The Burning Bag of Dung— and Other Environmental Antipatterns

Phillip Laplante, Penn State University

In my youth a favorite prank of the local delinquents (being a geek, I was not one of them) was to place a paper bag full of doggy doo on a neighbor's porch, light it on fire, ring the doorbell, and then flee. The homeowner, upon answering the door, had no choice but to stomp out the incendiary feces, getting their shoes dirty in the process. Why this scatological anecdote? Because it is a metaphor for work situations in which things have gotten so bad that the only way to "put the fire out" is to step into it. I call this the "burning bag of dung" antipattern.

In their groundbreaking book, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis* (John J. Wiley and Sons, 1998), William Brown, Raphael Malveaux, Hays "Skip" McCormick, and Thomas Mowbray describe a litany of problems that repeatedly occur in software architecture, design, and project management. They also describe solutions or refactorings for these situations. Providing such a taxonomy helps to quickly identify problem situations, provides a playbook for addressing the problems, and yes, offers some comic relief to the people in these situations.

Extending their taxonomy to organizations leads to another kind of antipattern, the "environmental antipattern" (we might also call these "cultural" or "situational" antipatterns). This is where the "burning bag of dung" antipattern fits.

Environmental antipatterns differ from the other types because they are somewhat harder to pin on a specific individual or practice. Instead, the antipatterns are due to a set of static and dynamic internal environmental conditions that create a toxic atmosphere. Moreover, the environmental antipatterns are often harder to detect than the other antipatterns because they become so pervasive that they may be hidden. You know a bad software architecture when you see one. You can identify a bad software design. But it is not always easy to characterize a negative culture. Usually, it's described as: "This place sucks."

WHAT ARE ANTIPATTERNS?

To software engineers a pattern is a named problem-solution pair that is used to enable large-scale reuse of software designs and architectures. Based on the work of

And you think

YOU HAVE PROBLEMS?

architect Chris Alexander, who sought repeatability in the design of buildings when one-of-a-kind

designs were being used, patterns explicitly capture expert knowledge and make this expertise more widely available. There are numerous sets of patterns ("pattern languages") for software design, analysis, management, and so on.

Shortly after the emergence of patterns for design, analysis, management, and so forth, practitioners began discussing problem-solution pairs in which the conventional solution does more harm than good, known as "antipatterns."

My colleague Colin Neill and I have identified a number of environmental antipatterns (which will appear in our forthcoming book, *Antipatterns for IT*, Auerbach Press, 2005), and we keep adding to this list. I'll introduce a sample of these antipatterns here, describe a typical scenario in the software development or IT world, and offer a refactoring to the situation. In some cases, there may be no clean way to put out the fire. In other cases, courage may solve the problem with minimal collateral damage—for example, by confronting long-festering personnel problems. Please note that fleeing the situation is always a potential refactoring, and often the most desirable one. In any case, by describing these antipatterns, I hope that you will become more adept at identifying a similar situation and taking appropriate action.

BURNING BAG OF DUNG

This antipattern has already been described in humorous metaphor. In the real sense, it describes any negative work environment left by one or more fleeing managers. In fact, a sequence often involves passing the "bag of dung" from one manager to another—talk about a "hot potato"! The situation fits precisely when every obvious remedy to the negative situation involves some sort of traumatic suffering—for example, layoffs.

In the software world, the flaming doggy-doo bag can consist of software development practices (or lack thereof) that are so thoroughly embedded in the

Continued on page 78

Continued from page 80

company that retraining would be futile (e.g., an “agile” shop where every malpractice is explained away as part of the agile manifesto). It can also appear as a culture of project management involving the misuse of metrics that leads to paranoia—you can’t suggest that metrics not be used, but you can’t get folks to use them properly, either.

In other cases, the fire alarm has to be sounded—the rest of the organization has to help out. This is a plausible solution strategy because it may be that the previous manager was left to deal with the burning bag of dung alone because he was disliked. In the situations described before, for example, repairing the abuse of agile practices can be done only by proselytizing the correct application of methods. This takes great courage; if the conditions are right, there can be collective reform. If you are a senior manager, you can force retraining, or you can restructure your team through hiring and firing. The same kind of refactoring might work for an environment where metrics are being accidentally or maliciously misused.

FOUNDERITIS

In this antipattern, the founder of a company has difficulty letting go when growth exceeds his or her skills. The founder still wants to control everything as in the days when the company was a one-person shop. The reason why founderitis and its effects are so pervasive is that the skills needed to launch a company and make it successful are very different from those needed to run even a medium-size or public company.

Although this antipattern is attributable to a single individual (or small group of founders), the effects are so profound and so tightly woven into the fabric of the company that it is environmentally transforming. The effects are pervasive micromanagement from the founder on down and a stifling climate. I have seen many organizations from a few million to billions of dollars in revenues suffering from founderitis (no, I don’t mean Microsoft; although the founder is deeply involved, clearly, this is a healthy organization with a unique culture). For example, I know of two \$100-plus million software companies, each with more than 200 employees, where the founder still codes and personally approves the code changes made by anyone else.

Most sophisticated venture capitalists recognize founderitis and will remove the founder of even a successful start-up once it reaches a certain level of revenue. Other refactorings include: developing a published succession plan; getting a highly respected person to talk with the founder about the negative culture; learning to accept the

situation; and leaving the company. The latter refactoring is the most likely solution—often, even the sons of founders leave because they can’t survive the environment. And leaving the company avoids the ugly situation of having to confront the founder.

SHOELESS CHILDREN

This antipattern is related to the fable of the shoemaker’s children who have no shoes because their father is too busy scratching out a living. In the software and IT world, the shoeless children antipattern is analogous to the company denying itself equipment or self-service because all its resources are focused on product development, delivery, and customer service. In the extreme, a certain software company might be denying itself adequate software and hardware resources, even though it is providing “state-of-the-art” solutions to its customers. It’s an environmental antipattern because going barefoot is often based on the embedded cultural practice of penny-pinching.

The refactorings depend on the cause of the shoeless children environment. In some cases, the situation may be related to founderitis, suggesting a set of refactorings. In other cases, the antipattern may arise from poor management that leads to bad cash flow and necessary deprivation of internal resources. In this case the solution is strategic budget cutting that excludes essential infrastructure, which is never fun or easy.

WORSHIPING THE GOLDEN CALF

In any environment where there is poor vision or leadership, it is often convenient to lay one’s hopes on a technology or a methodology about which little is known, thereby providing a hope for some miracle. Since no one really understands the technology, methodology, or practice, it is difficult to dismiss. This is an environmental antipattern because it is based on a collective suspension of disbelief and greed, which couldn’t be sustained by one or a few individuals embracing the ridiculous.

To a certain extent, the dot-com business model was a golden calf (aka, “The Emperor’s New Clothes”). Some failing businesses (and even nonfailing ones) quickly switched to this model in the hopes of salvation or quick rewards. A collective belief that changing the base programming language or development environment will cure problems that are related to a bad software architecture or management culture is another example of praying to the gilded veal.

This phenomenon is also related to the “hockey stick” revenue projection—after recording n months of declining profits, with some hand-waving justification, the

analyst projects a sudden, dramatic upward trend, which resembles a hockey stick. Everyone wants to believe such a miracle is possible.

The refactoring for a golden calf environment is to expose the calf for what it is. This will require significant courage and effort on the part of the exposers, as those who believe will cling to any thin hope—even in the face of incontrovertible evidence to the contrary. Recall the famous cartoon with two professors at the blackboard with “and then a miracle occurs” written between two equations. The skeptical professor says, “I think you have to be more explicit in step two.” Likening the current golden calf to previous exposed ones can also be helpful.

CONCLUSIONS

This list of situational antipatterns is simply representative of some of the most common environmental situations that can indicate sick organizations. While, in retrospect, the solution to each problem seems obvious, this is the case only when the problem is correctly identified. Moreover, applying the refactoring will require courage, strength, and wisdom.

Unfortunately, when dealing with a flaming bag of dung, there is no way to avoid getting burnt or stepping

in you-know-what. Sadly, the best refactoring is often to flee, when institutional inertia is difficult to overcome. In any case, I hope that this subset of environmental antipatterns will lead you to correctly identify your own situation, and hence solution. Q

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

PHILLIP LAPLANTE, Ph.D., is associate professor of software engineering at the Penn State Great Valley School of Graduate Studies. His research interests include realtime and embedded systems, image processing, and software requirements engineering. He has written numerous papers, 20 books, and cofounded the journal, *Real-Time Imaging*. He edits the CRC Press Series on image processing and is on the editorial boards of four journals. Laplante received his B.S. in computer science, M.Eng. in electrical engineering, and Ph.D. in computer science from Stevens Institute of Technology, and an M.B.A. from the University of Colorado. He is a senior member of the IEEE, a member of ACM and the International Society for Optical Engineering (SPIE), and a registered professional engineer in Pennsylvania.

© 2004 ACM 1542-7730/04/1000 \$5.00

4 out of 5 of Fortune Magazine's most profitable companies purchased dtSearch developer or multi-user licenses in the past two years.

dtSearch®

Instantly Search Gigabytes of Text Across a PC, Network, Intranet or Internet Site

Publish Large Document Collections to the Web or to CD/DVD

- ◆ over two dozen indexed, unindexed, fielded & full-text search options
- ◆ **highlights hits** in HTML, XML & PDF while displaying embedded **links**, formatting & **images**
- ◆ converts other file types (word processor, database, spreadsheet, email, ZIP, Unicode, etc.) to HTML for display with **highlighted hits**

dtSearch
DESKTOP with Spider
\$199
"Industrial-strength... superb"—PC Magazine

dtSearch
Text Retrieval ENGINE
for Win & .NET for Linux
"Industrial-strength... superb"—PC Magazine

dtSearch
WEB with Spider
from \$999
"Industrial-strength... superb"—PC Magazine

dtSearch
PUBLISH for CD/DVDs
from \$2,500
"Industrial-strength... superb"—PC Magazine

dtSearch
NETWORK with Spider
from \$800
"Industrial-strength... superb"—PC Magazine

1-800-IT-FINDS
sales@dtsearch.com

See www.dtsearch.com for:
◆ hundreds of developer case studies & reviews
◆ fully-functional evaluations

The Smart Choice for Text Retrieval® since 1991

dtSearch Reviews...

- ◆ "The most powerful document search tool on the market" — *Wired Magazine*
- ◆ "Intuitive and austere ... a superb search tool" — *PC World*
- ◆ "Blindingly fast" — *Computer Forensics: Incident Response Essentials*
- ◆ "A powerful arsenal of search tools" — *The New York Times*
- ◆ "Covers all data sources ... powerful Web-based engines" — *eWEEK*
- ◆ "Searches at blazing speeds" — *Computer Reseller News Test Center*