Growing Pains for Space Shuttles?

"RELEASE EARLY, RELEASE OFTEN" AS AN ALTERNATIVE DEVELOPMENT PROCESS

WRITTEN BY SEBASTIAN DZIALLAS

Most technology nowadays is developed using a top-down approach that has been used for decades, with design decisions coming from top management and engineers left with little leeway regarding the execution of orders they do not necessarily deem fitting. Recent technological disasters like the Apollo 13 incident have demonstrated the limits of the current paradigm in managing the development of complex technology. This doesn't have to be the case. There are alternatives: Thomas Kuhn describes emerging development paradigms as different mindsets, which lead to higher quality end-products.[1] One of them, the open source way, differs drastically from the current paradigm; design decisions are made by the engineers implementing them, within a public environment, and using a variety of communication processes chosen on-the-fly. A paradigm shift towards the open source way could have averted Apollo 13 -- and more.

On April 11, 1970, 55 hours and 53 minutes into the mission, the crew of the spaceship Apollo 13 noticed what they described as a "loud bang". After coordinating with ground control, it turned out that Apollo 13 had just lost two oxygen tanks and fuel cells.[2] The incident caused multi-million dollar damage and almost cost the lives of the three crew members. And while Apollo 13 is still one of the most famous space disasters, most people don't know that the first signs of upcoming trouble arose about a month before the launch.[3] *"[…] The countdown demonstration test (CDDT) revealed that the thermal switches were overloaded [which] caused very high temperatures (700° to 1000° F) inside the heater tubes."*[4] However, the engineer performing this routine test did not notice the crucial warning signs: The temperature sensor he was using was limited to display only temperatures below 80°F. Effective quality assurance

[1] Rob Martello, *Thomas Kuhn's Scientific Revolutions (a summary by Rob Martello)* (Needham, Olin College).
[2] Flight Control Division, *Mission Operations Report: Apollo 13* (Houston, NASA, 1970), p. II-1.
[3] Chiles, *Inviting Disaster*, p. 181.
[4] Apollo 13 Review Board, *Report of Apollo 13 Review Board* (Washington D.C., NASA, 1970), p. F-3.

methods would have identified such a significant limitation of the equipment and given ample time to react to the damages. This would have been only a small incident if left by itself; combined with other issues, the top-down approach fueled the disaster. When the spaceship's second oxygen tank was accidentally dropped at Beech Aircraft during a maneuver to lift the tank out of its service module, the resulting damage permitted the thermal switches to overload.[5] But when the test results were raised during the final flight review, NASA was simply not aware of the damage. Had there been proper communication structures involving all contributors in place, this incident would have almost certainly been escalated.

Today, the predominant development paradigm is a direct descendent of the top-down approach used during the development of Apollo 13. Improvements have been made: independent code review is now being used as a crackstopper to prevent issues from accumulating and causing a significant malfunction.[6] Literature on developing secure code recommends outsourcing code analysis to external consulting companies whose highly-trained professionals conduct reviews in order to make the code more secure.[7] Within this paradigm, it is imperative to limit the access to the source code in such an environment. Disclosing security information is considered a danger. Making an entire codebase accessible to any member of the public would be inconceivable. These restrictive policies are contrasted by the concept of peer review, a vital part of the increasingly successful open source way. An example of a peer review process has been implemented by the Sugar Labs community producing the desktop environment used on One Laptop Per Child computers. Each change in the codebase goes through a series of five phases, beginning with a preliminary discussion that involves all concerned teams on a public mailing list. The proposer then submits a formal proposal with a detailed description of the suggested implementation, as well as patch that provides the required changes to the code base. These materials are sent again to a public mailing list, where discussion around the proposal takes place until concerns are resolved and the code is considered ready. If this is the case, the patch will be signed off again and committed – it has

[5] Chiles, *Inviting Disaster*, p. 188.
[6] Chiles, *Inviting Disaster*, p. 184.
[7] Michael Howard, David LeBlanc, *Writing Secure Code* (Redmond, Microsoft Press, 2003), p. 45.

been accepted.[8] This means that by the time any given line of code has entered production, it has been checked and signed off publically at least four separate times. Open source projects rely heavily on this concept of radical transparency. Eric S. Raymond describes a fairly practical reasoning for the phenomenon in his essay titled “The Cathedral and The Bazaar” as “Linus’s Law”: *“Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.”*[9] For large codebases, smaller teams face problems maintaining iterations at the same high quality. Raymond raises the factor scalability: Adding more people to the problem will identify issues sooner. However, this concept inevitably clashes with the top-down development paradigm. Developers who had no part in the decision process for the implementations they are working on are less motivated to contribute quality code. Intrinsic motivation is a critical piece in the open source way. As former Red Hat Community Architect Greg DeKoenigsberg writes about contributing to a project*: “It must, however, always be rewarding. Because if it’s not, people won’t do it anymore.”*[10] Had there been a motivation for Beech Aircraft to properly report and escalate the oxygen tank drop initially, NASA would have been able to make informed choices during their final flight review. Time pressure and financial concerns both provide entirely different motivations.

While Linus’s Law explains why open source communities operate the way they do, it does not focus on how they operate: using the “open source way.”[11] In short: It is a methodology that embraces free and open exchange and participation through communication within a community in which successful work leads to successful projects.[12] In the open source paradigm, communication is the key that permits the entire community to access all content. Raymond compares in a separate part of his essay the differences between top-down and bottom-up development approaches. He utilizes the metaphor of a cathedral to describe the approach that gives third-party developers only access to subsequent software releases, but restricts access to in-development versions. On the other hand, the bazaar represents an

[8] Michael Howard, David LeBlanc, *Writing Secure Code* (Redmond, Microsoft Press, 2003), p. 45.
[9] Eric S. Raymond, *The cathedral and the bazaar: musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, O’Reilly Media Inc., 2001).
[10] Material From: http://gregdekspeaks.wordpress.com/2010/03/01/working-on-fedora-fun/
[11] Red Hat Community Architecture Team, *The Open Source Way* (Raleigh, Red Hat, 2009).
[12] Material From: http://opensource.com/open-source-way

approach that requires all development to happen online and in public.[13] Raymond's essay is one of the most popular readings related to open source – it portrays the differences between these so fundamentally different approaches. NASA employed a top-down development approach for decades: similarities to the incidents of Apollo 13 can even be found later in the Challenger Spaceshuttle disaster, where a contractor's Vice President overruled the engineers' recommendation because of time pressure. Proponents of the top-down approach have argued that the open source way would not be applicable to a larger set of problems because communities participation may vary, rendering the open source way unusable. Red Hat Inc.[14] , a Standard and Poor's 500 company[15], releases information concerning their software updates publicly and does not exclude security-relevant changes from this procedure. Quoting their security measurements website, it is noticeable that they *"provide reports and metrics, but more importantly, […] also provide the raw data so customers and researchers can produce their own metrics, for their own unique situations, and hold [Red Hat] accountable."*[16] This information permits third parties to analyze Red Hat's response time to disclosed threats.[17] It is, in fact, part of Red Hat's business model, permitting third parties to access, contribute and particularly leaverage their information, leading to a better final product. Several international large scale projects go one step further in terms of utilizing open source methodologies. Former Fedora Project[18] Leader Paul W. Frields writes in an article: *"One of the fundamental principles I think our community expects […] is that we default to open wherever possible. […] And by and large, we really do."*[19] While Frields leaves the definition of "whenever possible" open, community projects take this responsibility seriously. The Wikimedia Foundation behind the well-known encyclopedia Wikipedia releases their financial reports publicly[20] and the Fedora Project is in the process of opening its financing process.[21] In fact, the openness is what enables

[13] Eric S. Raymond, *The cathedral and the bazaar: musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, O'Reilly Media Inc., 2001).
[14] Disclaimer: Starting this May, the author will be employed by Red Hat Inc.
[15] Material From: http://press.redhat.com/2009/07/27/red-hat-included-in-sampp-500-index/
[16] Material From: https://www.redhat.com/security/data/metrics/
[17] Material From: http://www.awe.com/mark/blog/tags/metrics
[18] Disclaimer: Fedora is a Red Hat sponsored project. The author is a contributor and member of the community.
[19] Material From: http://paul.frields.org/2011/03/01/defaulting-to-open/
[20] Material From: http://wikimediafoundation.org/wiki/Financial_reports
[21] Material From: http://fedoraproject.org/wiki/Finance_SIG

contributors located in different continents to work together to maintain large codebases and yet release their product in short release cycles of e.g. every six months. This demonstrates effectively that open source practices such as radical transparency and peer review do not hinder development, but in fact rather encourage it.

Most of this work is happening outside the currently existing paradigm of top-down development. A predominant argument against the bazaar notes that releasing code or content to the public might impose a security risk. And while this argument certainly holds-true in some mission-critical cases that cannot be exposed to the public, it is worth noting that the National Security Agency is releasing their Security-Enhanced Linux project under the common GNU Public License, stating their goal to *"use [their] resources as efficiently as possible to give NSA's customers the best possible security options […]. The objective […] is to develop technologic advances that can be shared with the software development community […]."*[22] While the NSA also clearly states that it does not favor a certain business model over another, it argues that it favors enhanced security. The paragraph provides a couple of interesting insights. First of all, the National Security Agency highlights the interaction with the software community, raising the importance of collaboration – no matter whether between single contributors, companies or government agencies – and the accessibility of the technological advancement to everybody. However, and most importantly, the NSA effectively argues that employing means of the open source methodology in their development permits them to work not only more effectively, but also creates a better option for their target audience.

It has become apparent how the incidents of Apollo 13 could have been averted by applying open source best practices. In fact, this is neither limited to the realm of aerospace, nor the one of open source software development. Following Kuhn's theory of paradigms, non-top-down project management anomalies appear, as more and more groups such as the NSA choose to utilize open source practices like radical transparency, public code review, and a focus on communication between decentralized collaborators. If NASA choses to continue their spaceshuttle program, applying open source techniques could permit them to reach shorter

[22] Material From: http://www.nsa.gov/research/selinux/faqs.shtml#I23

release cycles than the current paradigm -- a goal that NASA was never able to reach up until now. All points considered, it seems like there are major movements, not just within the software industry, that indicate that we are heading for a paradigm shift in the future.