Genius One Inc.
Smart Solutions for Growth

The **Fresh Eyes Approach:**

How German Efficiency Turned Me Into America's Software Project Rescue Expert

Frank Kanu

They call me the software project lifesaver. The fixer. The guy who turns chaos into order. But I wasn't always this person.

My name is Frank Kanu, and for more than two decades, I've been rescuing failing software projects for Fortune 500 companies, government agencies, and struggling enterprises. I'm the person they call when deadlines are missed, budgets are blown, and everyone is pointing fingers. I'm the one who steps in when the expensive consultants from Accenture, IBM, and Capgemini have failed to deliver.

What makes me different? It's not what you might expect. It's not my technical knowledge—though I have plenty. It's not my experience—though I've seen nearly everything in this industry. It's my background: I'm a German-born former hairdresser with a completely different perspective who asks questions others don't think to ask.

This is the story of how I became the person companies call when failure isn't an option, and the unconventional approach I use to fix what others can't. It's about how being an outsider became my greatest strength, and how my "fresh eyes" approach might just save your next software project.

Chapter 1: **German Beginnings - The Foundation of My Approach**

I was born in Wuppertal, Germany, a city known for having the world's only suspension train system. This engineering marvel—trains that hang from the tracks instead of sitting on them—never experienced a fatal accident until decades after my childhood. This early exposure to efficient German engineering had a lasting impact on me.

My father worked as a translator for Spanish and was fluent in more than a dozen languages. His gift was extraordinary—he could learn enough of a new language in just four weeks to travel to a foreign country and handle the basics. This linguistic talent led to a memorable incident in Spain in the 1950s when someone nearly beat him up because they couldn't believe a non-Spaniard could speak such perfect Spanish.

My original dream was to follow in my father's international footsteps—I wanted to move to France. Life had other plans. My family moved to Duisburg, the heart of Germany's steel industry, where my father worked for a steel mill. Ironically, this early exposure to steel manufacturing would later prove valuable when I began implementing software for steel companies like Nucor.

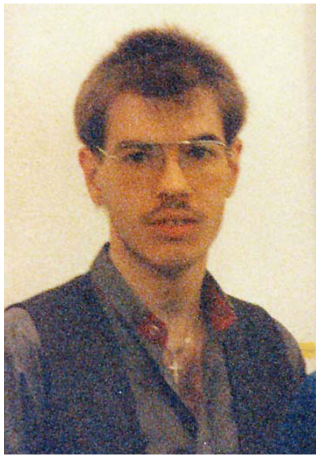
My educational journey wasn't always straightforward. I studied at FernUniversität in Hagen, a correspondence university where all materials were sent by mail, and we formed study groups to work through difficult concepts. In one economics class, which was part of my computer science program, only 10% of students passed the written exam. I was among that 10%. The material was so dense that the binder for just the footnotes was enormous, explaining the high failure rate.

The rigorous German educational system taught me something critical: efficiency matters. When I later began working with American companies, I was shocked by their inefficiency—especially in meetings. This cultural difference would become one of my greatest strengths.

Growing up in Germany in the post-war era instilled in me a certain pragmatism. We weren't focused on theoretical ideals but on what actually worked. This practical approach to problem-solving would later define my career. It's not about what should work in theory; it's about what actually works in practice.

Chapter 2: **The Unexpected Path**

My professional journey took unexpected turns from the beginning. Initially, I had an apprenticeship lined up to work with handicapped people, creating better prosthetics. But the German military had other plans—they drafted me, and that opportunity vanished.



After military service in the special forces as a paratrooper, I needed a job. Hairdressing wasn't my passion, but it taught me something crucial: understanding people's real needs, not just what they say they want. A client might ask for a specific cut, but understanding why they wanted it—perhaps to look younger or more professional—was the key to true satisfaction.

My next position was in sales at one of Germany's largest department stores, working in the music department selling TVs, radios, and audio equipment. Here, I learned another valuable lesson that would serve me well in software rescue: customers would return to buy from me even when competitors had lower prices, simply because I provided better service. My manager once called me back from lunch to help a customer who specifically asked for me.

These seemingly unrelated career experiences gave me something most software experts lack: a holistic understanding of human motivation and needs. In the tech world, we often focus exclusively on technology while ignoring the people using it. My diverse background meant I instinctively approached problems differently.

Looking back, I now see that these early career detours weren't detours at all—they were foundational experiences that would shape my unique approach to software implementation. The skills I developed in these "non-technical" roles—deep listening, understanding customer psychology, and providing exceptional service—would prove just as valuable as any programming language I would later master.

Chapter 3: **Coming to America**

In 1996, I applied to a company from California and was granted an H1B visa to work in the United States. The US Department of Education evaluated my education as postdoctorate level—an interesting designation considering I don't actually hold a doctorate. But titles never mattered much to me; results do.

When I arrived in America in 1997, my English was, as I describe it, "kind of bad." This didn't matter as much as you might think. For my first project at ADP Benefits Services, I was paired with a project manager to develop a benefits calculation engine. During our third meeting, the project manager and I started discussing the engineering specifics of the system, and we lost everyone else in the room. They couldn't follow our technical discussion, but we understood each other perfectly despite my language limitations.

What was supposed to be a six-month contract extended to a full year. The company even brought on three additional programmers from my firm. This pattern would repeat throughout my career: every single contract I worked on in both America and Germany was extended beyond its original timeframe. I never left a project until it was truly complete.

One of my most memorable early experiences was working on a Navy project for parts replacement systems. As project lead, I shocked the Navy managers by completing meetings in 20 minutes that typically took two hours. "What? We're done? These meetings usually take two hours," they said. I replied, "We covered everything we needed to cover. Why waste more time?"

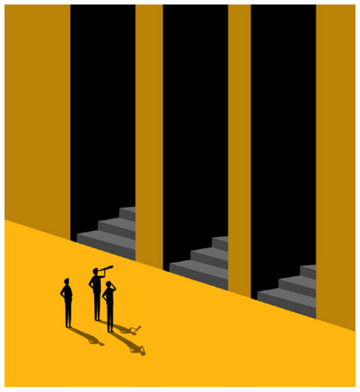
This German efficiency approach permeated everything I did, from meetings to code review to system architecture. Americans often described it as refreshingly direct, though occasionally it ruffled feathers—especially when I questioned established processes.

After ADP and the Navy project, I worked with AI Solutions on satellite steering software called "Free Flyer." This experience with high-stakes, precision software development further refined my approach to software implementation. When you're writing code that has to guide satellites in orbit, there's absolutely no room for error or inefficiency.

My career path continued through Quick Hire (which was later bought by Monster) and then to various independent consulting roles, culminating in the founding of Genius One in 2013. Throughout this journey, each project taught me something new about what makes software implementations succeed or fail.

Chapter 4: **The Fresh Eyes Approach**

My unique approach to rescuing failing software projects crystallized over my first decade in America. I call it the "Fresh Eyes Approach," and it's based on a simple premise: the people closest to a problem are often the least equipped to solve it.



This isn't because they lack intelligence or dedication. It's because they're too familiar with existing processes and constraints. They're trapped within the paradigm that created the problem in the first place.

Take one recent example at a steel mill. They needed to import data but complained the data wasn't importing properly. When I asked if they understood how the import process actually worked, they admitted they didn't. This was a typical scenario I've encountered countless times: the company had moved people from one department to another without proper knowledge transfer. The top sales representative was now the shipping lead without understanding shipping processes. When problems arose, the new lead would try to consult with the former lead, who might now be working as a nurse in the mill or moved to a different facility entirely.

This knowledge gap creates a cycle of confusion and failure. I fill that gap not by knowing more than everyone else, but by asking the questions no one thinks to ask. My outsider perspective—German background, diverse career experiences, and willingness to challenge assumptions—lets me see solutions that remain invisible to those embedded in the organization.

As I often tell clients: nobody would think a former hairdresser would be helpful when you're trying to shoot a rocket into space. But sometimes that completely different perspective is exactly what you need to break through entrenched thinking patterns.

I've found that organizations often become victims of their own expertise. They develop blind spots precisely in the areas where they have the most experience. My value isn't in knowing their business better than they do—it's in seeing their business with fresh eyes, unclouded by years of "that's how we've always done it."

The "Fresh Eyes Approach" also means I don't let anything limit me. I don't accept artificial constraints or assumptions. When a tester at ADP thought 10,000 records was enough for testing, I insisted on using 500,000. When everyone accepts a deadline as immovable, I question whether it's realistic or necessary.

This willingness to challenge everything—even (or especially) the things everyone else takes for granted—is at the core of my rescue methodology.

Chapter 5: **The Question Method That Transforms Projects**

I know it upsets people, but I'm one of those individuals who often answers a question with a question. This habit comes from my conviction that asking the right questions is more powerful than providing immediate answers.

My book, "Stop Telling... Start Leading! The Art of Managing People By Asking Questions," explores this philosophy in depth. The approach stems from the Socratic method of teaching, where seemingly simple questions force people to confront what they don't know.

The most common scenario I encounter is a team focused entirely on an arbitrary deadline: "We've got this deadline, we need to get this done." Nobody stops to ask: "Why do we have to do it in this specific way?" or "Is this deadline even realistic based on current resources?"

One testimonial about my questioning approach that I'm particularly proud of noted that my questions "sound easy and then when you start thinking about it, they get more and more complicated." That's precisely the point—the questions should make you think differently about the problem.

Here's a real example: In one ERP implementation, the company wanted to force one mill to use the same processes as another mill. But one produced steel plates while the other produced coils. It's far easier to load coils on a truck than plates, which have different dimensions and weight distributions. The leadership wasn't considering these fundamental operational differences, focusing instead on standardization for its own sake.

By asking targeted questions about the physical logistics—"How many plates can fit on a standard truck?" "What's the workflow for loading plates versus coils?"—I revealed that the standardization approach would actually reduce efficiency rather than improve it.

Questions expose assumptions. They reveal knowledge gaps. Most importantly, they make people think about problems differently. And different thinking is what rescues failing projects.

This questioning approach can sometimes cause discomfort. In societies where we're taught it's impolite to answer a question with a question, my method can feel challenging. But this discomfort is productive—it forces people out of autopilot thinking and makes them confront the real issues.

Some of my most powerful questions are deceptively simple:

- "Why do you need to do it this way?"
- "What would happen if we didn't meet this deadline?"
- "Who actually uses this feature?"
- "Have you asked the end users what they need?"
- "What are we trying to accomplish, and is this the only way to get there?"

These questions often reveal that what teams think is important isn't what's actually important for project success. They expose misalignments between technical goals and business objectives. And they frequently uncover the real problems that have been lurking beneath the surface all along.

Chapter 6: **Case Studies in Project Rescue**

Throughout my career, I've seen the same patterns of failure repeat across industries and companies. Let me share three case studies that illustrate how my approach turns failing projects around:

Case Study 1: The Dusty AS/400

A major European manufacturer had purchased an expensive AS/400 system at the insistence of upper management. Due to personality conflicts between a mid-manager and the IT manager, the mid-manager told his team to ignore the new system entirely. The expensive technology gathered dust while the IT manager was blamed for the failure.

When upper management learned the new system wasn't being used, they demanded immediate implementation and threatened the IT manager's job. The typical consultant response would have been to force a rapid implementation, likely causing operational chaos.

My solution: Within two months, I created an in-house production planning system that connected the old system with the new one. This satisfied upper management's demand for AS/400 implementation while allowing mid-level staff to continue using the familiar system they preferred. The IT manager kept his job, and the company avoided operational disruption.

The key insight wasn't technical—it was human. I recognized that the conflict wasn't really about technology; it was about workplace politics and personal conflicts. The technical solution had to accommodate these human factors to succeed.

Case Study 2: The Stalled Software Rewrite

A software company with 100 users had spent 10 years developing their system. They needed to rewrite it with a new compiler and user interface. Despite impressive specifications, requirements changed weekly, degrading the software's value.

The project was six months late, over budget, and at a standstill. Project managers refused to discard outdated source code because creating new code seemed too expensive. Office politics and team conflicts had paralyzed progress, and the contractor was about to lose their customers.

My solution: Within one year, I helped push out two releases, secured multiple budget increases, and had every unrealistic deadline removed. Two years later, the contractor was still earning profits on the software with a staff of five working on implementation of new features.

The breakthrough came from getting the team to focus on delivering value rather than adhering to fluctuating specifications. By locking down core requirements and implementing incremental releases, we created momentum that had been missing for months.

In each case, the technical challenges were intertwined with human challenges—politics, communication breakdowns, and conflicting priorities. Addressing only the technical aspects would have failed. My approach succeeds because it addresses both the human and technical dimensions simultaneously.

Chapter 7: **The German Efficiency Framework**

Through decades of rescuing software projects, I've developed what I call the German Efficiency Framework. It consists of five principles that can be applied to any struggling software implementation:

1. Ignore Everything Everyone Else Tells You

The most valuable thing I've learned is to look at problems from a completely different perspective. As I often say, nobody would think a former hairdresser would be valuable in launching a satellite, but that's precisely why my approach works. I ask: "What would you have done with a similar problem as a hairdresser?" or "What would you have done with a similar problem in the steel industry?" Cross-pollinating solutions from unrelated fields often yields breakthroughs.

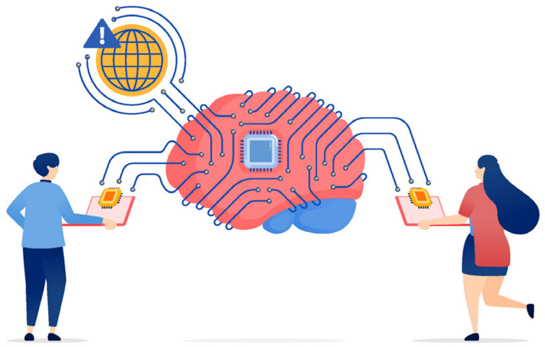
When someone tells me, "This is how we've always done it," I immediately know I've found the source of their problem. The most dangerous phrase in business isn't "We can't afford it" or "It won't work"—it's "We've always done it this way." Breaking free from established patterns of thinking is essential for solving problems that established thinking has failed to solve.

2. Test With Real-World Data Volumes

I once worked with a tester who believed 10,000 records was sufficient for testing. I insisted on using 500,000 records, which initially angered him because his "select * from *" queries became very slow. When ADP later onboarded their second customer with over one million records, our robust testing meant the system handled the load without issues. Never underestimate the volume of data your system will eventually need to process.

This principle applies beyond just database performance. Test with real users, real workflows, and realistic timeframes. Many projects fail because they're tested under idealized conditions that bear little resemblance to how the system will actually be used. When you test under realistic conditions, you discover problems early, when they're still fixable.

3. Focus On People, Not Just Technology



In every failing project I've encountered, the root problem isn't primarily technical—it's human. You must figure out what makes people tick. Once you understand that, you have two choices: either work with them by leveraging their motivations, or ensure they're no longer part of the project.

I learned this principle from my best friend in Germany, who was a financial director for a Crown Cork and Seal subsidiary. When they fired him, they had to replace him with three additional people because everyone under him had willingly worked overtime, unpaid. The new manager couldn't inspire the same dedication. Understanding human motivation is more powerful than any technology.

On one Navy project, an engineer wasn't happy that a non-US citizen was brought in as his lead. He started working against me and was removed from the project within two days—at the Navy's insistence. They recognized that his resistance would jeopardize the entire project, regardless of his technical skills.

4. Cut Meeting Times by 80%

The original standup meeting time in Agile methodologies was 15 minutes. I typically complete them in less than 2 minutes. Meetings expand to fill available time, and most of that time is wasted on office politics and discussion of irrelevant details. Set an agenda, stick to it, and end when you've accomplished your goals—not when the calendar says you should.

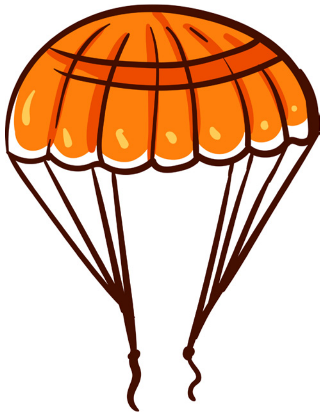
At Nucor, I've implemented this principle with striking results. Teams that previously spent hours in meetings now accomplish more in a fraction of the time. This isn't just about efficiency—it's about respect for people's time and focus. Every minute spent in an unnecessary meeting is a minute not spent solving real problems.

5. Ask What They Really Need, Not What They Say They Want

Many software projects fail because they're built to specifications that don't actually serve users' needs. At one Crown Cork and Seal subsidiary, management had no idea what their users struggled with or what would help them work more efficiently. Only by directly engaging with users—asking questions about their daily challenges and observing their workflows—could I determine what the system actually needed to provide.

This principle echoes my experience as a hairdresser. Clients would ask for a specific cut, but understanding why they wanted it was the key to delivering what they truly needed. In software, users might request a feature without realizing there's a better way to achieve their underlying goal. By focusing on the problem they're trying to solve rather than their proposed solution, you often discover more elegant and effective approaches.

Chapter 8: **Recognizing When Your Project Needs Rescue**

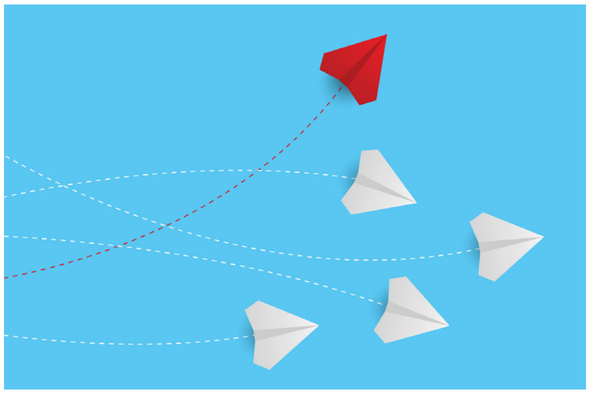


Before I conclude, let me share some warning signs that your project might need rescue:

1. **Your scope keeps expanding but your deadline doesn't.** This is a recipe for failure. Either the deadline must move, or the scope must be trimmed.
2. **Team members avoid giving direct answers about progress.** When simple questions like "Is this on track?" yield complex, evasive responses, something's wrong.
3. **Testing is postponed or scaled back to "save time."** This actually guarantees you'll lose time later dealing with issues that could have been caught early.
4. **Documentation is treated as an afterthought.** Without proper documentation, knowledge lives only in team members' heads—making your project vulnerable to turnover.
5. **Meetings focus on assigning blame rather than solving problems.** When the culture shifts from collaborative to defensive, your project is already in trouble.
6. **Users aren't involved until launch.** If the first time users see the system is after it's built, expect major rework.
7. **Management overrides technical concerns.** When business leaders dismiss technical warnings from their teams, disaster usually follows.

If you recognize three or more of these warning signs in your project, it's time to consider bringing in fresh eyes before things deteriorate further.

Conclusion: **The Value of Being Different**



Throughout my decades-long career in software implementation, my greatest asset has never been technical knowledge—although I have plenty. It's been my willingness to approach problems differently, to question assumptions, and to bring a completely different perspective to the table.

My "Fresh Eyes Approach" works because I'm not tied to industry conventions or organizational politics. I'm the outsider who can see solutions that remain invisible to those inside the system.

As someone who has gone from hairdresser to satellite software developer, from German military to American corporate consultant, I've learned that conventional paths don't always lead to the best outcomes. Sometimes the detour—the unexpected perspective—is precisely what's needed to break through seemingly insurmountable challenges.

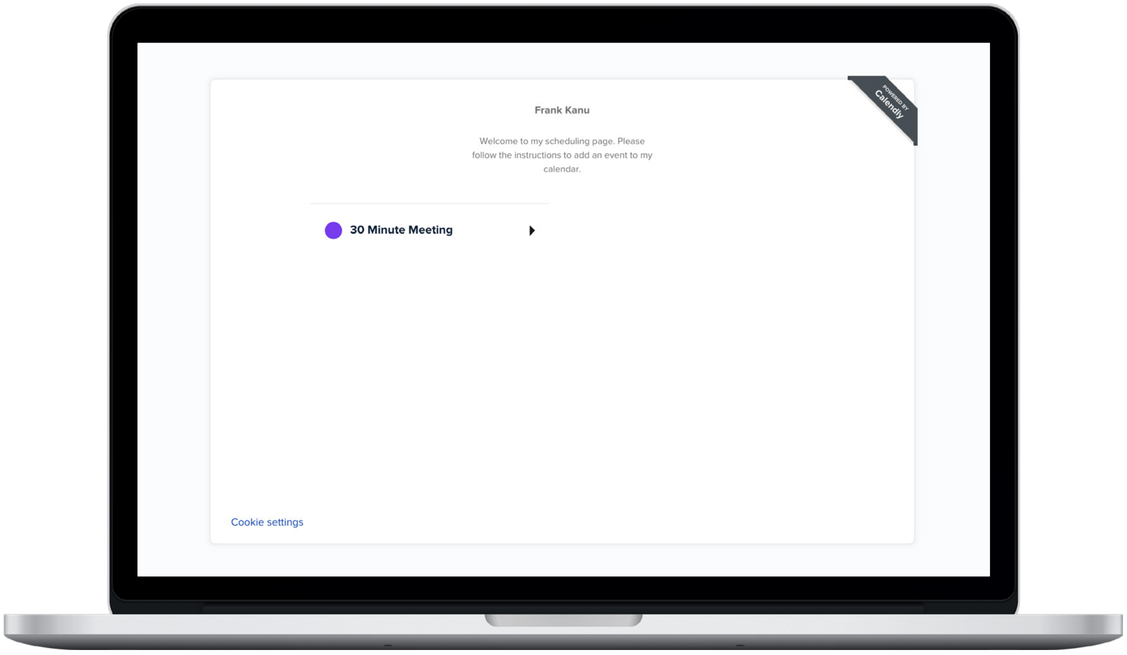
If your software project is struggling—if you're facing delays, budget overruns, or functionality issues—the solution might not be more programming resources or extended deadlines. It might be a completely different perspective.

This is what I bring to every client engagement: German efficiency, question-based leadership, and the fresh eyes that can transform a failing project into a success story. The faster I work myself out of a job, the better I've done my job.

When projects fail, you know where to find me.

Ready for a Fresh Perspective?

Schedule Your Free Consultation here: https://calendly.com/frank-kanu



Frank Kanu

Welcome to my scheduling page. Please follow the instructions to add an event to my calendar.

30 Minute Meeting

Cookie settings

About the Author

Frank Kanu, founder of Genius One, Inc., is a renowned technical troubleshooter who specializes in rescuing failing software projects and optimizing underperforming systems for large enterprises. With his signature "fresh eyes" approach and German efficiency, Frank has helped manufacturing firms, Fortune 500 companies, and organizations undergoing complex system integrations achieve remarkable results—including a 500% improvement in operational efficiency and reducing release rollbacks from multiple yearly occurrences to less than one every five years, saving clients $20 million.

A masterful software development strategist with particular expertise in ERP/MES implementation, Frank is known for his no-nonsense leadership style that cuts through bureaucracy to deliver measurable outcomes. His cross-industry insight allows him to apply innovative solutions from one sector to solve problems in another, while his talent for asking the right questions and understanding others' perspectives has earned him high praise from executives, entrepreneurs, and tech industry leaders.