar Galactica* television series

reboot depicts an "end of the world" scenario: the *Second Cylon War*, in which robots, called Cylons, return to mercilessly attack the human colonies, nearly eradicating the human race. The few remaining survivors flee into space with faster-than-light (FTL) *jump* technology as a last resort. The central story follows one of the spaceships, the *Battlestar Galactica*, and the crews' search for a new home. However, the officers on *Galactica* suspect that there are Cylon cyber-terrorists on the ship masquerading as humans.

Battlestar Galactica presents a dystopia in which the crew has a deep distrust in technology, as a result of fears that technology could be hacked by the Cylons. This hostility towards technology is exemplified in an exchange in which the Secretary of Education Laura Roslin asks Commander William Adama about installing a networked computer system to promote better teaching. Adama declines and says that "many good men and women lost their lives aboard this ship because someone wanted to make a faster computer to make life easier" [7]. Despite the additional effort, designated crew members are often seen performing complex calculations with pencil and paper, limiting interactions with the computer systems only when necessary to control the spaceship (Figure 1).

Role of software engineering. The justified skepticism of technology presents perplexing design constraints for software engineering researchers. How can toolsmiths support developers when developers are reluctant to use computers to develop their programs in the first place? And what measures can the crew take to increase their confidence that their systems have not been tampered with?

Designing for dystopia. In this dystopian scenario, we are reminded of a mainframe era up to the mid-1980s where developers predominately wrote software without the use of a computer. In this era, developers fed decks of physical punch cards to the computers to run programs. To facilitate the editing of these programs, developers could handwrite programs on *coding sheets*, a special form that would be meticulously converted to machine-readable cards by trained human operators.

One dystopian advantage of this approach is the inherent security properties that come with a physical deck: even if Cylons were to hack into the main computers and destroy software, no virtual computer attack could ever alter a physical card. In resurrecting a long-abandoned line of research, however, we ask: could a researcher design a "system" for developers to write software more naturally using pencil-and-paper? One intriguing possibility may be to remix old technology with new. We may, for example, deploy a read-only interpreter on ROM. This interpreter could use optical character recognition to eliminate the need for the dedicated human operator; instead, it could read the coding sheets directly. In this way, we could maintain the physical security properties of cards, while simultaneously offloading computation to the read-only software interpreter. Unfortunately, as punch card technology became obsolete, so too did efforts to pursue this line of research [5].

The benefits of a 'pencil-and-paper IDE' apply not just to dystopia but might have applications even today. As one example, having a pencil-and-paper IDE can enable developers to write algorithms, even when they are in situations where consistent access to computers is not available. Importantly, some developers even postulate that there are cognitive benefits to writing code away from the computer [14].

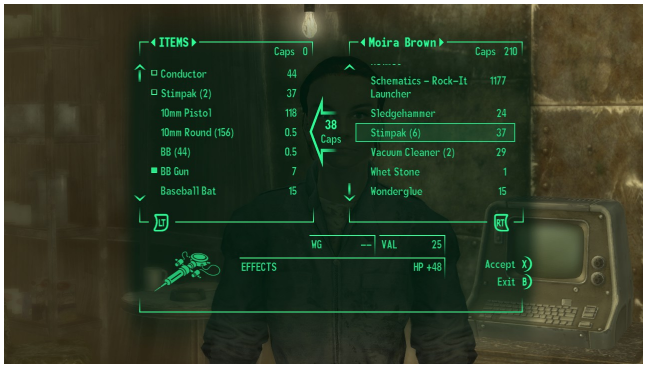

Figure 2: The people in *Fallout* rely on a bartering economy as a fundamental means of exchanging goods and services.

2.2 Fallout 3

Story and setting. *Fallout* is an open-ended, retro-futuristic role-playing video game that takes place generations after a global collapse that has resulted from resource scarcity. This conflict over the limited resources culminated to a nuclear holocaust that horrifically transformed the world into a hostile wasteland and destroyed most of the human race [13]. During the holocaust, a few surviving groups fled to underground vaults, where they remained in isolation and ignorance of the outside world for generations before finally returning to the surface.

Role of software engineering. In the absence of any reliable economic and technological infrastructure, we envision that the sharing of data and software, like other goods in *Fallout*, are limited to in-person bartering (Figure 2). Acquiring new programs and patching the software in this dystopia is both risky and costly, because obtaining such data requires a person to travel under the potential threats of numerous hostile enemies.

Designing for dystopia. Today, we take for granted the availability of reliable networks that make the transmission of software essentially free. For example, we routinely take advantage of third-party libraries from remote package sources and use them in our own programs. But such convenience has not always been the case. In the pre-Internet era, software developers might use bulletin board systems, or BBSes, to send and receive pieces of software through *software exchanges* [6]. Developers would use dial-up modems over analog telephone lines, often incurring significant long-distance costs and sometimes having to re-transmit data because of unreliable communication channels. Some BBSes only had a limited number of lines, supporting 2-4 simultaneous users. Because of this scarcity, BBSes instituted tit-for-tat *quotas*, in which developers were first required to contribute a piece of useful software before downloading software within the repository. And because patches often took months to propagate through these BBS networks (if they appeared at all), local developers would instead trade their own "unofficial" patches to third-party software.

The dystopian perspective of Fallout is reminiscent of the scarcity of the BBS era: although the needed software or library theoretically exists, the cost of obtaining the software is steep. Rather than having a central authority for software libraries, developers instead must exchange software through

Figure 3: In *Children of Men*, a worker from an aging demographic mourns the death of the world's youngest person: 18 years old.

agoric, peer-to-peer markets [20]. Software engineering researchers must explore avenues to support and encode an economics of computation, such that the costs of software *functionality* are somehow propagated alongside the functionality itself. Algorithms could use perform cost-benefit trade-offs on libraries and aid developers in deciding whether to use a library, or to write the functionality from scratch.

History reminds us that periods of scarcity and excess occur in cycles. The period of high-speed Internet was followed by challenges in mobile computing, where like the BBS era, disconnections are frequent and power constraints restrict bandwidth [9]. Designing with dystopia in mind prepares us for the next inevitable cycle of scarcity.

2.3 Children of Men

Story and setting. Alfonso Cuarón's film, *Children of Men*, presents a future in which Cuarón asks: "if there were no future, how would we behave?" [1]. The story is framed within a childless Great Britain, where nearly two decades of a global infertility crisis have led to societal collapse — the youngest known human on the planet is 18 years old. To survive, Great Britain has transformed into a militarized police state in which citizens are required to submit to mandatory fertility testing; government-distributed suicide kits are advertised on television for those who have lost hope. As British propaganda tells the viewers: "the world has collapsed; only Britain soldiers on." Having surrendered to the possibility of finding a cure to infertility, the government's aim is above all for the security, comfort, and pleasure of its citizens [15].

The technology of this dystopia is not unlike that of the present: as a result of the authoritarian actions of the government, Great Britain, unlike the rest of the world, has managed to maintain most of its infrastructure. Case in point, citizens have access to technology such as televisions and computers, as well as access to public transport, such as the London Underground. However, with the end of times within reach, societal and technological progress has essentially halted.

Role of software engineering. *Children of Men* reflects a demographic shift in the types of people who are software engineers as well as a shift to primarily support the maintenance existing software, rather than the development of new software for end-users. Consequently, the skills needs to be precisely allocated to fit the software maintenance needs of the population, and balanced against other necessary skills.

Designing for dystopia. The authoritarian dystopia of *Children of Men* allows us to manipulate the scope of software engineering education when the government can explicitly tune the workforce. At one extreme, which we might consider the utopian ideal, every individual can dabble with programming, as implied by government initiatives such as CS for All.[1] Thus, as the population increasingly shifts to higher age brackets, the government would need citizens to rapidly develop basic programming competency. In such an environment, software engineering researchers must consider how to quickly teach software engineering skills should the need arise, in ways beyond that of a slow, traditional, four-year education. For example, job training may even need to be on-the-fly, in which systems like *Tutorons* can provide on-demand help to teach the citizen the task that needs to be performed as they work on it [12].

But simply adding more programmers doesn't necessarily imply better programs. Harrison argues, for example, that "it's simply unfathomable that we could expect security from the vast majority of software applications" when software is written by undisciplined programmers [11]. At this other extreme — a perspective that is consistent with human-capital theory of economics [16] — the government would allocate professional programming training only to those most capable of performing the task. Essentially, CS for the Select Few.

In the utopian ideal, it is possible that universal coding skills would increase the overall welfare of society. However, it is also possible that the supply of available coders would far outweigh the demand for this skill set, resulting in lower wages or increased unemployment. Perhaps this is why professional organizations in other disciplines, like the American Medical Association and American Bar Association, employ uneasy measures such as licensing, to artificially limit the number of individuals who may practice these disciplines [25, 23]. Both of these organizations argue that enforced caps are necessary in order to maintain high standards.

3. IMPLICATIONS

We examine how future dystopian scenarios can be used to help us think about the present, beyond the scenarios discussed in this paper.

Dystopian thinking forces us to confront difficult potential futures. Dystopian thinking is not simply pessimistic thinking. If we think of a potential future as a single point in an infinite event space, dystopian thinking challenges us to confront our own work with "distant" event spaces that may be undesirable for software engineering researchers, but ones in which we may be forced into nevertheless. For example, teaching programming to everyone to address social injustice is no doubt a noble goal in computer science education, but our unchecked enthusiasm for it may blind us to the economic realities of supply and demand that govern the job market, as in the hypothetical situation posed through *Children of Men*.

Dystopian thinking benefits us even if the dystopia doesn't materialize. As our design scenarios have shown, dystopian thinking can provide an orthogonal lens, a creative playspace of sorts, through which we explore what might otherwise be controversial ethical or technological

[1]https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

issues. Many dystopian narratives are in fact critical reflections on present societal issues, framing these issues in terms of fictional dystopian scenarios rather than non-fiction [21]. This recasting can foster discussion in research and lead to more imaginative solutions. For example, Munz and colleagues have used *zombies*, that is, reanimated human corpses that feed on living human flesh, to mathematically model the important problem of understanding infectious disease propagation [22]. And Prottas uses the trope of *vampires* to highlight how organizations unintentionally discriminate against individuals with disabilities [24].

Dystopian scenarios make research accessible to non-experts. Pop culture is not strictly for entertainment; this medium can also serve to inform the public about complex issues and make them accessible to non-experts [8]. Dystopian scenarios in particular can be used to illuminate specific problems and highlight the importance of software engineering research in addressing them. For example, the essential ideas of secure software design can be illustrated using narratives like *Battlestar Galactica*, without the need for technical jargon. Dystopias may also serve to inspire others to contribute to software engineering research, analogous to how video games have been used in software engineering to spark interest in programming [4]. Although researchers are inspired at times by dystopian scenarios for ideas, intentionally situating our work within a dystopian frame is also useful for demonstrating the relevancy of our work to others [19].

4. CONCLUSION

We have argued for the utility of thinking in terms of dystopias for inspiring visionary research directions and mitigating optimism bias. Other communities, such as human-computer interaction, have formalized this style of fantasy thinking through *design fiction*, that is, the borrowing or constructing of narrative elements to understand possible future designs. For example, the HCI community offers "alternate endings" workshops as a means to circumvent visions of the future that would otherwise be described as "typically utility-driven and focus[ing] on the short term" [17]. Our community might likewise benefit from exploiting the synergy between software engineering research and dystopian ideas to accomplish the same.

5. ACKNOWLEDGMENTS

This material is based upon work in part supported by the National Science Foundation under Grant No. 1252995.

6. REFERENCES

[1] S. Amago. Ethics, aesthetics, and the future in Alfonso Cuarón's Children of Men. *Discourse*, 32(2):212–235, 2010.

[2] D. A. Armor and S. E. Taylor. When predictions fail: The dilemma of unrealistic optimism. In *Heuristics and Biases*, pages 334–347. Cambridge University Press, 2002.

[3] R. Baccolini and T. Moylan, editors. *Dark Horizons: Science Fiction and the Dysopian Imagination*. Taylor & Francis Group, 2003.

[4] T. Barik, M. Everett, R. E. Cardona-Rivera, D. L. Roberts, and E. F. Gehringer. A community college blended learning classroom experience through artificial intelligence in games. In *Frontiers in Education*, pages 1525–1531, 2013.

[5] R. Bornat and J. Brady. Using knowledge in the computer interpretation of handwritten FORTRAN coding sheets. *International Journal of Man-Machine Studies*, 8(1):13–27, 1976.

[6] P. R. Dewey. *Essential Guide to Bulletin Board Systems*. Meckler Publishing, 1987.

[7] J. T. Eberl. *Battlestar Galactica and Philosophy: Knowledge Here Begins Out There*. Blackwell Publishing Ltd, 2008.

[8] J. Fiske. *Understanding Popular Culture*. Routledge, 2010.

[9] G. Forman and J. Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, Apr. 1994.

[10] M. D. Gordon, H. Tilley, and G. Prakash, editors. *Utopia/Dystopia: Conditions of Historical Possibility*. Princeton University Press, 2010.

[11] W. Harrison. From the editor: The dangers of end-user programming. *IEEE Software*, 21:5–7, 2004.

[12] A. Head, C. Appachu, M. A. Hearst, and B. Hartmann. Tutorons: Generating context-relevant, on-demand explanations and demonstrations of online code. In *VL/HCC '15*, pages 3–12, Oct. 2015.

[13] D. Hodgson. *Fallout 3: Prima Official Game Guide*. Prima Games, 2008.

[14] A. Hunt. *Pragmatic Thinking and Learning: Refactor Your Wetware*. Pragmatic bookshelf, 2008.

[15] P. D. James. *The Children of Men*. Vintage, 2010.

[16] D. P. Lepak and S. A. Snell. The human resource architecture: Toward a theory of human capital allocation and development. *Academy of management review*, 24(1):31–48, 1999.

[17] C. Linehan, B. J. Kirman, S. Reeves, M. A. Blythe, J. G. Tanenbaum, A. Desjardins, and R. Wakkary. Alternate endings: Using fiction to explore design futures. In *CHI EA '14*, pages 45–48, Apr. 2014.

[18] M. Mannermaa. In search of an evolutionary paradigm for futures research. *Futures*, 23(4):349–372, May 1991.

[19] B. C. Milburn. Modifiable futures: Science fiction at the bench. *Isis*, 101(3):560–569, 2010.

[20] M. S. Miller and K. E. Drexler. Markets and computation: Agoric open systems. *The Ecology of Computation*, 1, 1988.

[21] T. Moylan. *Scraps of the Untainted Sky: Science Fiction, Utopia, Dystopia*. Westview Press, 2000.

[22] P. Munz, I. Hudea, J. Imad, and R. J. Smith. When zombies attack!: Mathematical modelling of an outbreak of zombie infection. *Infectious Disease Modelling Research Progress*, 4:133–150, 2009.

[23] M. Noether. The effect of government policy changes on the supply of physicians: Expansion of a competitive fringe. *The Journal of Law & Economics*, 29(2):231–262, 1986.

[24] D. J. Prottas. The vampire in the next cubicle: The Americans with Disabilities Act and the undead. *Employee Responsibilities and Rights Journal*, 24(1):79–89, Jan. 2012.

[25] S. Rosen. The market for lawyers. *The Journal of Law & Economics*, 35(2):215–246, 1992.