

The Time Machine

Back in July 2000, I was working on a custom Customer Relationship Management (CRM) system for one of the nation's largest telecommunications companies. We were adding customers at a rapid clip, with no signs of slowing. I had worked on the system in various capacities for a number of years. The system kept record of every customer of the company, and if a customer wanted to change any aspect of their account, it had to come through the system. Change a phone number? CRM. Change address? CRM. Add a feature? CRM. Ask a billing question? CRM. There were 10,000 representatives who used the software every day.

The problem? It wasn't working, and hadn't for several weeks. The system was partitioned by geography, and the largest partition (New York), which held 20% of the system's customers, would cease to function under peak load—meaning weekdays from 9 a.m. to 7 p.m., Eastern time.

This was particularly surprising because up until then, things had seemed to be working fine. One of our technicians was concerned about some statistical anomalies he was seeing, and we had occasional flare-ups in the database that would resolve themselves. But the business, by and large, was working fine.

The main technical symptom was that the primary database server would start to exhibit contention, where different processes would be fighting for control of the same resources. So some of the processes would have to wait, which increased CPU consumption until there was none left, which would make everything slower, causing the problems to snowball until all processes were at a standstill.

At the time, I had only been in IT for six years and was in a state of panic. I was the manager of a team intended to optimize availability and performance. We had made modest improvements in a few parts of the system but were not prepared to tackle this. Yet somehow we had to.

My routine was to come in early, around five a.m., and manually monitor the system. It would break down in pretty much the same fashion every

weekday at around nine. Throughout the day, we would try to minimize the impact by curtailing nonessential activities on the system, and we'd get a daily tongue-lashing (most of it in a stern New York accent) from the business, which was utterly and understandably frustrated. In the afternoon, we'd plot what changes to make to the system to improve things, and after the system was brought down at night, we'd apply those changes and hope for the best. I usually stuck around to see if the changes went in as planned.

Personally, I was not faring well through all of this, not that anyone was. In addition to managing a team that was positioned to help fix things, I felt doubly on the hook because I had helped to create the system in the first place. I felt like a failure. To say I

The Business

Your IT department likely doesn't exist for its own sake. It's there to support some business objective. We often refer to the people in your company who are not in IT, yet who are trying to achieve business goals, as **"the business."**

was stressing out was an understatement—despite being extremely tired, I wasn't sleeping well, and I was having stomach problems to boot. (I noticed more than a few people chewing on antacids those days.) And I wasn't a lot of fun to be around, either.

Throughout this iterative process of monitoring and making changes, I was grasping for answers. I figured there had to be a better way, not only to resolve this problem, but also to stay out of this type of situation in

the first place. I asked the experts brought in to help us. I asked my management. I searched the web. I went to the bookstore and looked for something relevant. And largely I came up empty and disappointed.

After several more weeks, and after making dozens, if not hundreds, of changes (both technical and procedural) to resolve the problem, we got back slightly above water (though it took nearly two years to get everything fully stabilized). At first, we kept searching for the one thing that was busted and could be fixed to bring things immediately back to normal. Unfortunately, this silver bullet didn't exist, and, as it turns out, it rarely does. The turnaround in this case required a lot of trial and error, and crazy amounts of heroism from folks throughout the enterprise.

Despite everyone's best efforts, a lot of damage had been done to the company as well. It cost the company hundreds of thousands of "hard dollars," from overstaffing and overtime in the call centers, as well as incalculable "soft dollars," such as the opportunity costs of not having the IT teams focus on more valuable initiatives, low employee morale, and damage to the brand through poor customer experiences.

Does this seem like a freak occurrence—something that could never happen to you? Unfortunately, these events are not all that uncommon. It would be easy to dismiss this type of thing as the perfect storm, but all you have to do is search the Internet for news stories about massive systems failures, and you will see numerous examples.

In the end, we had all learned the hard way how to do some things properly and, conversely, how not to do some other things. I mused that if I could have traveled back in time to give myself some advice in early July 2000, or better yet earlier that year, things would likely have gone much, much better. And I vowed at that time to do two things:

1. Not to let it ever happen again.
2. To help out anyone else as much as possible if they were in the same predicament.

That is the impetus for this book. I cannot create a legitimate time machine. But I have learned a lot working in this field the last six years. My coauthors, Mike and Christophe, have built up a lot of quality experience as well, based, in part, on similar crises. Combined, we have worked on dozens of different applications of different types, and over time, learned how to reliably keep IT services in a healthy state so that they rarely suffer crises. Time and again, platform to platform, business to business, there are certain methods that just work. We call this book “The Opposite of Luck” because the only way to effectively manage IT service quality is to utilize a systematic and comprehensive approach. Anything else is just a well-intentioned roll of the dice.

If I’d received that advice from my future self back in July 2000, I would have known that no matter how dire the circumstances, good crisis management techniques (as described in chapter 1) can help you make the best of a bad situation, and at minimum avoid making a bad situation worse. I would have known what the four cornerstones of production quality are (chapter 2) and what key measurements I should have been collecting (chapter 3) to avoid being caught by surprise when I least expected it. I would have also known how to turn my problems into opportunities (chapter 4), and how to quickly find the root cause of my ailing system’s problems (chapter 5). Even more importantly, I would have been able to avoid these problems if I had known about some fundamental rules of prevention (chapter 6). I would have also greatly benefited from better ways of envisioning information (chapter 7) and innovative ways of greatly expanding my influence to make organization forces work for me (chapter 8). I would have known the power of “the invisible hand of IT” (chapter 9), and how a new way of organizing my team (chapter 10) could have turned this monumental task into business as usual. I’d even have known how to prioritize all of the above to make the best use of my limited free time (chapter 11).

IT Service? Application? System?

Throughout the book we will use these terms according to context. In general, the content is aimed at improving the quality of **IT Services**, which has to do more with the overall experience of the users and value obtained by the business than with the underlying technologies. At times, however, content will relate to these underlying technologies. In these cases we will use terms with the following meanings:

- **Application:** a business software program
- **System:** either a host server or a combination of hardware and software

These techniques work for us, and they can work for you, too. Most of the advice is versatile, for anyone from a junior IT manager up to the CIO of a large shop. It ranges from tactical to strategic, technical to managerial, and throws in a healthy dose of organizational and political as well. Many of our observations are IT-specific, while others may apply more broadly to other aspects of management. You might find some of the insights to be obvious, while others may open your eyes for the first time. Those are the most fun.

Bottom line: we want to help you learn techniques that will prevent your business from feeling the same pain that ours did. I suppose it's a time machine substitute ("I can't believe it's not a time machine!"). I wish you well and hope that this helps you get on the path to making your own luck.

Chris Oleson, March 2009