The “Dropped Washer Effect”: Low-Profile Production Failures with High-Risk Consequences

Ken Ivanov

Take Digital Limited

Abstract

Two core prerequisites for a resilient system, product, or workflow are concise and robust design that defines the concept of that system, coupled with an efficient and error-free production process that embodies that design. The two factors are equally important for the system to fit its purpose efficiently: however high the design quality is, its perfection can be easily undermined by failure of the manufacturer to follow it right; conversely, even the perfect implementation won’t counterbalance flaws incorporated into the system on the design stage.

In this paper we discuss a class of production/manufacturing failures that we broadly refer to as the “dropped washer effect”, or DWE. To classify as a DWE, a production failure must satisfy three primary conditions: (i) create a subtle, hard-to-trace flaw in the product that presents a significant risk to the users of the product or the product itself; (ii) originate on the manufacturing stage, implying that it violates, openly or unintentionally, one or more of its design requirements or assumptions; (iii) occur despite reasonable quality assurance efforts undertaken by the manufacturing team. This represents a scenario where a properly designed, developed, and tested system or product, despite being produced in compliance with commonly accepted industry practices, ends up containing a hidden manufacturing flaw that presents a major risk to itself or its users.

In practice, dropped washer issues are rarely recognised as a special kind of failures and are often managed through general risk and quality management processes. However, evidence available to us suggests that conventional approaches lack efficiency when battling the DWE issues and often fail to spot them. As the risks associated with such issues are too high to be ignored, the problem of identifying and preventing the DWE issues requires a closer look.

In this paper we review the reasons that contribute to emergence of the dropped washers and discuss techniques of reducing the potential for their occurrence.

The Dropped Washer Effect

On 20th of August 2007, a five-year-old Boeing 737-800, having just landed in Okinawa, Japan after completing a Chinese Airlines flight 120 from Taipei, caught fire and burned out completely just after stopping at the gate. Luckily, none of the 165 aircraft occupants was killed or even hurt in the accident.

The investigation concluded that the ultimate reason for the accident was a missing metal washer on the slat stopping mechanism that had accidentally been dropped by an aircraft mechanic during recent maintenance works (Goto et.al 2009, 64). Due to the missing washer,

the stopping mechanism became loose and eventually came off its bolt, staying in the slat cavity. When the pilots retracted the slat after the landing, the loose slat stopping mechanism got caught between the retracting slat and the wall of the cavity, which, unluckily, shared its back wall with a fuel tank. The slat kept pressing on the detached stopping mechanism, eventually causing it to penetrate the wall of the fuel tank. The fuel started to leak through the hole into the slat cavity and further down the wing surface. Upon reaching the hot surfaces of the engine and wheel brakes, the fuel ignited, causing a large-scale fire.

The Flight 120 incident is one example of a phenomenon that we, for that reason, will be referring to as *the dropped washer effect*, or *DWE*. The DWE occurs as an outcome of a failure in the product or system manufacturing effort, and can be characterized by the following core properties:

- The delivered product[1] contains an unidentified, subtle, hard-to-trace flaw ("the dropped washer") that presents a significant risk to the users of the system or the system itself.

- The flaw originates on the manufacturing stage: had the product been built in full compliance with the designer's (architect's, lawmaker's) vision and expectations, the flaw wouldn't be there.

- The production of the product followed common industry practices for the projects of that kind, including due care, reasonably expected judgment by the project team members, and adequate risk management and quality control mechanisms[2].

Ultimately, we are speaking about *high-risk implementation errors that have a high chance of circumventing reasonable quality controls*.

The subtle nature of the dropped washers makes them difficult to tackle using standard project management tools (such as hiring better skilled people, increasing investment in QA, etc.), as chances are high that, despite the extra effort, they still slip the relevant quality controls. At the same time, risks associated with them are too high to tolerate.

This presents the need for a special approach that would deal with dropped washers efficiently, going beyond standard product management approaches. Further in this paper we will have a look into the typical reasons behind the dropped washer flaws, and discuss techniques that can reduce their chances of getting into the product.

Examples of the Dropped Washers

Dropped washers can happen in all kinds of production environments: physical or digital, innovative or traditional, in private or public sector. High-tech environments, though, due to their complexity, shorter TTMs, hotter markets, and, often, lower standards of professional qualification of the people involved, present the widest surface for them. Below are a few recent examples of the dropped washer-kind flaws from different industries.

---

[1] Here, we use a general term "product" to refer to a wider variety of man-made objects, systems, mechanisms, frameworks, processes, or workflows that provide value by solving a practical problem.

[2] And, certainly, included a reasonable margin for an "honest" error.

Flight 120, the "original" dropped washer, is best summarized in the investigation report:

It is highly probable that during the maintenance […] the washer on the nut side of the assembly fell off, following which the downstop on the nut side of the assembly fell off and then the downstop assembly eventually fell off the track. It is considered highly probable that a factor contributing to the detachment […] was the design of the downstop assembly, which was unable to prevent the assembly from falling off if the washer is not installed. […] Despite the fact that the nut was in a location difficult to access during the maintenance works, neither Boeing nor the Company had paid sufficient attention to this when preparing the Service Letter and Engineering Order job card, respectively. Also, neither the maintenance operator nor the job supervisor reported the difficulty of the job. [F120]

Using unsafe security parameters. This includes allowing users to choose weak passwords, using weak default security settings (such as a security camera PIN set to all-zeroes), or storing encryption keys next to the encrypted data (CWE321: Use of Hard-Coded Cryptographic Key. Mitre). These practices result in one of the biggest strategic fallacies, the false sense of security (the object appears protected, whereas it isn't). The following two examples can help understand the gravity of the problem: (i) research has shown that poorly chosen passwords are behind as many as 81% of corporate data breaches, with the percentage increasing over years, despite the problem being decades old (Bassett and Hylender 2022); (ii) between 2018 and 2023 there have been 2272 *new* cryptographic modules validated by NIST that support SHA1 – a security algorithm deprecated back in 2011 (Claburn 2022). Many vendors continue with the faulty practices not because they are unaware of the problem (they often are), but because the cons of sticking with the status quo are not tangible/punishable enough for them, whereas the benefits (compatibility with older tools and no extra upgrade costs involved) are quite clear.

The failure of the *MCAS system* on early Boeing 737 MAX airplanes became a reason for two deadly crashes that together took lives of 346 people. Only few Boeing engineers were aware of the existence of the system, with its operation being unnoticeable for the pilots. Aggressive cost cutting by the manufacturer and lack of balancing oversight resulted in numerous, though rarely manifested flaws of logical (non-overridability) and technical (relying on one, instead of two, angle of attack sensors) nature (The Design, Development & Certification of the Boeing 737 MAX, 2020). It is estimated that Boeing's direct losses from the mishap (not including legal liability, credit interest, or reputational losses) had exceeded $20bn by the end of 2020 (Isidore 2020).

Grenfell fire. The fire that started with what was referred as a "small kitchen fire" ended up engulfing the whole building and claiming 71 lives. At the time of writing the inquiry of the accident is still ongoing, but it is already clear that the primary catalyst for the fast spread of fire was the poor choice of the external insulation and cladding material, which ultimately acted as a source of fuel (Grenfell Tower Inquiry, 2017). The transcripts from public inquiry sessions reveal tragic combination of information loss in cross-supplier communications and negligent cost cutting through poorly defined responsibility bounds, which exposed itself through aggressive post-accident responsibility shifting ("passing the buck") (Apps 2022).

Deficient TLS certificate chain validation is a prominent example of a DWE flaw in digital environments. Numerous TLS implementations and integrations get server chain validation routine wrong, either by not performing it at all (and thus trusting *any* server certificate), or by seriously weakening it through oversimplification (to be fair, there are serious design

reasons that lead to that outcome (Mooney, Anise, and Barclay, 2019), but this doesn't make things any better). The TLS chain flaw opens an easy way for man-in-the-middle attacks, rendering the effect of the remaining security mechanisms of the protocol (however strong) void. Worse yet, higher-level modules and systems that consume such flawed TLS implementations often have no means of becoming aware of this major deficiency in their network transport ("it just works").

Dropped Washers: The Building Blocks

To be able to suggest efficient countermeasures against dropped washers, we need to understand the reasons that contribute to their occurrence in first place. Many of the dropped washer situations gravitate around certain attributes of the production process, with the primary causes being *low visibility*, *competence gaps,* and *diluted responsibility*.

Low business visibility[3]. The components that have a potential for becoming dropped washers often have little or no observable effect on the primary business task addressed by the product or the larger component they are part of. Whether there is a deficiency in such component or not, the key benefits provided by the product or component are preserved, with the product appearing to be doing its job as expected. It may take time for the deficiency to manifest itself, either through chance, long-term accumulation of error, or intentional exploitation. The slat mechanism was coping just fine for a prolonged period after the washer came off, creating an impression that everything was functioning properly.

The low visibility of the affected system components can propagate through the entire production hierarchy. On every stage of the production process, within every link of the supply chain, we aim to do things in the simplest, cheapest possible way. Spending extra time, effort, or money on a seemingly unimportant (if recognised at all) feature appears as wastage of valuable resources. The DWE-susceptible features create no *visible* added value; they often don't make it to the requirements list, and, when they do, they do not look important enough to provoke serious attitude from the project team. All that makes them the first to be dropped, ignored, or neglected.

Competence gaps. The majority of modern business environments organise production processes vertically: from business goals set at the top, down through the requirements specifications, planning and phasing processes, structuring work breakdown, to, ultimately, individual tasks performed by hands-on workers (Ikeda, Takao, and Sakamoto 2010, 478-481). With every step down this production hierarchy ladder, the essence of tasks at hand shifts from abstract and business-oriented matters ("what and why"), towards more specific, implementation-oriented tasks ("how"). Additionally, for knowledge-intensive products, there is often a qualification "hump" in the middle of the ladder that is represented by subject matter experts and senior members of the engineering teams, which typically are much more knowledgeable about the subject than their colleagues working above *and* below them.

Competence gaps happen when the differences between skills, competences, or mindsets of persons working on adjacent levels of the project hierarchy ladder are so big that they prevent

[3] Note that we are using the term "visibility", not "value", because the *value* of the potential dropped washer, though often overlooked, is high: it is the *reduction of the risk created by the component not being implemented properly*. If anything, it is more appropriate to speak of their *perceived* value, which, still, often influences our decisions.

full understanding of the task set by the higher-level person by the lower-level person. Being clear to the person that assigns it, the task appears vague, ambiguous, or intricate to the person who becomes directly responsible for implementing it. Unless the assignee realises and admits the deficiencies of their capabilities and asks for clarifications (and how many would?), there arises a risk that the task will be fulfilled improperly. Meanwhile, the person who delegated the task may have no realisation that it can be unclear to the assignee.

It is important to stress that competence gaps are not about poorly skilled workers that are unfit for their roles (we will consider that case separately below): it is about the assignment being too complex *for a reasonably qualified assignee*. If a transport company assigns a cat B driver to an HGV rotation, and the driver jack-knifes the lorry, it is hardly the driver's fault[4]. Competence gaps is a fundamental management flaw that affects the project work by introducing obstacles to composition, decomposition, and flow of tasks.

Competence gaps can happen not only in vertical hierarchies, but between any collaborating roles whose competences or mindsets differ significantly, either horizontally or vertically: architects and implementers (design documents too laconic/blurry), subject matter experts and project managers (jargon preventing understanding of the process fully), senior and junior engineers (task description omits important facts that appear obvious for the senior engineer), legislators and compliance officers (ambiguous wording), engineers and UI designers (a feature appears too complex for users), and many other kinds of interactions.

Diluted responsibility. The saying "when everyone is responsible, no one is" holds particularly true when it comes to DWE issues. The low visibility of potential dropped washers makes it very easy for them to slip through the fingers, and if there is no one to keep them on their list, it is very hard for them to re-gain attention.

One particularly risky area for the responsibility to wane are competence gaps at the intersection of two different project roles, separated either vertically or horizontally. Where the roles are separated vertically (manager-report, architect-developer), the assignor may assume that the implementer accepts responsibility for all the aspects of the task, even those not mentioned explicitly in their job description. The assignee may have a different view and interpret the task literally. Where the task is shared by two horizontally separated roles (two colleagues working on the same assignment), each of the parties may expect the other to be responsible for certain aspect of the task.

Deliberate avoidance is another common reason for the responsibility to get shrug off. The high risk associated with recognized DWE-susceptible components may discourage project members from taking responsibility for them, either pre- or post-factum, or stimulate them to shift responsibility to their colleagues, managers, reports, or users of the product.

Low visibility, competence gaps, and failure to establish strong individual responsibility form a sturdy basement for dropped washers. However, there also happen to be smaller, down-to-earth contributing factors that can either catalyse the primary three factors or produce dropped washers by themselves. Those are discussed below.

[4] Not to mention that the driver *does* have a chance to realise that they lack qualifications to perform the job – which is not the case in many dropped washer scenarios.

Complexity. The capacity of human brain is not limitless. Overly complex tasks can exceed natural limitations of human intelligence and willpower, leading to the person's inability to handle them up to expectations. Work on inherently complex jobs requires increased concentration and focus for prolonged periods of time. This can be mentally exhausting, particularly in modern high-pace work environments. The more complex the component of the system is, the more likely it is to exceed the natural limits of the product team members responsible for producing it. If that component, additionally, has reduced business visibility (meaning it has a higher chance of being neglected), the error may make it into the final product and stay there unnoticed.

Resource limitations. Most businesses run on the edge of their capacities, aiming to organize resources in such way that not a penny gets wasted. If a product can be developed without spending extra resources on a certain component or feature and still tick all the boxes in the requirements list, it *will* be developed in that way. Unless a feature has its own box to tick in the list, it will be discarded. *Time* here is just the same resource as any other.

Even if the feature gets recognized in the requirements document, it may still be cut short (or "unduly optimized") on the manufacturing stage because of resource shortage. Most production and management roles operate some form of task prioritization. Project staff, overloaded with a variety of product commitments, tend to focus on clear and tangible deliverables. QA focuses on ensuring that the project makes it up to the users' expectations before proceeding to the internals. Components that lack business visibility or are hard to understand for production team rarely end up at the top of that list.

Low workforce qualification. The speed of modern economy, at least in some of its sectors, makes it difficult for an average professional to acquire and maintain adequate level of qualification. One example is the skyrocketing IT industry, which in recent years has created a sheer deficit of qualified engineering workforce (Digital Leadership Report, 2022). This inevitably affects the overall quality of products created in the industry: the number of failures due to weaknesses in the open-source parts of a software supply chain increased by 650% between 2020 and 2021 (Krasner 2022); also, it is estimated that by 2025, 40% of IT budgets will be spent simply maintaining technical debt, and it's the primary reason that many modernization projects fail[5]. A significant proportion of people involved in software engineering activities lack skills (IET Skills and Demand in Industry. 2021 Survey, 2021), which often leads to them coming up with substandard and/or superficial solutions to problems they are assigned to solve.

By and large, the IT industry, which combines the overheated, lightning-fast market, the inherent complexity of the projects, and the high potential for security-related risks, provides a perfect ground for dropped washers. Still, IT is not the only such industry. The more demanding the project is, and the more complex the technologies it relies on are, the more likely it is that inadequately qualified specialists introduce dropped washers.

Poor communications clarity. Even in monocultural product teams, people that hold different functions on the team effectively talk different languages that depend on the roles they fulfil in the project. Ask a product manager, an architect, a project manager, and an engineer working on a specific module to describe the product they are making, and you will

[5] The concept of technical debt is related to the subject that we discuss in this paper, as both share the property of the low visibility and concealed nature of the deficiency.

get four very different answers. It takes effort for the project team to adapt, or "translate", their ideas when communicating them to their colleagues to ensure clear and accurate delivery. The most "undockable" joints, such as the one between the business and engineering functions, have long been recognized, and are typically assigned to specific roles (project managers) who act as interpreters between the two.

Smaller joints lack dedicated interpreters and, as such, are more susceptible to information losses during translation. Unlike roles with clearly recognized communication skills requirements, less "communicating" and more "problem-oriented" roles may fail to recognize the necessity to adapt their ideas to the mindset, skill level, and language of the intended recipient. Paired with the competence gap, this may result in the essence or importance of the communicated idea getting lost and not delivered to the recipients efficiently. It is not uncommon among authors of design specifications to choose not to disclose the reasons behind essential system components to the implementers, leaving them unaware of the purpose of the component and the risks associated with it not being implemented properly.

Improving clarity and integrity of product communication is one of the most efficient mechanisms of counteracting competence gaps and dilution of responsibility.

Poor risk management. Dropped washer components often slip properly constructed risk management processes, as the risk associated with them is not recognized in time due to their low visibility. Still, poorly implemented or non-existent risk management can add to the chances that dropped washers are not identified and mitigated in time.

Lack of accountability and control. It is often the case that assigners of the DWE-susceptible tasks, unlike the assignees, are knowledgeable of the potential hidden risks that improper fulfilment of the task can carry. Failure to exercise control over successful completion of such assignments, paying extra attention to the risks, can result in an overlooked dropped washer.

Personal attitudes (permanent or situational), such as negligence, laziness, poor work ethics, or personal circumstances, are behind a significant share of implementation errors in any kind of production environment (Strebler 2004). It is reasonable to assume that the more subtle, obscure, unclear, or complex the task at hand is, the higher chances there are for it to fall victim of the work ethics deficiencies. It is therefore vital that managers assign recognized DWE-susceptible tasks to team members with impeccable record of meticulousness, attentiveness to detail, and responsibility.

Passing the buck. Sometimes the DWE potential of a system component is properly recognized on certain production stage, but the owner of that stage decides not to mitigate it within their own area of responsibility but pass it over to the next stage in the production chain. Typical examples are instructions such as "park here at your own risk", "use a strong password", "follow building regulations when integrating this module". Responsibility shifts can happen on any stage of the product life cycle, from architects passing the task of picking up appropriate materials to the engineers, to car dealerships instructing customers to only use specific shampoo for the car they are buying.

There is nothing wrong in passing the buck over, *as long as the owner of the next stage in the chain understands the liability that comes with it and has sufficient competence to handle it*.

Yet, numerous manufactures choose to ignore that condition, using the loophole simply to indemnify themselves from the consequences of the risk.

Mitigating DWE Risks

The first question that we need to answer is this: is the DWE really a special kind of problem that requires bespoke handling? Can't the standard quality control mechanisms that we've had in place for many years – risk management, quality assurance, managerial oversight, certification, and validation – cope with the DWE issues just like they have successfully done with other kinds of production errors?

The author's experience in the IT security industry, as well as examples from other industries given above, suggest that they are. All the production environments mentioned in the examples section above did and do employ strict risk and quality management procedures. China Airlines, like any commercial airline, has a statutory obligation to comply with a stack of very tight regulations to be allowed to operate passenger flights, and so does Boeing. We have no reasons to assume that either company's quality processes were less robust or less efficient than those employed by other, less strictly regulated businesses.

The peculiarity of the DWE-susceptible components is that they tend to overload "normal" business processes because of their complicated, concealed, or low-value nature. The processes designed to work in a typical environment, driven by common business factors, and performed by adequately qualified workers, fail to detect them, as the dropped washers are neither typical nor common, and they often lay way above what is considered adequate qualification. Dropped washers are border (or even across-the-border) cases that fall out of mainstream production routines.

In practice, DWE problems are not typically recognized as a special class of issues. Even when the underlying risk ultimately manifests itself, the investigators rarely bother to have a closer look, categorizing the root cause as an unfortunate chain of events, human factor, or act of god: "a short-sighted developer made a mistake", "an irresponsible business optimized out an important safety measure", or "a poorly qualified technician pulled a wrong switch". And yet, the DWE issues can be too expensive to only be viewed from the post factum perspective ("why did it happen?", or, worse, "who should carry the blame?"). Due to the cost of the risk, it is often much cheaper to stop the washer from falling off in the first place, than to deal with the disaster of losing the aircraft or business reputation afterwards.

An efficient solution to the dropped washer phenomena would focus on *preventing* the dropped washers from creeping into the product, rather than hoping to *identify and eliminate* them when they already have. The post-processing methods do not work well with DWE issues, for the same reasons that make them appear in first place: it is not smart to invest time in validating a seemingly unimportant feature; it is hard to realise that the feature was implemented deficiently without being a specialist in that field; and, finally, it is impossible to locate a problem with an obscure feature that is *just not there* because it had been dismissed back on the requirements definition stage.

Not all products or production environments require dedicated DWE procedures though. Special attention to DWE should be paid by manufacturers whose products have a potential of causing major negative consequences through any kind of malfunction, inaction, or failure.

Basing on the three core properties of the DWE features, we can point out the following straightforward counterbalancing strategies of preventing them:

- increase visibility of the components that may become dropped washers,
- minimize the effect of competence gaps,
- manage responsibility efficiently.

The following techniques can be efficient in implementing the above strategies.

Prevent, not react. The inherent lack of visibility makes it much easier to spot components with the DWE potential on the planning stage, and then follow them up carefully throughout the production process to make sure they are handled adequately on each step, rather than try to identify and rectify them once the production completes. As soon as a potential DWE component has been identified, all the effort should be taken to keep it on the radar.

Avoid violation of competence limits. Using accurate, specific, and well-packaged task delivery across all production levels and matching the complexity of the tasks to the capabilities of the assignees helps avoid competence gaps and ensure smooth progress of the product requirements from the top to the bottom of the product hierarchy. Complex tasks can be simplified by using tools such as examples, checklists, and metaphors.

Deal with DWE one level higher. The best opportunity to spot the DWE potential in a component has the person that designs that component into the system. That person – which can be a system designer, a subject matter expert, or a product manager – has the best understanding of the role of the component in the system and the hidden risks it may carry. It is therefore natural to let that person "own" the component by making them responsible, fully or partially, for its error-free production.

Make features simpler. Complexity is one of the primary contributors to the breadth of competence gaps. A complex task can also impose a significant pressure on mental resources of the implementers, increasing their chances of making an unintentional error. *"Most current systems present the user with an intricate interface for specifying his protection needs. The result is that the user has trouble figuring out … and verifying that he requested the right thing. User interfaces that more closely match the mental models people have … are needed."* (Saltzer and Schroeder 1975, 1278-1308). Breaking a complex component into a selection of simpler, atomic sub-components can make it easier to understand, embrace, and make. Whereas a compound and/or complex component can be implemented partially (and rounded up to "implemented"), an atomic subcomponent is either implemented or not, making it easier to control and validate.

Design anti-DWE measures into the system. In certain cases, counter-DWE measures can be designed into the system, rendering it non-workable if the affected components are not manufactured properly (and by this enforcing the developers to make them right). Systems that have potential for grievous consequences, such as missile launch facilities, often require several inputs to be provided at a distance from each other, which makes it physically impossible for them to be activated by a single person. Additionally, each such activation point typically expects that a physical object ("a key") is inserted, which acts either as a conductor or circuit breaker, making the system physically inoperative without the key being put in place. Compare this to the certificate validation module in TLS: the other party's certificate sanity check is completely optional to the workability of the protocol, with the

parties still being able to talk to each other even if neither of them chooses to perform it. But what if the protocol refused to work, by design (e.g. through a cryptographically strong verification function), if the adequate certificate check was *not* performed properly by both parties? What if a signed email was unreadable, unless the recipient unambiguously proves that they *did* validate the signature? Building computationally strong, self-enforcing safeguards is a very promising direction in making resilient and provably safe systems.

Be proactive about DWE. Product team members who assign or subcontract packaged tasks to others, especially if such assignment involves major skill drop, should be considerate of DWE issues and take effort to reduce their surface. This may involve identifying potentially risky aspects of the task and either eliminating them or increasing their visibility to catalyse correct implementation. Thinking safety, trying to stand in the assignee's shoes, exchanging thoughts with less skilful colleagues, and scrutinising the task components for things that may go wrong are just some of the methods that can be used to identify potentially risky elements.

Production of a complex or security-critical product may go even further in preventing DWEs by instilling a culture of "deficiency whistleblowing": anyone on the team must have an open channel of communication with the management and be able to express their concerns about potentially risky features.

Joint effort. Counteracting DWE should be a joint effort of all parties involved in the production process: architects, project managers, team leads, and QA, but also parties external to the product: regulators, auditors, and professional bodies. Note that perceived importance of the DWE component and the amount of associated risk may be miscalculated by the entire project team, in which case external safeguards, of administrative or regulatory nature, may need to be introduced to satisfy safety requirements.

Facilitate communication. As dropped washers often happen on the border between the product's adjacent abstraction (vertical) or functional (horizontal) layers, reflecting gaps in knowledge, viewpoints, or visions of the respective team members, care should be taken to provide for efficient delivery of ideas and expectations between such layers. Establishing friendly communication culture between team members and encouraging them to speak freely and ask questions if they have concerns is a good method of increasing visibility of obscure and low-profile product elements.

Integrate DWE into business processes. Production environments that have dedicated formal or well-defined processes aimed at reduction of the DWE risks obviously have a higher chance of spotting those risks. Maintaining clear communication channels between architects, product management team, engineers, and quality assurance teams; training the QA to ask the right questions; creating and maintaining a high-quality documentation, both internal and external; incorporating DWE into the risk management framework all contribute to increased awareness of the DWE potential and higher chances of not letting them through.

Make violations expensive, either technically or administratively. Examples include direct penalties for not implementing the feature properly and compliance requirements that impose indirect costs through increased insurance premiums or lack of customer trust. Associating substandard implementation of a risky feature with an added cost improves its business visibility for the implementer and creates a clear incentive (loss aversion) to make it right.

Identify the right incentives. Punishment and having a price to pay are just one category of incentives that enforce adherence to design requirements. Taking it to the next level, identifying and suppressing incentives that pull assignees *away* from implementing the feature right, and creating counterbalancing incentives that push them *towards* it, is a good tactics of reducing the dropped washers' potential.

Avoid responsibility shifting across competence gaps. Shifting responsibility is a common technique on all levels of product development hierarchy, from requirements team ("Stick with ISO 27001 regulations when designing this module"), to system designers ("Ensure the user adheres to strong password rules"), to team managers ("Add security to the program database"). Responsibility shifts are not a bad thing by themselves. However, they may lead to implementation issues, particularly where a party delegates the task *without making sure the other party holds the competence necessary to deliver the job*.[6] Scrutinize tasks that you assign to the next-in-chain thoroughly to make sure the assignees have adequate competences to deliver them right.

Record newly discovered DWE risks. Whenever a potential for a DWE feature is recognised (on the planning, design, or production stage), record the finding in the project documentation and create a note for the implementers in the requirements document.

Make someone responsible. It is very important that the features that can become dropped washers have a designated owner who is responsible for implementing them right from the beginning to the end. We observed that in many dropped washer cases the blame ultimately ended up being spread between the manufacturer, the user, some external "adverse effects", and "human errors". The actual reason, however, - which was never mentioned aloud - was that no one had been ultimately responsible for the feature on the production stage. This caused the risks associated with it to be neglected, and the feature to eventually time-bomb – which could have possibly been avoided, should there be someone to cover the risks with their personal responsibility.

Balancing overheads, benefits, and costs. Integration of counter-DWE techniques into production processes certainly introduces overheads, which may face resistance from the product owners. It is rational to employ the cost-benefit approach when selecting the scope and extent of the measures, using DWE risk assessment results and expert judgement on potential consequences of those risks as inputs to the process.

That said, the techniques considered above integrate well into typical business processes. The dropped washers introduce a special kind of risk that is not recognised as a risk (creating a *meta* risk: a risk of a risk not being spotted). The above techniques, particularly, facilitate recognition and integration of that special risk into the routine risk management framework.

Many environments won't require any specific DWE procedures or will only need a minimal set. A system, every component of which can be wholly understood by every person working on it will rarely contain dropped washers. It is complex, modular, knowledge-intensive, and tightly coupled products and systems, with individual components interacting across abstraction layers and subject areas, that are the primary candidates for DWEs.

[6] It is not unusual for this clever technique to be used to shrug an inconvenient task off by blindly assigning it to the next person in the production chain.

Success Stories

Identifying successful examples of combating DWE issues is a tricky challenge because of the lack of availability of supporting product documentation, the reluctance of companies to take their rubbish out, the vague definition of the phenomenon itself, but primarily due to the "unknown unknown" nature of the DWEs (the fact that a product has not exposed any DWE problems *to date* does not imply that there are none). Still, a few industries and products are known to have practiced techniques similar to the ones described above and have a long-lasting track record of quality.

Civil aviation is one such industry. Between 2014 and 2018 there have been 7953 aviation accidents, with 4909 fatalities – a 47% and 24% decrease in absolute figures from 14991 accidents and 6489 fatalities that took place in 1984-1988 (Aviation and Plane Crash Statistics. 2019). This raises to 88% and 83% drop respectively when adjusted to the total increase in air traffic over these time periods (843M passengers were carried in 1986, versus 3.71B in 2016) (Air Traffic, Passengers Carried. 2020). Also, the data for the more recent period includes a Lion Air Boeing 737 MAX accident (which was caused by a DWE issue) that claimed 189 lives, and a shot down Malaysia MH17 flight that killed 298 people.

Another example is Microsoft's Windows Security Push initiative (Howard and Lipner 2003, 57-61), that resulted in major improvements in overall security and stability of their Windows OS and other products. Many Windows professionals consider Windows XP Service Pack 2, which was one of the immediate outcomes of the Push, to represent a huge leap in the operating system's overall security and technical quality.

Conclusions

The dropped washer issues pose a serious threat to manufacturing and deployment processes. They are hard to catch with general-purpose product and project management tools, not least due to their inherent ability to circumvent customary business practices. Often, we only find out about a dropped washer when the risk associated with it manifests itself through an adverse event, and even when that happens, we often resort to blaming, witch hunting, and attributing the issue to a "human error" or an "unfortunate chain of events" instead of recognising a fundamental product management problem behind the overlooked risk.

By putting a tag on the DWE phenomena and admitting the existence of the problem, companies can start working towards reducing the chances of the dropped washers slipping into their products. The three primary strategies of counteracting dropped washers are increasing the visibility of the potentially susceptible product components, avoiding competence gaps in task delegation, and ensuring unambiguous responsibility of the team members for every component of the product, however small or "unimportant". These strategies can be harmoniously integrated into existing product management frameworks, allowing for efficient mitigation of the DWE risks.

Bibliography

1. Goto, Norihiro, and Kusuki, Yukio, and Endo, Shinsuke. 2009. *Aircraft Accident Investigation Report. China Airlines B18616*. Japan Transport Safety Board. http://www.mlit.go.jp/jtsb/eng-air_report/B18616.pdf

2. Howard, Michael and Lipner, Steve. 2003. "Inside the Windows Security Push." *IEEE Security & Privacy*, vol. 1, no. 01.

3. Isidore, Chris. 2020. “Boeing’s 737 Max Debacle Could Be the Most Expensive Corporate Blunder Ever.” *CNN Business*, November 17, 2020. Accessed February 18, 2023. https://edition.cnn.com/2020/11/17/business/boeing-737-max-grounding-cost/index.html

4. Ikeda, Satoshi and Ito, Takao and Sakamoto, Makoto. 2010. “Discovering the efficient organization structure: Horizontal versus vertical.” *Artificial Life and Robotics*, 15.

5. Saltzer, Jerome H. and Schroeder, Michael D. 1975. “The protection of information in computer systems." *Proceedings of the IEEE*, vol. 63, no. 9.

6. Bassett, Gabriel, and Hylender, David, and Langlois Philippe et.al. 2022. “Data Breach Investigation Report 2008-2022”. *Verizon*, 2022. Accessed February 18, 2023.

7. Mooney, Nick, and Anise, Olabode, and Barclay, James. 2019. “Chain of Fools: An Exploration of Certificate Chain Validation Mishaps.” *Duo Security*. Accessed February 18, 2023. https://duo.com/labs/research/chain-of-fools

8. Claburn, Thomas. 2022. “NIST Says You Better Dump Weak SHA-1…by 2030.” *The Register*, December 16, 2022. Accessed February 18, 2023. https://www.theregister.com/2022/12/16/nist_sets_sha1_retirement_date/

9. Apps, Peter. 2022. “Grenfell Inquiry ‘able to conclude every death was avoidable’ as its lawyer slams ongoing ‘merry-go-round’ of buck-passing.” *Inside Housing*, November 10, 2022. Accessed February 18, 2023. https://www.insidehousing.co.uk/news/grenfell-inquiry-able-to-conclude-every-death-was-avoidable-as-its-lawyer-slams-ongoing-merry-go-round-of-buck-passing-79051

10. Krasner, Herb. 2022. “The Cost of Poor Software Quality in the US: A 2022 Report”. *CISQ,* December 15, 2022. Accessed February 18, 2023.

11. Strebler, Marie. 2004. “Tackling Poor Performance.” *Institute for Employment Studies*, IES Report 406.

12. Grenfell Tower Inquiry. 2017. "Phase 1 Report Overview. Report of the Public Inquiry into the Fire at Grenfell Tower on 14 June 2017”. ISBN 978-1-5286-1620-1.

13. The Design, Development & Certification of the Boeing 737 MAX. Final Committee Report. 2020. The Committee on Transportation and Infrastructure.

14. Digital Leadership Report 2022. 2022. Nash Squared.

15. IET Skills and Demand in Industry. 2021 Survey. 2021. The Institution of Engineering and Technology. Accessed February 18, 2023.

16. CWE321: Use of Hard-Coded Cryptographic Key. Mitre. Accessed February 18, 2023. https://cwe.mitre.org/data/definitions/321.html

17. Aviation and Plane Crash Statistics. 2019. PSBR. Accessed February 18, 2023. https://www.psbr.law/aviation_accident_statistics.html

18. Air Traffic, Passengers Carried. 2020. The World Bank / International Civil Aviation Organization. Accessed February 18, 2023. https://data.worldbank.org/indicator/IS.AIR.PSGR