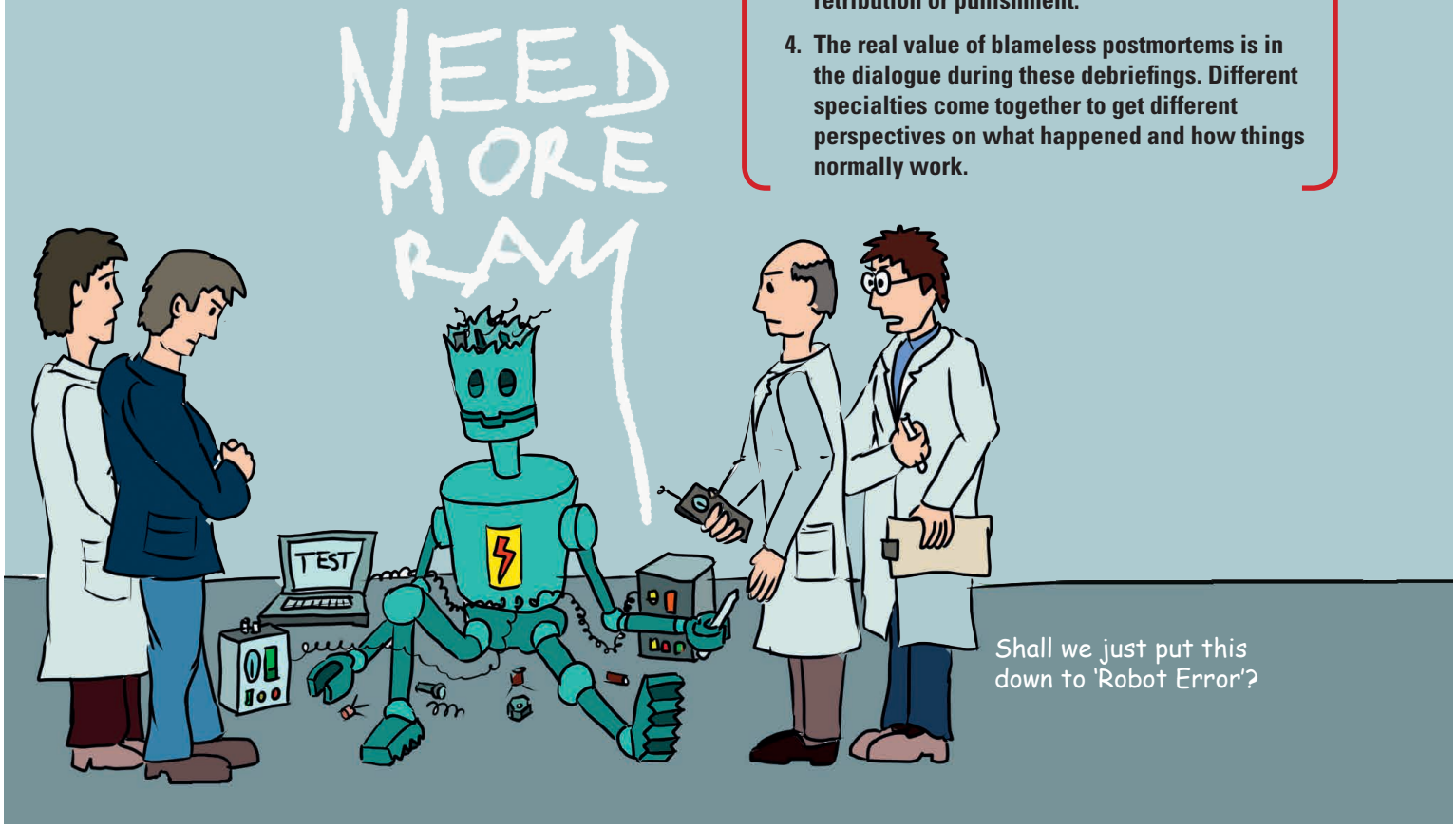
COLLABORATION IN POST-INCIDENT REVIEW

Online software is a fast-growing field that many industries, including aviation, depend on. It is a complex domain that crosses geographic and geopolitical boundaries and depends on multidisciplinary collaboration. For a fairly new industry, it has been innovative in introducing a collaborative form of learning from incidents, often called 'blameless postmortems', which we could learn from. **John Allspaw**, who has been critical to this, outlines the field and the approach to post-incident review.

John Allspaw has worked in software systems engineering for over twenty years in many different domains: government, online media, social networking, and e-commerce. John's publications include the books The Art of Capacity Planning (2009), Web Operations (2010) and a chapter in Human Factors and Ergonomics in Practice on HF practice in Web Operations. John holds an MSc in Human Factors and Systems Safety from Lund University. He is currently co-building Adaptive Capacity Labs, LLC.

KEY POINTS

1. **Outages or degraded performance in online software can have enormous impact, costing millions or even tens of millions in total lost revenue. Learning from such incidents is critical.**
2. **When there is an incident, a post-incident review (sometimes called a 'postmortem') is held. This is usually a semi-structured facilitated group debriefing.**
3. **Postmortems are 'blameless' to understand work-as-done (as opposed to work-as-imagined, by other colleagues or management) without fear of retribution or punishment.**
4. **The real value of blameless postmortems is in the dialogue during these debriefings. Different specialties come together to get different perspectives on what happened and how things normally work.**



The most important industry you've never heard of

Most people have probably never heard of 'web engineering and operations', and yet rely on it for many aspects of their lives and work, including controllers and pilots. There are some surprising similarities between this field and aviation. Both worlds involve many specialised professions, with individuals and teams working in high-tempo and competitive markets. There are many technical, cultural, and organisational challenges. People have to cope with complexity and time pressure on a daily basis. The displays and controls have similar challenges to those of people in safety-critical sectors such as the provision of air traffic services. And all aspects of the work are steeped in automation; except far more so.

The software and infrastructure delivering the software may need to support hundreds of thousands of users at any given time. There is huge interconnectedness between websites, applications and other network-connected services, which are often independently designed, owned, and operated. An e-commerce website, for example, may rely on external services whose functionality, availability, and performance that are not within its control. The software itself is built of many components, some of which are standardised and in the public domain, some of which are proprietary.

When incidents occur, it is often difficult for engineers to understand breakdowns and faults, and there are many opportunities to make a bad situation worse

Dealing with incidents in web operations and engineering

Just as incidents occur in aviation, outages or degraded performance occurs in web operations and engineering, sometimes with enormous impact. The routers and switches that make the global internet work fail often. Servers that contain content for websites and other increasingly critical services (official government statements and policies, payments processing, bank transfers, electronic medical records, etc.) go 'down' for various reasons (hardware, software), and more frequently than most of the public realise. These incidents affect business continuity at a cost of millions of dollars, and can have unintended consequences that spread to non-web domains, such as the loss availability of electronic medical records.

When incidents occur, it is often difficult for engineers to understand breakdowns and faults, and there are many opportunities to make a bad situation (e.g., an outage) worse (e.g., by corrupting data permanently). It is also difficult to understand and learn from outages and other events after they have happened.

'Blameless postmortems'

Typically in software-centred companies (like Facebook, Amazon, etc.) when there is an incident such as an outage, degradation, slow performance, or other significant surprising event, a post-incident review (sometimes called a 'postmortem') is held. In such cases, usually no single individual (or even a team) can fully understand what is happening, and there is no 'bird's eye view' of how it all works. Engineers specialise in doing things like building new features, fixing bugs, responding to outages, and maintaining all the technology that the business relies upon. So engineers must rely on the perspectives that others have on an issue to build a picture of what has happened, what is happening now, and what needs to be done. Collaboration is essential for normal operations and dealing with unwanted events.

Engineers must rely on the perspectives that others have on an issue to build a picture of what has happened, what is happening now, and what needs to be done

In 2012, I wrote a post for the engineering blog for my company, Etsy, Inc. (an e-commerce marketplace) called *Blameless PostMortems and a Just Culture* (https://codeascraft.com/2012/05/22/blameless-postmortems/). The post was about the need for 'blamelessness' in after-incident debriefings, in both verbal form (in the case of facilitated debriefings) and in written form (in the case of reports or other artefacts that come from the analysis).

Five years since writing that piece (which proved to be influential in my industry), I now understand that blamelessness is required for two important things.

1. to get real details from people as they experienced the outage (whether it's a degradation like a website or app being down or even a response to an active security breach).
2. to get different perspectives and specialties to come together and compare the different models they have in their minds about how things normally work.

The blameless postmortem is usually a semi-structured facilitated debriefing that involves some preparation of the timeline of events. Unlike in air navigation service providers, these are done in groups. This timeline will contain software logs, online 'chat' transcripts of what engineers communicated to each other during the event, and other artefacts such as diagrams or charts involving performance of various components impacted or involved during the issue.

The discussion uses the timeline as a scaffold for the group to build out context for details of the event. How people 'saw' problems and generated solution ideas are all the critical to flesh out the timeline. The resulting documentation of the event places importance on the perspectives given by people familiar with the event, as well as placing actions and decisions in the context in which they happened.

These blameless postmortems can provide rich data on work-as-done (as opposed to work-as-imagined, e.g., by management and colleagues), in the forms of both technical artefacts (logs, dashboards, etc.) and narratives about what happened, what people were trying to do, and what was affecting their work. Organisations that take this approach give engineering staff support for giving details about mistakes that they've made without fear of retribution or punishment.

The value of blameless postmortems turns out not to be the 'action items' that come from recommendations from this process. Of course, making recommendations for future design changes and introducing 'safeguards' for engineers working with the system (to reduce the likelihood of making a mistake) is valuable, reasonable, and good. But I have come to understand that the real value is in the dialogue during these debriefings.

Engineers can only make inferences about how things actually work – and therefore how they can break or fail. They have 'mental models' about what's happening in the code, in the network, between the components, etc. These can be compared to the air traffic controller's (mental) 'picture' of the traffic. The group debriefings (when facilitated well) encourage and support people to compare and contrast their mental models of how things work (and break) against each other, allowing a form of recalibration to take place. This can be understood via the parable of the 'blind men and the elephant':

> Six blind men were asked to determine what an elephant looked like by feeling different parts of the elephant's body. The blind man who feels a leg says the elephant is like a pillar; the one who feels the tail says the elephant is like a rope; the one who feels the trunk says the elephant is like a tree branch; the one who feels the ear says the elephant is like a hand fan; the one who feels the belly says the elephant is like a wall; and the one who feels the tusk says the elephant is like a solid pipe.
> A king explains to them:
> "All of you are right. The reason every one of you is telling it differently is because each one of you touched the different part of the elephant. The elephant has all the features you mentioned."

Figure 1: Like the blind looking to describe an elephant by pieces, software engineers can only glimpse and imagine parts of what they are responsible for

It's as if the blind men in the parable understood that they were all only experiencing part of the elephant, and were encouraged to talk through what each of them found, in order to aggregate their experiences, to produce a richer 'picture' of what an elephant is.

Of course, all metaphors have limits. To be a bit more accurate with respect to complex modern software systems, the elephant should be undergoing surgery, under attack by hunters, and engaged in some sort of elephant triathlon all at the same time.

The need to collaborate to bring combine individual perspectives into a more holistic picture of what is happening seems understandable, especially to those familiar with the real messy details. As well as accepting that one has a limited perspective, it is critical to be explicit with others continuously about what you are working with:

a) "here's my perspective on what is happening…now"
b) …how does what I'm seeing fit with what you're seeing?"

This applies to anyone working collaboratively in complex adaptive work, whether they are engineers with different specialised expertise and perspectives, or the various professions involved in the provision of air navigation services/air traffic management. The acceptance that your understanding is always incomplete and therefore always needs to be combined, contrasted, compared, and recalibrated with others' understanding is critical.

But as American author David Foster Wallace once stated in a now-famous commencement speech:
"…the most obvious and important realities are often the ones that are hardest to see and talk about."

Collaboration is one of those obvious and important realities.

Find out more

Allspaw, J. (2016, May 22). *Blameless postmortems and a just culture* [Blog post]. Available at: https://codeascraft.com/2012/05/22/blameless-postmortems/

Allspaw, J. (2016, November 17). *Etsy's debriefing facilitation guide for blameless postmortems* [Blog post]. Available at: https://codeascraft.com/2016/11/17/debriefing-facilitation-guide/

Allspaw, J., Evans, M., & Schauenberg, D. (2016) *Debriefing facilitation guide: Leading groups at Etsy to learn from accidents.* New York: Etsy. Available at: https://extfiles.etsy.com/DebriefingFacilitationGuide.pdf