ChatGPT

Managing Change: Why Smooth Transitions Go Unnoticed and How to Drive Continuous Evolution

The Repeating Scenario of Change Resistance

Imagine a team that has a well-established way of working with a **process or tool "A."** They perform tasks manually with familiar routines and are comfortable with the predictability of this status quo. Now suppose a leader proposes replacing A with a new **system "B."** The pattern that unfolds is common in change management:

- **A → B (Manual to Spreadsheet):** Initially, the team resists adopting B (e.g. a spreadsheet) because they don't immediately see the value. They ask, *"Why do we need this change? The current way works for us."* In reality, their comfort with A stems from its familiarity – it feels efficient to them because it's known and predictable, even if objectively A is inefficient. This comfort exists because doing what's already known requires less mental effort than learning something new [1]. The **brain prefers the "comfort zone"** of established habits, which minimize cognitive load and let people operate on autopilot [1]. Consequently, adopting B means a **temporary increase in effort and uncertainty**, which people instinctively avoid. Often, there is also a **loss-aversion bias** at play: the team focuses on the immediate "loss" of time/comfort in switching, rather than the future gains B will bring. For example, one might fixate on the hours of training needed now rather than the minutes saved each day later, overweighting the short-term cost of change over its long-term benefit [2]. All these factors cause **natural resistance** to change.

- **B → C (Spreadsheet to Online Tool):** Suppose the change leader succeeds in implementing system B and the team eventually becomes comfortable with it. Now the leader suggests moving from the spreadsheet (B) to an **online collaborative spreadsheet or database (C)** for better integrity, backup, and efficiency. **Once again, the same resistance surfaces.** The team questions the need: *"B works fine; why switch to C?"* The irony is that this is **the same team who initially resisted B** but now has grown accustomed to B and treats it as indispensable. They again struggle to see the *perceived value* of C and worry about the cost (in time and effort) of learning yet another new system. Just as before, the known tool feels safer and easier than the unknown one, so moving to C triggers the same cycle of pushback.

- **C → D (Online Tool to Custom Solution):** After C is adopted and running smoothly, the leader identifies further inefficiencies (for example, manual steps or limitations in the online tool) and proposes building a **custom automated solution or application D** (e.g. a serverless app with tailored UI and workflows). And **yet again – from C to D – the team's first reaction is resistance**, echoing the prior concerns: *"We just got used to C, why change to D?"* The pattern repeats itself: each time the current state becomes normalized, people treat it as the comfortable baseline and view any new change as a risky disruption.

This scenario highlights a core truth in change management: **people resist change repeatedly, even if they have gone through multiple successful changes before.** Comfort with the status quo, aversion

to loss, and fear of the unknown are powerful forces at every stage. It often surprises change leaders that even after *proving* the benefits of past changes, the next change meets fresh skepticism as if those past wins never happened.

Why Do People Resist Change Every Time?

Understanding the human psychology behind this repeated resistance is key to managing it. Several factors explain why team members push back at each step:

- **Comfort and Cognitive Ease:** People prefer what's familiar because it's mentally easier. Once a process is learned, it becomes routine and requires little conscious thought. Switching to something new resets that learning curve. As one analysis notes, *"The human brain prefers efficiency... The comfort zone is called that for a reason: it minimizes cognitive load and allows the brain to operate on autopilot."* [1] In our scenario, each system (A, then B, then C) became the team's "autopilot" mode. Even if A or B was inefficient objectively, it felt efficient to the users because they were used to it. Adopting a new system forces them back into a learning mode, increasing cognitive effort. This inherent preference for mental ease means **any change feels inconvenient at first**, prompting resistance.

- **Fear of the Unknown:** Change introduces uncertainty. Team members may worry whether the new system will truly work better, or whether they will struggle to adapt to it. There can be anxiety about short-term disruptions or even failure of the new approach. This fear is often not explicit, but manifests as skepticism: *"Why do we need this? What if it makes things worse?"* People naturally seek security in what they know; uncertainty triggers discomfort.

- **Loss Aversion and Immediate Costs:** A well-known bias in human behavior is to **favor avoiding losses over achieving gains**. In the context of change, this means people often perceive the *immediate "loss"* — the time, effort, and temporary dip in productivity required to transition — as more significant than the *future gains* the change will bring [2] [3]. For example, if adopting a new tool requires a few hours of training, users might fixate on those lost hours and downplay the fact that the new tool could save them several minutes *every day* thereafter. The pain of the initial inconvenience looms larger than the promise of long-term efficiency [4]. This skewed perception leads to comments like, *"It's not worth the hassle"*, even when rationally the change would pay off soon. In our scenario, at each step (A→B, B→C, etc.), the team was more concerned with the short-term disruption ("learning a new system, adjusting our workflow") than the prospective improvements, which felt abstract or uncertain.

- **Habit and Identity:** Over time, people often take pride in their mastery of a given system and in the reliability of their current process. Changing that can unconsciously feel like a threat to their sense of competence or control. They might think, *"We finally got things running smoothly with the current tool; changing it might mess things up or make my expertise less relevant."* This attachment to the existing way of working reinforces resistance.

In summary, resistance is the default human reaction to change, **even for people who coped with previous changes.** Each new change is seen in isolation as a fresh risk to comfort and stability. As a change leader, recognizing these psychological factors (comfort, fear, loss aversion, habit) helps in addressing concerns empathetically rather than dismissing them. It's not that the team is being "difficult"; their reactions are quite natural.

The Paradox of Invisible Change Success

One might expect that after leading a team through several improvements (A to B to C, etc.), a change agent would build credibility and trust. Intuitively, the team should remember that *"last time we resisted but it turned out great, so maybe we should listen this time."* However, a paradox emerges: **successful changes often become "invisible" to those who benefit from them**, and the change leader does **not** accumulate the expected "credit" for future initiatives.

Why does this happen? When change is executed smoothly and the new system works well, people quickly take it for granted. **Yesterday's innovation becomes today's normal.** The improvements gained from B or C soon feel like they were always there. Users may barely recall how clunky or inefficient the old process was, because the new process has seamlessly replaced it in their day-to-day experience. In fact, the only way they might truly appreciate the magnitude of improvement would be to *go back* and use the old system A or B again – which of course they won't do. As a result, the team doesn't viscerally feel how much better off they are now; the change agent's contributions fade into the background.

Moreover, the change agent likely shielded the team from the messy early phases of implementing the new system. Much of the experimentation, false starts, and extra work happened behind the scenes before rolling out each change. From the team's perspective, they saw a relatively polished new solution introduced with minimal disruption. This is good change management practice, but it has a side effect: **the easier you make the transition for the users, the less they realize the effort and ingenuity that went into it.** In their eyes, the new tool "just works" or the transition "just happened." Consequently, they may not attribute that success to the change leader's foresight or technical skill – it simply feels like a natural evolution.

When it comes time for the next change, team members treat it as a new, separate proposition rather than automatically trusting the change agent's track record. Each proposal (to move to the next system) is evaluated on its own merits and again triggers the familiar doubts. The leader might internally feel, *"After all the times I've improved things, why don't they give me the benefit of the doubt now?"* But the reality is the **credit from past changes is often invisible or quickly forgotten**. Sociotechnical research and anecdotal experience both confirm that when a transition is smooth, people rarely stop to celebrate it – they simply incorporate it and move on. The lack of drama means less memorable impact. As one industry observer of innovation noted, when pioneers hand off improvements to the mainstream, the pioneers often don't get credit for those improvements [5]. The operational team (settlers or town planners) may adopt the new tools and even take ownership of them, effectively "appropriating" the benefits while the pioneer's role fades from view [5]. This isn't usually malicious; it's a natural outcome of human focus on the present tasks.

In short, a successful change, by design, feels unremarkable once it's complete. The better you execute the change, the less "pain" is felt, and thus the achievement seems less extraordinary to those who didn't see the before/after contrast. This paradox means a change leader must be prepared to continually educate and persuade stakeholders anew for each step of evolution – past success alone won't always carry forward as persuasion capital.

Pioneers, Settlers, and Town Planners: Different Perspectives on Change

Simon Wardley's strategy framework provides a useful lens to understand this dynamic. In Wardley Maps, **systems evolve** from novel, custom-built solutions toward more standardized products and utilities. Corresponding to this evolution, there are different organizational roles: **Pioneers, Settlers, and Town Planners** [6].

- **Pioneers** thrive on innovation and exploration. They work on green-field ideas (Wardley's *"Genesis"* stage) and are comfortable with trial-and-error. In our scenario, the change leader acts as a Pioneer – envisioning new possibilities (B, C, D) and building early versions of those solutions. Pioneers accept that early-stage solutions can be messy and prone to failure as they figure things out.

- **Settlers** take those promising innovations and refine them. They turn a rough prototype into a reliable product, smoothing out the edges for broader use. In the context of the team, if a few team members get involved in piloting the new system and adjusting it for daily use, they are playing a Settler role.

- **Town Planners** focus on stability, scale, and efficiency. They take well-understood, proven tools or processes and industrialize them – think of maintaining core systems, optimizing cost, and ensuring consistency. The bulk of the team using system A or B in a routine, repeatable way are acting as Town Planners. They value predictability and low risk.

Crucially, **these roles correspond to mindsets.** The *pioneer mindset* is excited by change ("let's try something new!"), whereas the *town planner mindset* is wary of change ("let's not break what works"). Neither is right or wrong – both are necessary – but they naturally come into tension. In our story, the leader was effectively a Pioneer introducing change into a group of Town Planners. The Town Planners did their job: they kept things running and only engaged with the new solution when it was sufficiently stable and productized. From their viewpoint, the leader's new ideas were always a disruption of their well-oiled machine, rather than an immediately obvious improvement.

Wardley's model helps explain why **prior successful changes don't automatically reassure the Town Planners**. Each time, the team's Town Planners saw the change only when it was nearly complete (a new "productized" system ready to deploy). They were not part of the Pioneer's journey to get there. Thus, they never experienced the full before-and-after jolt – they only stepped from one relatively solid platform to the next. Because Pioneers and Town Planners "live in different worlds" in terms of perspective, trust isn't transferred automatically. The Town Planner mindset focuses on execution within the current paradigm; it doesn't concern itself with the *process* of change, only the *outcome*. So when the Pioneer comes again with a new idea, the cycle of skepticism resets.

Understanding these distinct perspectives is helpful for a change leader: it reminds us that **speaking the language of those who value stability** is important when pitching a change. For example, to persuade Town Planners, one should emphasize how the new solution (D) will eventually become just as reliable and routine as the old one, and how it addresses real pain points or inefficiencies in the current system. It's also a reminder that the *pioneer agent of change should not expect applause or automatic buy-in* – the reward often lies in seeing the organization improve, not in receiving credit. As one essay on the Wardley framework notes, *"As pioneers find something with growth potential, it is time for settlers to make that growth happen. That transition is never smooth because settlers may appropriate the tools and techniques and even deprive pioneers of the credit."* [5] In other words, once the change is in effect, the

organization's "settlers" and "planners" run with it, and the pioneer's role (and recognition) naturally diminishes. A savvy change leader accepts this and keeps pushing forward to the next innovation regardless.

Strategies for Implementing Change Effectively

Given these challenges, how can a leader successfully drive change, knowing that human resistance is inevitable and that great execution can make their work invisible? Here are some **strategies for effective change management** drawn from experience and the scenario described:

1. Develop the New Solution in a Safe Sandbox: Rather than dragging the whole team into the chaos of building a new system from scratch, **create a separate pilot team or project group** to explore and develop the change. This could be just the change leader alone or a small "innovation team" – essentially the Pioneers. By isolating the exploratory phase, you **shield the broader team from the high-failure-rate trials and complexity** that come with anything new. For example, the leader might build the initial spreadsheet or prototype database on their own or with one volunteer, testing workflows and ironing out kinks, *before* asking everyone to move to it. This protects the main team (the Town Planners) from having to experience the new system in its rough, unfinished state – which would only frustrate them and increase their resistance. When the pilot team has a working, refined solution, then it's time to introduce it to the wider group. This approach respects everyone's roles: the pioneers innovate freely, and the town planners aren't bothered until there's something of proven value to adopt.

2. Make New and Old Coexist (for a While): Wherever possible, **plan an evolutionary path where the new system can run in parallel with the old before fully replacing it.** This might involve data migration scripts, interoperability, or phased rollouts. The goal is to let users get comfortable with the new system gradually, and to ensure it can handle all essential tasks of the old system. In our scenario, this could mean keeping the manual process A as backup while introducing the spreadsheet B for certain parts, or maintaining the spreadsheet B while a few users trial the online system C, and so on. During this overlap, gather feedback and adjust the new system to cover any gaps. When people see that *both A and B work*, and B clearly makes some tasks easier, they'll be more willing to let go of A. **The transition feels less like a risky leap and more like a natural progression.** Additionally, having a fallback (even if only perceived) reduces fear—users know that if something goes wrong, the old way is still there temporarily.

3. Emphasize Compatibility and Familiarity: Design the new solution such that it **feels as familiar as possible** to users of the old solution. The less drastic the change in user experience, the lower the resistance. In the example, moving from a manual ledger (A) to Excel (B) preserved the tabular record-keeping concept, just digitized it. Moving from Excel (B) to an online Google Sheet (C) kept the spreadsheet interface, just in a web browser. Each step was an evolution, not a radical departure in how the work is done. This is intentional. It leverages the concept of **small, incremental changes** [7] to reframe change as safe and manageable. When proposing the next step (C to D), smart change leaders will highlight elements that remain *the same*: e.g., "D will have a dashboard similar to what you're used to, but now it will automate the reporting for you." By reducing the perceived "distance" from old to new, you lower cognitive load and fear.

4. Demonstrate Value Early: It's easier to win support when people can **see concrete improvements** the new system offers. Wherever possible, share prototypes, demos, or pilot results that showcase time saved, errors reduced, or new capabilities. For instance, the leader could show that system C (online) eliminates duplicate data entry that was plaguing the spreadsheet, or that system D can pull up a report in seconds that used to take an hour. Visualizing the benefits helps overcome the "I don't see why

this is necessary" argument. It turns an abstract promise into a tangible reality. Even better, if you can quantify the improvement (e.g., "This will save each of you 30 minutes a day" or "We'll reduce data errors by 50%"), do so – it appeals to the logical side and can outweigh the emotional aversion to change [4]. **Communicate the "why"** behind the change in terms that matter to the users: faster results, less drudgery, more reliability, etc., rather than just "because it's new or better technology."

5. Secure Leadership Support and Resources: Significant changes often require investment (time, money, or authority). Having executive endorsement can be crucial, both to allocate necessary resources and to signal to the team that the change is important. In the scenario, if moving to a new system is framed as an official project (with sponsorship), it underlines that this is a supported strategic move, not just one person's whim. Support from the top can also help in overcoming stalwart resistance – for example, if some processes need to pause or adjust during the transition, leadership can authorize that. **Use leadership support carefully** – it's better to persuade with benefits than to mandate by decree, but knowing you have backing means you can push through reasonable resistance confident that it's sanctioned.

6. Educate and Train Users – with Empathy: A major source of resistance is the fear of feeling incompetent with the new system. To counter this, provide **training, documentation, and hands-on support** during the transition. Show empathy that "learning something new is hard," and arrange peer mentoring or extra help in the early phase. When people feel they will be supported (and not judged) as they climb the learning curve, their fear of change is lessened. Celebrate small wins as the team adapts – for example, acknowledge when a team member finds the new system useful for the first time. This positive reinforcement encourages others that the change is workable. Remember that from the user perspective, **the change isn't "done" until they feel as competent and in control with the new system as they were with the old.** Providing the means to reach that comfort is part of change management.

7. Iterate and Incorporate Feedback: Treat the rollout of change as a two-way process. Solicit feedback early and often from the team as they start using the new system. There will inevitably be tweaks needed – perhaps a feature from the old process that people quietly relied on needs to be added into the new, or some workflow in the new system is causing confusion. By listening and improving, you show that the system is *for* them, not imposed *on* them. This collaborative approach can turn skeptics into contributors. It also increases the likelihood that the new system truly covers all the use cases the old one did, plus the new benefits. (Recall that a common failure is when a new system lacks one or two key functions of the old, giving holdouts a reason to stick with the legacy tool "just for that one thing." User feedback helps prevent such oversights.)

8. Retire the Old System Deliberately: Plan for the decommissioning of the old way once the new is established. Leaving the old process in place "just in case" for too long can drag out full adoption and even allow regression. Once you've migrated all necessary functions and proven the new system, **make a clean cut** – archive the old spreadsheets, remove access to deprecated tools, etc., so that everyone commits to D fully. This sends a message of confidence in the new system and prevents backsliding. Of course, do this only after sufficient overlap and proof that the new system can handle everything. When done at the right time, retiring the old system can also be a ceremonial moment to recognize how far the team has come.

By following these strategies, a change leader can navigate the delicate balance of pushing for improvement while respecting the very real human factors at play. The overarching theme is to **minimize the pain of change for the users** – which, as we discussed, might also minimize their awareness of how much changed! That's okay; success is measured in outcomes, and if the outcome is a better, smoothly adopted system, then the change management effort has done its job.

Continuous Evolution vs. Big Bang Changes

It's worth highlighting that in our scenario the changes were iterative: A to B to C to D, rather than jumping straight from A to D in one leap. This approach of **continuous evolution** is generally more effective than attempting a "big bang" radical change. Large, revolutionary overhauls sound exciting but often fail in practice, because they run headlong into user resistance, unforeseen complexities, and the risks of trying to do too much at once. By evolving in stages, you gain several advantages:

- **Learning in Stages:** Moving step-by-step allows the change team to learn from each phase and incorporate those lessons into subsequent changes. For instance, the experience of going from A to B might reveal what data structures or workflows are critical, which can inform the design of C. The team accumulates domain knowledge and technical knowledge incrementally. If one tried to design D (a fully automated system) from the starting point of A without those intermediate steps, it's likely they'd miss important requirements or subtleties that only became evident while using B or C. Each evolutionary step is an opportunity to refine understanding.

- **Maintaining Continuity:** Smaller changes mean the organization can continue operating with less disruption. You don't halt everything for a year-long project and hope that at the end it works. Instead, you implement manageable changes that each deliver some value and keep things running. This steady pace avoids the scenario of a "revolution" that fails and leaves the organization with nothing to show for a massive effort. Instead, you're *always* moving forward, even if in modest increments.

- **Building Cumulative Benefits:** Lots of little improvements can add up to a substantial leap in performance over time. It may not be flashy, but compounding efficiencies are powerful. For example, each system upgrade might save 15% time on certain tasks; after a few iterations you might have doubled productivity versus the original process, even though no single change felt like a game-changer. And because each change was absorbed successfully, the organization actually realizes those gains (as opposed to an ambitious change that never fully gets adopted).

- **Avoiding Obsolescence of New Ideas:** In fast-moving fields, a long big-bang project can be risky because by the time you deliver, the solution could be outdated. Continuous evolution ensures you're iterating in shorter cycles, able to adjust to new realities. Perhaps between B and C a new technology (like a better online service) emerged – by doing things in steps, you had the chance to incorporate that rather than locking in all decisions from the start.

However, continuous evolution does not mean change for change's sake. It's important to **know when to stop or pause**. Each change should be driven by a meaningful improvement. Eventually, you may hit **diminishing returns** – where moving from, say, D to E would only yield a tiny extra benefit not worth the disruption. The art of change management is to sense when a system has reached a sweet spot of efficiency and stability, so you don't keep changing things unnecessarily. In our story, the leader would stop pushing new systems once the team's needs are met and no glaring inefficiencies remain. Pushing a content team from D to E to F without clear purpose would only breed "change fatigue."

A key measure of success for each evolutionary step is **complete adoption and retirement of the old system**. If after moving to D, you discover that some team members are secretly still using the old spreadsheet C or A "because it does X better," that's a red flag. It means the new solution didn't fully replicate or improve an aspect the old one had, leaving a gap. This situation – where multiple legacy tools linger on – is the worst-case scenario for efficiency. It often happens when changes are rushed or not fully thought through (for example, a new system is missing a report that the old one had, so

people keep the old system around just to run that report). To avoid this, the change leader must thoroughly understand the existing workflows (including those "little" features people rely on) and ensure the new system covers them. It goes back to taking time to learn from each stage and not skipping steps. When done right, each new system *completely subsumes* the old system's functions (and adds more). You can then confidently turn off the old system and have everyone working in the new environment, which maximizes the benefits.

In summary, **continuous improvement** is typically more sustainable than dramatic overhaul. It aligns with how humans adjust (gradually) and how complex systems can be evolved. Each incremental change should be justifiable by clear value, and the process should continue until the **cost of changing exceeds the benefits** (i.e. diminishing returns). At that point, you may stick with a well-optimized "D" for some time, and that's fine – the goal isn't change itself, but improvement. And when a new technology or need eventually arises that *does* promise a big jump in value, the organization will be in a good position to tackle it, having built a muscle for change through the continuous evolution practice.

Conclusion

Change management is as much about people as it is about technology or processes. The scenario of taking a team from A to B to C to D illustrates the fundamental challenge: **people don't resist *improvement*, but they do resist *being changed*.** Good change management finds ways to deliver improvement while respecting the natural desire for stability and familiarity. As a change leader, you must anticipate resistance at every step, no matter how many past changes you've successfully implemented. Rather than becoming frustrated, use that knowledge to plan your approach: communicate clearly, minimize disruption, and support your team through the transition.

Remember the paradox that if you do your job well, it may feel like the change "just happened" and everyone swiftly moves on. You might not get accolades for every smooth transition – in fact, the smoother it is, the less fanfare it garners. **That's okay**. The true reward is seeing the team become more effective and knowing you facilitated that leap, however quietly. Over time, an organization that continuously adapts will far outperform one that clings to comfort. And while team members might not cheer each incremental change, they will certainly appreciate (in hindsight) that their work is a lot easier and better than it was a few years ago.

Finally, it's important to foster a culture that slowly becomes more change-ready. Each successful evolution can be a subtle lesson that change isn't something to fear. By involving people in improvements and showing empathy to their concerns, you build trust that *this new change is not chaos, but a carefully guided step forward.* In the long run, you want your team to internalize that **change is a constant** – and even if they grumble each time, they know deep down that adaptation is part of growth. In fact, thriving in a modern work environment means becoming comfortable with being uncomfortable at times. As one commentary on organizational change put it, *"when the only constant is change, becoming a specialist in navigating change is a core competence"* [8]. In other words, helping your team (and yourself) develop the skills and mindset to handle continuous change may be the most valuable outcome of all.

By managing changes in a thoughtful, human-centric way, you not only implement a specific improvement, but also strengthen your team's adaptability for the future. And that adaptability — the confidence to evolve repeatedly — is the hallmark of resilient, successful organizations.

---

1 7 Why Some People Struggle with Change (Even When They Say They Don’t) — Openmind
https://www.openmindglobal.io/blog/why-some-people-struggle-with-change

2 3 4 Reasons Behind Change Resistance
https://www.mastersinminds.com/reasons-behind-change-resistance

5 6 8 2024-03-12-pioneer-settler-town-planner.md
https://github.com/nastacio/nastacio.github.io/blob/1220d9c4070ea109330bc5bb38eb7b153555d61d/_posts/2024-03-12-pioneer-settler-town-planner.md