Solve your problem with a wink

The System Changed, and I Got Overloaded

How to Navigate a Mandatory Workplace Tool Migration Without Taking the Entire Transition on Your Shoulders



Max Paradox

The System Changed, and I Got Overloaded

How to Navigate a Mandatory Workplace Tool Migration Without Taking the Entire Transition on Your Shoulders

Max Paradox

PROBLEM? RELAX. YOU'VE GOT THIS!

ISBN: 9788384906538

ALL-IN Rafał Płonkowski



CONTENTS

INTRO 5

Chapter 1 - When a System Change Becomes Your Second Full-Time Job 10

Chapter 2 - Why the New Tool Was Supposed to Save Time but Is Devouring It Instead 19

Chapter 3 - The Trap of Being the Person Who Always Figures Things Out 31

Chapter 4 - Five False Solutions That Only Create More Work 40

Chapter 5 - Who's Responsible for This? Or How to Stop Playing Organizational Detective 50

Chapter 6 - Your Inbox Is Not Migration Command Central 59

Chapter 7 - Two Systems, Three Versions of the Truth, and the Person Expected to Reconcile Them All 69

Chapter 8 - You Don't Have to Fix Everything Before the End of the Day 78

Chapter 9 - A Migration Plan That Fits into Your Calendar 88

Chapter 10 - How to Talk to Your Manager Without Ending Up at "We Just Have to Make It Work" 98

Chapter 11 - Report Problems So Someone Can Actually Solve Them 108

Chapter 12 - Learn the New System, but Don't Try to Become Its Encyclopedia 119

Chapter 13 - Hand Over a Task Without Having It Come Back Gift-Wrapped 131

Chapter 14 - Go-Live Day: How to Make Decisions When Everything Happens at Once.....142

Chapter 15 - When the Migration Goes Wrong and You Start Blaming Yourself 154

Chapter 16 - When the New System Still Doesn't Work and Everyone Tells You to Be Patient.....164

Chapter 17 - When You Have No Energy Left for Another Change: How to Simplify Work Without Abandoning Responsibility 175

Chapter 18 - How to Maintain Order After the Migration Is Officially Over 186

Chapter 19 - The First Signs of a Relapse: How to Avoid Becoming the Person Who Does Everything Again.....197

Chapter 20 - The System Can Change. You Don't Have to Become the Entire Implementation Department.....208

INTRO

On Monday morning, you open your computer and discover that the tool you could still work with perfectly well on Friday has officially become a thing of the past. In its place, there is a new system. More modern, more intuitive, and supposedly designed with the user in mind. Particularly the kind of user who has three spare hours, a flawless memory, and a special talent for finding buttons hidden by a designer who apparently grew up playing puzzle games.

An email is waiting in your inbox with the subject line "Important Information Regarding Digital Transformation." It informs you that the change will increase efficiency, improve collaboration, and simplify everyday processes. Sounds promising. It's just a shame that all your existing tasks still need to be completed, the new tool doesn't recognize some of the data, and the user manual is forty-seven pages long and begins with the sentence: "The process is simple and intuitive."

The most intuitive thing so far is the urge to close your laptop.

Then comes the first message from a colleague: "Do you know how to generate a report now?" The second: "Can you check why I don't have access?" The third comes from your manager, who politely reminds everyone that the system change should not affect the achievement of current targets. Of course. In exactly the same way that renovating a kitchen shouldn't interfere with preparing a three-course dinner. You simply need to demonstrate flexibility and avoid getting distracted by minor details, such as the absence of a stove.

Before noon, you are already performing three roles. You are doing your actual job, learning the new tool, and helping everyone else survive the digital revolution. Nobody has formally assigned you that third responsibility. You just happened to solve a few problems, so the organization decided it had discovered someone who was exceptionally good at handling difficulties. Congratulations. You have been promoted to the unofficial IT support department. Without additional pay, naturally, but with an unlimited number of incoming requests.

And this is precisely where the problem we will explore in this book begins.

The issue is not simply that new systems can be inconvenient, implementations poorly prepared, and corporate announcements suspiciously similar to advertising materials written by people who have never tried logging into their own product. Changing tools is often necessary and can bring genuine benefits. The problem begins when the cost of that change is quietly transferred to employees who are expected to maintain their previous productivity while simultaneously learning new procedures, correcting errors, and reassuring everyone around them.

You are particularly likely to fall into this trap if you are responsible, helpful, and reasonably comfortable with technology. You know the processes, understand how different activities depend on one another, and can usually work out why something isn't functioning properly. When others get stuck, turning to you feels natural. At first, you may not even mind. Helping a colleague takes five minutes; explaining a new procedure takes another ten. Only after several days do you notice that you are completing your own responsibilities in the evening, while your calendar has

started resembling a parking lot where everyone has abandoned their cars and apparently has no intention of collecting them.

Worse still, you may begin interpreting your overload as evidence that you are somehow not competent enough. If everyone else seems to be managing, perhaps you are learning too slowly. Perhaps you should organize your time better, read the instructions more carefully, or devote your weekend to additional training. Except that you can only see part of the situation. You don't know how much work other people are postponing, which unofficial workarounds they are using, or which tasks they have simply stopped doing. Comparing your own chaos with someone else's calm facial expression is a remarkably unreliable way of measuring performance.

So let's establish one important point: a mandatory system migration is an organizational undertaking, not a private examination of your resourcefulness.

That distinction changes a great deal. The organization decides to replace a tool, sets the deadline, selects the solution, and determines how the implementation will proceed. Employees are expected to cooperate within the scope of their roles, learn the required activities, and report problems. That does not automatically mean they should assume responsibility for inadequate training, incorrect access permissions, inconsistent instructions, missing technical support, or unrealistic deadlines. You can be committed to the change while also stating clearly that a particular task requires additional time, a management decision, or assistance from the appropriate team.

Of course, putting this into practice is not quite so straightforward. Writing "set boundaries" is easy. Actually

doing it during a meeting where your manager asks whether you could help the entire department "just for a moment" is another matter. Particularly when your professional reputation matters to you, you don't want to leave colleagues struggling, and you know that refusing a request might be interpreted negatively. This book will not pretend that raising one eyebrow assertively is enough to make every organizational problem disappear. It is a guide to taking practical action under real working conditions, including situations where you have limited influence over decisions and very little room for negotiation.

We will examine how to recognize tasks that genuinely belong to your role and responsibilities that land on your desk simply because you happen to know how to handle them. You will learn to distinguish difficulties associated with your own learning process from technical faults and shortcomings in the implementation itself. You will see how to report problems clearly, document their impact on your work, and discuss priorities without immediately turning the conversation into a confrontation. We will also look at how to support colleagues without becoming a twenty-four-hour information desk, because being helpful does not require accepting every incoming request with the enthusiasm of a call-center employee on the first day of a major promotion.

What you will not find here is a magical method that makes every new tool convenient, every instruction understandable, and every manager instantly receptive to your arguments. Some implementations really are badly prepared. Sometimes additional work is unavoidable, and sometimes a temporary drop in productivity cannot be prevented. What you can do, however, is reduce the chaos, distribute responsibilities more sensibly, and stop treating

every organizational gap as a personal invitation to rescue the entire operation.

To begin, one simple question will be useful. When another task appears during the next day of the migration, ask yourself: "Is this part of my job, an agreed responsibility supporting the implementation, or a problem I'm taking over simply because nobody else has dealt with it?" You do not have to refuse immediately. Sometimes it is enough to ask which existing task should be postponed, who needs to make the decision, or where the request should be forwarded. Such questions are not signs of unwillingness to cooperate. They help make visible the costs that have previously gone unnoticed.

Because the objective is not to become the person who answers every question with, "That's not my problem." Admittedly, that approach might free up quite a lot of your time, but it could also mean you stop receiving invitations to the more enjoyable meetings. The goal is something considerably more reasonable: to do your own job well, participate professionally in the change, and help where helping makes sense, without paying for the entire implementation with your own exhaustion.

The new system may require you to learn different ways of working. It does not have to require you to become the person responsible for absolutely everything.

And if the organization genuinely needs someone to lead the migration, that responsibility should have a name, a defined scope, an agreed priority, and appropriate support. It should not be a secret job function that activates automatically after your third successful password reset.

Chapter 1 - When a System Change Becomes Your Second Full-Time Job

A meeting titled "New Platform Implementation Kick-off" appears in the company calendar. It lasts forty-five minutes. The first fifteen are devoted to the strategic vision, the next twenty to a presentation of the features, and the final ten to questions. Unfortunately, just as participants begin asking specific questions, the presenter announces that time is up and all outstanding queries can be sent to a specially created mailbox. The mailbox has a beautiful name, an automatic reply, and probably an inner life of its own. Nobody has yet seen it solve a problem, but at least it confirms receipt.

The following day, you receive access to the platform. Logging in works, although only on the third attempt. Some data has been migrated, some is stored somewhere else, and several essential functions are nowhere to be found. The manual says everything is available under the "Management" tab, which you apparently don't have. You contact support. In response, you are asked to send a screenshot, your user ID, and a description of the problem. An hour later, you learn that your ticket has been forwarded to the appropriate team. Which team remains a mystery. Perhaps a team of digital archaeologists currently trying to establish where the missing tab was buried.

Meanwhile, you still have your regular duties to perform. A customer is waiting for information, your manager is waiting for a report, a colleague is waiting for confirmation, and the new platform is waiting for your patience to run out. Because you understand the previous process and know

how to connect the dots, you start looking for a workaround. You export the data, compare it with the old file, and manually correct several entries. It works. The report is completed, the deadline is met, and everyone is happy. Nobody, however, asks how much additional work was required to achieve this success.

This is the first moment when you should pay close attention.

The Success That Hides a Problem

From the organization's perspective, the task has been completed. The report arrived on time, the customer received an answer, and the on-time delivery indicator remains green. If nobody reports the additional effort involved, the management system will see no difference between work completed efficiently and work rescued through two hours of manually correcting data. That green indicator looks magnificent. It can be displayed in a presentation alongside a reassuring statement that the transformation is progressing according to plan.

Meanwhile, the person who made this success possible finishes the day with a backlog. Their actual responsibilities have not disappeared. They have simply been pushed back, along with their break, a conversation with the team, and the opportunity to think calmly about an important decision. Because the situation appeared to have worked out, expectations remain unchanged the following day. The organization received no signal that the process needed fixing. Instead, it received evidence that the process could somehow be made to work.

This does not mean you should deliberately allow tasks to fail. Not every workaround is a mistake. Temporary solutions

are sometimes necessary during implementations, particularly when deadlines, data security, or continuity of service are at stake. The problem is failing to communicate their cost. An emergency measure then starts functioning as the normal way of working, and the person who devised it becomes the unofficial owner of the process. Initially through personal initiative, and later because everyone involved has become accustomed to it.

Imagine that, for an entire week, you manually complete data that the new system fails to transfer automatically. On the first day, it takes twenty minutes. On the second, forty, because additional cases appear. By Friday, you have your own spreadsheet of corrections and a set of instructions for your colleagues. The following week, someone says, "Let's do what we did last time, since it worked." Congratulations. A process has just been born. It has no formal owner, no approved procedure, and no defined end date. What it does have is you.

And you certainly never signed an employment contract for the position of "Specialist in Everything That Temporarily Doesn't Work."

Three Types of Work That Look Like One

Overload during a migration is easy to overlook because the additional tasks resemble ordinary work. You open the same files, communicate with the same people, and pursue similar objectives. What changes, however, is the type of effort required to achieve those objectives. It is worth distinguishing three areas.

The first is your core work: the responsibilities you are accountable for regardless of which tool you use. These might include analyzing data, serving customers, planning,

selling, coordinating a team, or preparing documents. The second area is learning a new way of performing those responsibilities. This is a natural part of implementation. The third involves correcting shortcomings in the migration itself: searching for missing information, reporting faults, testing workarounds, explaining unclear procedures to others, and reconstructing activities that previously happened automatically.

These areas can overlap, but they should not become indistinguishable. If preparing a report used to take one hour and now takes three, you are dealing with a genuine change in workload. One hour is still devoted to completing the task itself. The remaining two may be caused by learning requirements or implementation problems. The distinction matters because each cause requires a different response. Learning may require training and practice. A system defect requires a fix. A lack of agreed priorities requires a decision from the person managing the work.

If you describe all three problems simply as "I have too much work," it becomes difficult to identify a solution. Your manager might suggest better time management, IT might recommend additional training, and the author of the manual might advise you to reread page twenty-three. Each person would be answering a slightly different question. Your task, therefore, is to identify the source of the difficulty before seeking help. This is not a bureaucratic obsession. It is a way to avoid trying to repair a system defect by creating a more ambitious calendar.

Why the Second Job Appears Without a Vacancy Announcement

Unofficial responsibility rarely arrives as a formal instruction. Usually, it grows through small decisions that

seem perfectly reasonable at the time. Someone asks for help, so you help. They then recommend you to another person. That person passes your contact details to someone else because "they know how this works." Within a few days, you become the natural point of contact, even though nobody has established how much time you can devote to these requests or whether you have the appropriate permissions.

Organizations readily take advantage of solutions that emerge spontaneously. When an employee can combine business knowledge with technical understanding, they often bridge a gap between teams. IT specialists know the system but do not always understand the specific demands of everyday tasks. Managers know the objectives but not necessarily the finer details of how the tool works. Users know what they want to achieve but may struggle to describe the technical problem. Someone who understands all three perspectives becomes exceptionally useful.

Being useful is not inherently a bad thing. It can lead to professional development, recognition, and new career opportunities. The trouble begins when the organization benefits from that competence without formally defining the scope of responsibility. Occasional assistance turns into a permanent obligation, while all your other tasks and deadlines remain untouched. This is how a position comes into existence that appears nowhere on the organizational chart but is remarkably visible in the number of unread messages in your inbox.

What makes matters even more complicated is that the role can be psychologically appealing. It feels good to know that other people trust your abilities. The satisfaction of solving a difficult problem can effectively drown out the

question of whether you should have been dealing with it in the first place. Particularly when the results are immediate: someone thanks you, the task moves forward, and the tension subsides. Your own report, which has been waiting three hours to be prepared, offers no such gratitude. It sits quietly, counting on your evening guilt to get the job done.

Recognize the Cost Before It Becomes Normal

You do not need a complicated time-tracking system to determine whether the migration is starting to consume too much of your working day. For several days, make a note of additional activities connected with the tool change. Brief details are enough: what you were trying to accomplish, what prevented you from completing the task, how long the workaround took, and whether the problem was resolved. The objective is not to record every minute with the precision of a space mission control center. You need a picture that allows you to distinguish an isolated inconvenience from a recurring burden.

An example entry might look like this: "Preparing sales summary - no access to historical data - manual export from previous tool - additional 45 minutes - support ticket still open." This description is considerably more useful than declaring that the new system is hopeless. Even if, emotionally speaking, the second statement feels extraordinarily accurate. The first identifies the cause, its impact on your work, and the decision that may be needed.

After a few days, you might discover that the biggest problem is not actually the tool itself. Perhaps you are losing time explaining the same activities to different people. Or perhaps you are constantly switching between tasks because support requests come directly to you. In another situation, you may find that one missing feature is

responsible for most of the delays. Each discovery points toward a different solution. Without understanding the cause, however, it is easy to conclude that you simply need to work faster.

It is important not to turn observation into yet another burden. If documenting problems takes longer than resolving them, your diagnostic tool has started behaving suspiciously like the new company platform. Keep your records limited to information that will help you make a decision or explain the situation to your manager. Five specific examples can be more convincing than thirty pages describing your suffering in exquisite detail.

What to Say When Everything Still Has to Be Ready on Time

Imagine a conversation with your manager, who expects a report by Friday. Instead of replying that you are falling behind because of the migration, you explain the specific relationship between the tasks: "Preparing the report currently requires approximately two additional hours because the data has to be verified manually. I can meet the deadline if we move the supplementary analysis to Monday. If both tasks must still be completed by Friday, I'll need support or a decision to reduce the scope."

This is not a refusal to do your work. It is a presentation of the available options and their consequences. You allow the person responsible for setting priorities to make an informed decision. At the same time, you avoid promising that everything will be fine and then secretly attempting to fulfill that promise at the expense of your personal time. Communicating this way requires courage because it makes limitations visible. Nevertheless, it is more professional than

declaring complete readiness when delivering the work would require unplanned overtime.

Of course, not every manager responds enthusiastically to this kind of information. You may hear that everyone needs to adapt or that the current situation is only temporary. In that case, it is worth returning to the specifics: "I understand that this is a transitional period. I need to establish which task takes priority this week." You do not have to convince the other person that the migration is being badly managed. What you need is a decision about the actual workload.

If no decision is provided, keep a record of the limitations you have raised and proceed according to the agreed priorities, where these are known. For matters critical to safety, customer commitments, or data integrity, follow the applicable escalation procedures. Improvising on your own is not always safe, particularly when a quick workaround would involve bypassing access controls or transferring confidential information into unauthorized tools.

Not Every Difficulty Means Overload

One final distinction is important. An initial slowdown while learning a new system does not necessarily indicate a poorly managed implementation. A new interface takes some getting used to, different names for familiar functions can be irritating, and activities that once happened almost automatically now demand your conscious attention again. This is a normal part of change. There is little point in declaring a crisis every time you spend several minutes looking for a button.

It makes particular sense to talk about overload when additional responsibilities are recurring, have no clearly

designated owner, interfere with your regular work, or require skills and permissions that fall outside your role. Duration also matters. Helping a team once during a system launch is very different from solving the same problems every day for weeks with no prospect of a permanent fix.

You can therefore finish the day with one simple but important observation: which activities genuinely belonged to your job, which involved learning, and which represented invisible support for the migration? You do not need a perfect solution yet. To begin with, it is enough to stop calling three different kinds of workload by the same word: "responsibilities."

Because once you know where your own work ends, it becomes much easier to recognize the moment when someone starts adding the entire implementation to it. Even if they do so with a smile and the reassuring promise that it will only take five minutes.

Chapter 2 - Why the New Tool Was Supposed to Save Time but Is Devouring It Instead

Everything looks elegant during the implementation presentation. The presenter opens the new application, clicks three times, and displays the finished result. The data arranges itself into a clear chart, the report is generated automatically, and the screen looks so aesthetically pleasing that it almost seems a shame to use it for actual work. Participants are told that the new platform will reduce the number of steps required, simplify communication, and allow everyone to focus on activities that create value.

Two weeks later, you attempt to reproduce the same process. Except your data is not sample data, users have different access permissions, and a customer needs additional information that was not included in the demonstration. After seven clicks, you discover that the automatic report is missing one important field. So you export the file, correct it manually, and send it on. The number of operations was supposed to decrease. The only thing that has decreased is your faith in presentations containing the word "seamless."

This experience does not necessarily mean the new tool is worse than its predecessor. Often, it reveals the difference between what a system is capable of doing and the conditions under which people actually use it. Software can offer excellent features and still make work more difficult if the implementation overlooks important dependencies, the data has not been properly prepared, or

users have not been given enough time to change the way they work.

Understanding this distinction matters because it helps you stop interpreting every delay as evidence of your own incompetence. However, there is no need to swing to the opposite extreme and assume that software developers are responsible for absolutely everything. Reality is more complicated. That is precisely why it is worth examining what the cost of change actually consists of.

A System Is More Than a Screen

When a company replaces a tool, the conversation often revolves around features. The new platform offers better reporting, additional automation, more sophisticated permissions, and easier collaboration between teams. Yet everyday work does not consist of using individual features in isolation. It is a sequence of interconnected activities in which one person prepares information, another checks it, a third makes a decision, and a fourth uses the result.

Changing one element of that sequence can affect every other element. A report that was previously generated automatically may now require additional approval. Data that once went directly to a team may need to be exported. Permissions that used to be granted at department level may now be assigned individually. Each change may have a reasonable justification, but together they create an entirely new way of organizing work.

Imagine the process of preparing a monthly forecast. An analyst retrieves the data, a manager checks the assumptions, the finance department verifies the figures, and the document is then submitted for approval. In the old tool, some of the information updated automatically. The

new solution offers more advanced analytical capabilities but requires historical data to be assigned to categories manually. Nobody involved in designing the implementation accounted for the time this activity would require.

At the feature level, the company has gained a more modern platform. At the everyday process level, an additional step has appeared. If the organization measures only the availability of new capabilities, the migration looks successful. If it measures the time needed to prepare the forecast, the picture may be quite different. Both observations can be true simultaneously.

This is why asking "Is the new system better?" is often too general to be useful. A more productive question is: "Which specific tasks can we complete faster with it, which take longer, and why?" Only this kind of analysis makes it possible to separate the advantages of the tool from problems with the way it was introduced.

The Cost of Switching That Nobody Put on a Slide

One of the most exhausting aspects of a migration is the constant need to switch your attention. You are working on a task when an access problem appears, so you look for a solution. You return to the task, but someone asks about a new procedure. You answer, reopen the document, and try to remember where you left off. A moment later, you discover that the data you need is still stored in the previous system.

Some of these interruptions take only a few minutes, but their cost is not limited to the time spent clicking. You have to reconstruct the context, review earlier decisions, and get back into the flow of thinking. This is particularly disruptive during analytical work, tasks requiring concentration, or

activities involving comparisons between multiple pieces of information. Ten short interruptions can leave you feeling busy all day without having finished many of the things that actually matter.

This explains why an employee can feel exhausted despite having what appears to be only a small number of additional responsibilities. It is not just about the total number of minutes. The fragmentation of work matters too. Your calendar shows two hours reserved for preparing an analysis, but in reality, you receive six separate fragments of time, interrupted by technical troubleshooting. Formally, the time still exists. In practice, it resembles cheese cut into slices so thin that making a decent sandwich becomes virtually impossible.

It is therefore worth examining not only how much time problems consume but also when they occur. If questions and support requests arrive throughout the day, you may need an agreed approach to handling them. Some issues can be collected and resolved during one dedicated period of work. Others genuinely require an immediate response. Distinguishing between these situations matters more than desperately trying to answer every message within thirty seconds.

This does not mean ignoring your colleagues. You can inform the team that questions concerning the new tool should be collected in a designated channel, while urgent matters should be clearly marked with an explanation of why they require immediate attention. If providing support is not formally part of your responsibilities, agree on this approach with your manager. Creating an unofficial support shift on your own could simply reinforce the very problem you are trying to reduce.