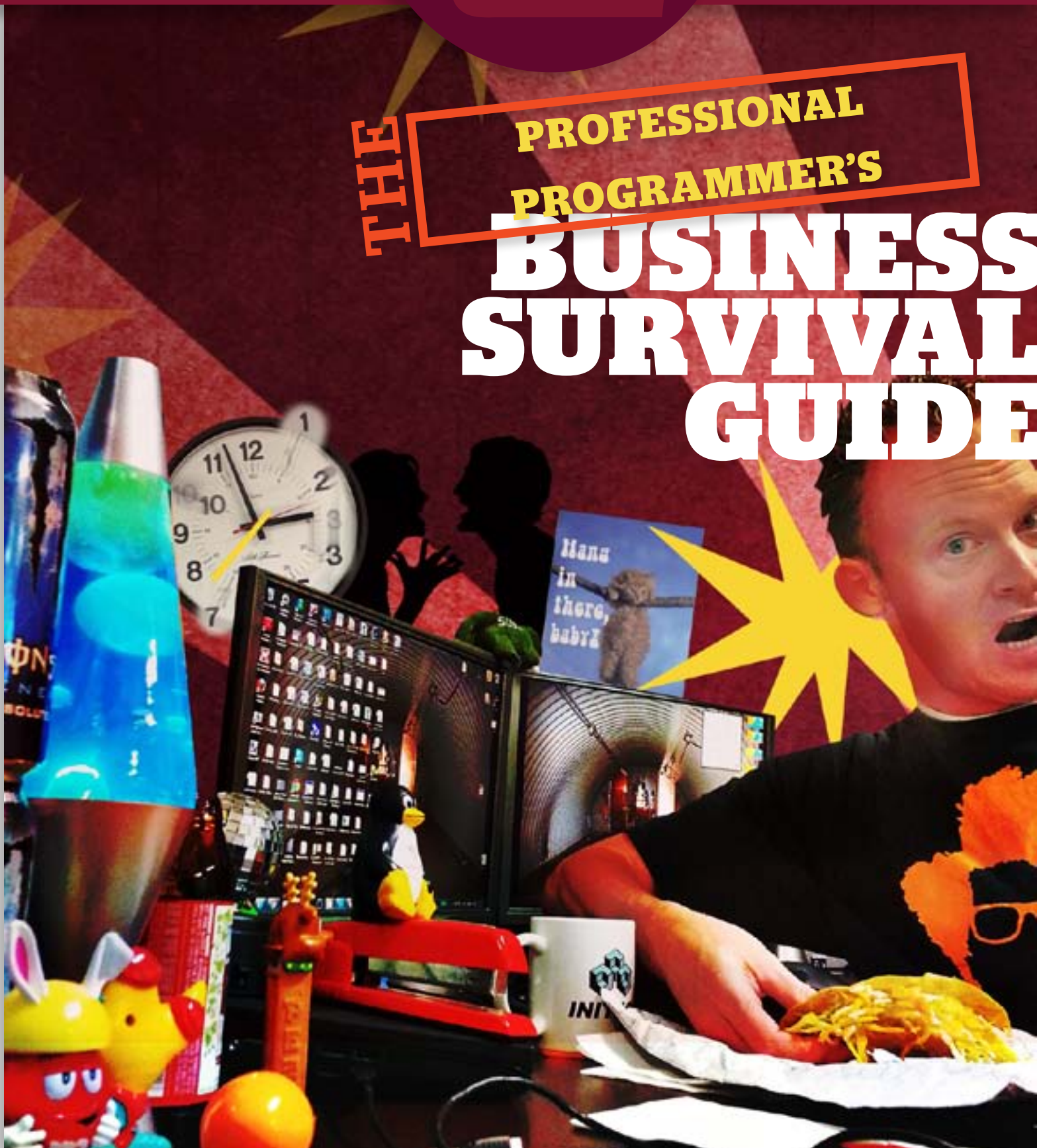
InfoWorld DeepDive

THE PROFESSIONAL PROGRAMMER'S BUSINESS SURVIVAL GUIDE



FLICKR/KEVIN

Deep Dive

How to salvage a (nearly) hopeless software project

Faulty foundations, AWOL contractors, bugs piling up – here's what to do before taking a sledgehammer to a faltering pile of code BY PAUL HELTZEL

Like a carpenter called in to salvage a home repair gone wrong, developers who've been around the block are used to seeing a handful of the same problems. The code gets creaky; bug reports file at an ever-increasing clip; the time spent maintaining the project surpasses any ability to add features to it. At a certain point, the question arises: Can you rehab the code, or should you scrap it and rebuild from the ground up?

We talked with seasoned pros for insights on how they have addressed the most common types of software projects on the brink: Projects with runaway costs, poorly architected projects, ones that simply no longer work.

FLICKR/KEVIN

Deep Dive

Not every software project kicks off with a bulletproof plan and a crew of fabled "10x developers" to execute it. Instead, you often get a kind of "homeowner's special."

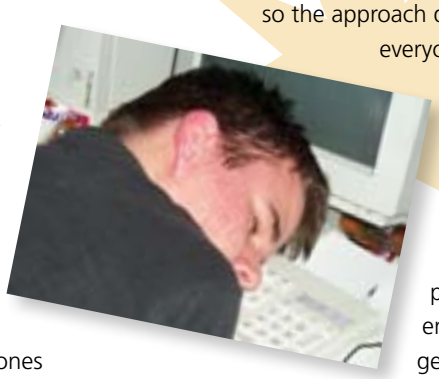
- **Daniel Jacobson,** vice president of edge engineering at Netflix, where he was brought on to lead the API team
- **Stefan Estrada**, an engineering manager at Verizon who oversees a team of five developers working on the OnCue streaming TV service
- **Dave Sweeton,** chief technologist of Stout Systems, a consultancy often called on to take over or repair projects gone awry at Fortune 500 firms, among others

Now let's take a look at our fixer-uppers.

The patch-up job

Not every software project kicks off with a bulletproof plan and a crew of fabled "10x developers" to execute it. Instead, you often get a kind of "homeowner's special" -- a pile of code cobbled together to solve an internal need using the skills of those on hand. In these cases, success can be a curse. Over time, this DIY effort, essential now for business purposes, becomes a tangled mess of bolt-on code and superfluous frameworks, and the only way forward is to bring in an experienced crew and hope for the best.

"There could be many culprits in a scenario like this," says Sweeton, of Stout Systems. "A common one is some fundamental flaw at the architectural level, so the framework of the house doesn't support what you're trying to build. That can be too little framework or too much framework -- too much is actually more common than you'd think."



If you're lucky and the bones are good, you may not have to tear down the whole endeavor and start anew.

"This is where the buzzword refactoring comes into play," Sweeton says, adding that the first step in refactoring code is to return to its requirements.

"I would understand the requirements, then review the code to see if it's meeting those requirements -- and unstated ones like quality and maintainability," Sweeton says. "If there are architectural problems, then sort out what it really should be and figure out the way to get the right architecture in place."

This doesn't always mean rip and replace. As with a house, when you run across problems with code, there are ways to improve the overall situation instead of merely fixing the previous owners' mistakes. Make incremental fixes and every time the code needs work, add an enhancement -- right the wrongs, Sweeton says. And refactor, refactor, refactor.

If it's a project that's only used internally instead of a consumer product, you have a better chance of sticking with it rather than scrapping it.

"But if it's being used by millions of people," Verizon's Estrada says, "then little changes can make a big difference in terms of cost. If it's really creating a bad user experience or costing a lot of money, then there's probably a better way. It's time to start over."

The accidental duplex

This common software project nightmare highlights the importance of strong vision and leadership. Stakeholders are assembled, and the squabbling begins. Users, managers, and engineers can't agree on how to go forward, so the approach devolves into trying to please everyone, with the project manager gathering requirements from everyone in the building. By the time the project gets handed off to engineering the big picture is lost.

"This is a common problem," Estrada says. "What ends up happening is that you get requirements from almost everyone, then nobody knows what you're trying to accomplish or what should be done first. Of course it cascades into more problems and more problems."

In other instances, executives and project managers can get caught up in note-taking -- priorities and process -- rather than finding the best overall solution.

In any case, soon enough you have the equivalent of three kitchens purpose-built for prepping specific meals, two bathrooms right

FLICKR/KEVIN/LUCHOEDO

Deep Dive

next to another, and a garage no one can get into. You set out in pursuit of a unified solution and ended up with myriad pet projects housed under the same roof. The result satisfies nobody.

Before you can even think of what to do with the code on this kind of rehab job, you have to go back to square one -- this time, say our pros, with vision. There has to be one blueprint that everyone agrees is the best plan.

"You have to have a foundation in terms of the architecture to make sure it accomplishes what you are trying to do," Estrada says. "You have to get input from all the stakeholders, but someone from management on down has to drive that vision, or you get really bad architecture in terms of the software design."

With the new unified plan in place, you can return to the code with purpose and better prioritize how to fix the mess.

"When prioritizing, you're basically saying: 'Which things are we going to do today or this week?" says Netflix's Jacobson. "The right question is: 'Do we care about this at all?' And if we do, then make it happen -- and if we don't, then make it gone."

> "The right question is: **'Do we care about this at all?'** And if we do, then make it happen -- and if we don't, then make it gone."
> **DANIEL JACOBSON**

The castaway

It's not uncommon for a software project to simply be left unfinished, with the developer AWOL. Perhaps the core has been sketched out, a few features have been implemented, but the contractor, dev shop, or employee who launched the project simply left one day, never to return. The code works, barely or not at all. Or perhaps the project was completed but lacked a maintenance plan and is now falling into disrepair. What next?

> "If one person created the whole thing, the problem may be the manager, **And if you're the manager, you've done a poor job."**
> **DANIEL JACOBSON**

First, the bad news: If you did the hiring, there's not much point in finger-pointing.

"If one person created the whole thing, the problem may be the manager," Jacobson says. "And if you're the manager, you've done a poor job."

All of our pros offer the same tip: Avoid it in the first place. One way to plan for this contingency is to cross-train the team to ensure more than one person understands the source code.

"Let's say the team working the project has six developers," Estrada says. "One is working on some of the functionality for online payments, another person is doing personal account management or presenting data to the user. You switch them over. Rather than assign the person who built the original functionality, get a different person to fix problems or create new features. They get exposure to that area and the source code. If the person working on account management gets lost in some foreign trip and never comes back, somebody else can still pick it up without much struggle."

If consultants are creating the code, Sweeton says it's still important to get multiple developers in the loop: "Engaging a lone consultant has inherent risk if you don't have direct-hire developers on your staff. Make a direct hire or engage a consulting company that approaches the development project with resource redundancy."

The money pit

The bills for this makeover are so extreme it would make a spouse -- or accounts payable -- flip out. The architecture had problems from the start, and predictably, boatloads of cash didn't solve them. How do you stop throwing good money after bad?

Sometimes the instinct is to prioritize and dig your way out. But then you end up with a long list where everything -- and nothing -- is urgent. Time goes by, more issues are reported, and the result is a backlog that would take years to resolve.

"It becomes a game of tactical cat and mouse," says Jacobson. "You're not attacking

Deep Dive

anything with real focus."

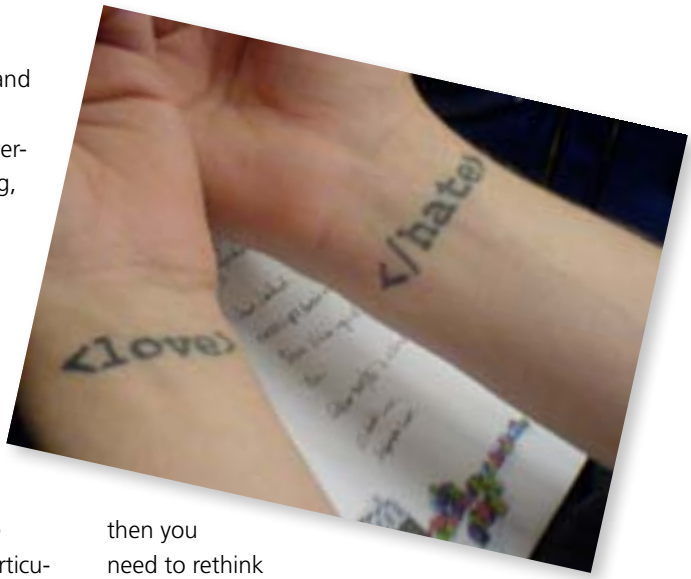
Decide what's important, say our pros, and don't get lost in gradients of importance.

Jacobson explains: "I was having a conversation with one of my peers who was saying, 'We need to add this to the priority list because it keeps getting pushed down and not executed.' And I said, 'It's clearly not important; let's not add it to the priority list. If it were important, we'd be doing it. Or if it is important, then we have to stop doing everything else.'"

In some cases, managers see an outside firm as the solution to fix the problem, but consultants can compound the errors if the software's purpose isn't clearly articulated by those doing the hiring. The company keeps throwing money at the problem -- when the solution isn't about cost.

"The consultants will do what they're hired for," says Estrada, "but because there's not a clear vision as to what they're supposed to be doing -- say, they're adding more features, but that might not be the end goal of what the company is trying to achieve. You need a good set of program managers who can communicate between all the groups to make sure the right functionality is being created."

In some cases, managers see an outside firm as the solution to fix the problem, but **consultants can compound the errors** if the software's purpose isn't clearly articulated by those doing the hiring.



The total teardown

Sometimes a rehab job isn't a rehab job at all. The foundation has obvious cracks, and everybody can see them. The architecture is so poorly designed it needs to be scrapped and started anew. But how do you know when it's time to toss the entire project? Why not stick with code you have and exterminate the bugs?

"Here's the litmus test: If you're spending more time on bugs than you are on functionality, then you need to rethink the implementation and start over," says Verizon's Estrada. "If it's a complete patchwork that needs constant maintenance, then you're wasting a lot of time."

It's a daunting task to throw out the old and start anew. Not everybody is going to be happy about it. If a demolition is necessary, our pros say communicating that effectively to the various workgroups is the first step.

"Within the first couple of months of being at Netflix," Jacobson recalls, "I basically said, 'This application that all of you are using, we're going to throw that away. We're going to go with a new approach.'"

But ditching the system immediately isn't usually an option. In Jacobson's project, too many legacy devices counted on the existing platform and it would take time to transition to the new tools.

"We said we're going to invest in a fundamentally different model," Jacobson says, "a more optimized model that is not one-size-fits-all -- and it's going to be at some cost to my teams to execute and to the other teams who have to consume from it rather than from the previous one. We need to take a chance here and have confidence in ourselves and go for it. That's what we did. And it worked."

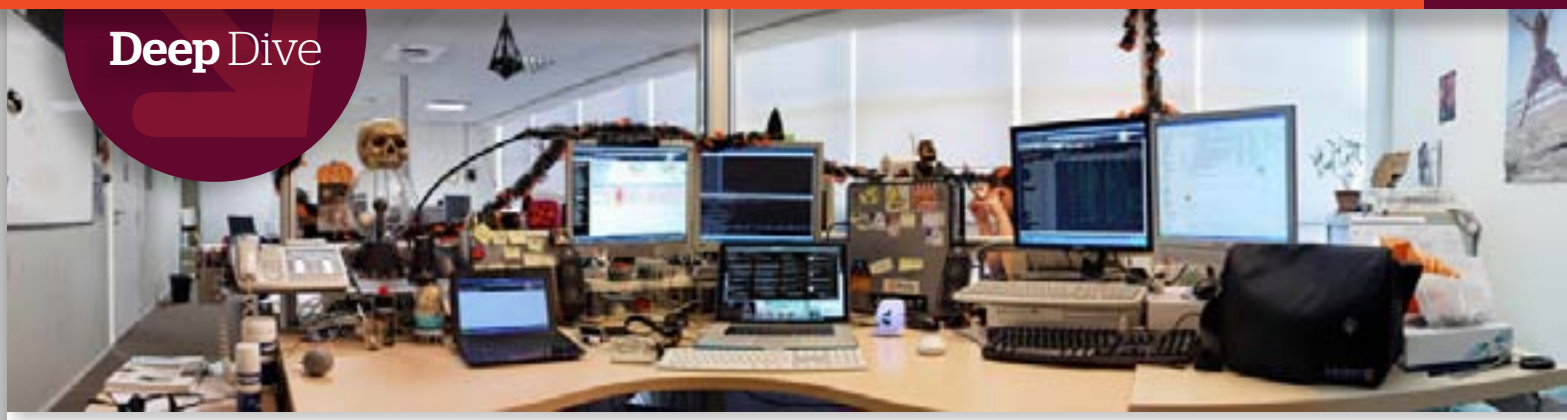
"**Here's the litmus test: If you're spending more time on bugs than you are on functionality,** then you need to rethink the implementation and start over."
STEFAN ESTRADA

The creaking new construction

New construction always leads to some settling. The pipes rattle. The beams creak. But when it keeps happening, you may start to wonder: Is this normal, or does it need repair? A wise

FLICKR/KEVIN/FRANCIS STORR

Deep Dive



FLICKR/ARTHUR CARANTA

product manager once said about newly launched code: "All new babies cry." She was letting users know the system wasn't broken -- it would eventually settle down.

"Parents learn the difference between cries," says Sweeton. "Some are best left alone. Others are the genuine article distress call. Run -- don't walk -- to handle. The same is true in software development."

Determine the minimum viable product, Sweeton says, and focus on that. While you're nailing it down, figure out what can wait. "Maintain a list of all other items that will need attention, but defer them until after the first release."

You have to be clear-eyed, says Jacobson. Look at the reality of the situation and assemble the evidence. Then make your recommendation. Here, metrics for assessing the health of the project -- and the cost of healing it -- is key.

"It's not just that I have this great idea, or I didn't build it, so I want to build my own thing. Say: 'We've talked to this number of developers, we're churning this number of development hours fighting against this system, we're exposing greater risk to our availability because we're handling this number of requests. We're actually producing complexity and bugginess,'" Jacobson says.

If the current model is faltering under its own weight, he adds, it's time to create a stronger and healthier option for the future. "Whatever those metrics are -- those data points that are exposing the weakness of the current system -- they're fuel to say, 'This needs to go away.'"

> "**Parents learn the difference between cries. Some are best left alone.** Others are the genuine article distress call."
> **DAVE SWEETON**

The retrofit

Increasingly common is the case in which the foundation is solid, but the techniques that produced it are dated. The technology used may have been the right choice at one time but no longer. Maybe too many dependencies are no longer supported. Or it relies on old tech that few people know how to support anymore. Can you prop up this project and keep it going, or is it time to call in the wrecking ball?

Estrada recalls working on an app where there was a push to use HTML5, which seemed like the latest and best choice. But as the project went on inefficiencies became apparent -- it wasn't ready for prime time yet.

"We redid the project in C++, and it takes a lot more effort because C++ is a much lower level," Estrada said. "But we achieved better performance. If you create your platform correctly, you develop a process that makes it easy to add new features."

Consultants might be a good fit here, says Estrada, where you have an isolated technology that needs updating. But there's a caveat: He argues that bringing in consultants is typically a bad choice to fix an internal tool in need of new features. If the tool is tied into other systems, consultants won't have the big picture or a long-term investment in the outcome.

"Business process should drive the software," says Sweeton. "Sort out the right business process, then adapt the code to it."

If Sweeton's sentiments suggest a theme, there's good reason for it. Runaway projects tend to be missing one of three elements: a clear vision from management, project managers who communicate effectively, and strong tech leadership. Projects that don't have all three? They tend to need renovation on more than one level. ■

Deep Dive

14 NIGHTMARE CLIENTS

and how to defang them

Here's how to identify and neutralize beastly client behavior before it gets the best of your project

STEVEN A. LOWE

THE KEY TO SURVIVAL

in a client-based profession like software development is recognizing the signals of a project heading south despite your best efforts. Here, difficult clients can be a clear impediment to your success.

In fact, anyone who has spent time working directly with clients has probably come across a few that reminded them of mythological beasts. Maybe the project kicked off well but took a wrong turn into a bureaucratic fog, only to die a slow death. Maybe a powerful executive suddenly rose up from behind the scenes to kill your project midway through. Maybe an unreasonable demand late in the game burned to a crisp your chances of ever getting paid.

Here are 14 nightmare clients you may very well encounter on your quest for success as an independent software developer. May you have strength in recognizing, avoiding, and neutralizing them, when possible. Please feel free to add your own in the comments below.

FLICKR/JEFFREY

Deep Dive

KRAKEN: FLICKR/URBAN BAMBOO, STIRGE: RFLOWERS

NIGHTMARE CLIENT NO. 1

The Kraken

Like the legendary sea monster for which this type of nightmare client is named, the Kraken surfaces suddenly, often in the middle of a project, to ensnare you in its multiple tentacles, also known as conflicting requirements.

On the one hand, you want the client to be satisfied and get the system they want. On the other hand, conflicting requirements are difficult to resolve. On the other other hand, more requirements means more billable work. On the other other other hand, unresolved conflicts are likely to cause the project to fail. On the other other other other hand -- you get the picture.

The Kraken sees no conflict among its multiple arms; after all, more often than not, each arm comes from a different department that's unaware of the others. Plus, the Kraken is so large that it is oblivious to the fact that its thrashing is breaking the mast and crushing your crew.

The instant you recognize that one arm doesn't know what the other arm is doing, call an all-hands meeting. Only by enumerating and exposing the conflicting requirements can you tame the Kraken. You may find that in a group discussion some of the conflicts dissolve, or you may have to work a little harder to disentangle what would otherwise end up destroying your project.

NIGHTMARE CLIENT NO. 2

Stirges

Much like its mythical, blood-sucking, mosquito-size namesake, the Stirge does little damage on its own -- a little prick that barely sets you back, leaving you to wonder whether anything untoward happened at all. But a swarm of Stirges is serious trouble. If you don't swat Stirges early, your project will soon be left bloodless.

The conversation starts innocently enough: The client has an "idea," and it sounds feasible, at least in theory, so you ask a question. In response to your question, the client calls in someone to help. This person sort of answers your question and brings up another possibility -- slightly conflicting, but perhaps resolvable.

You ask another question, and another person is brought in to help ... then another and another, each with more requests and suggestions and conflicts, generating more questions, and calling in more people. Before you know it, everyone is squawking conflicting, impossible requirements at once, and you're starting to feel agoraphobic.

Limit the number of people brought in to "help" in a conversation to only a few, especially if each person adds more confusion instead of answering your questions. At the point where you start to feel threatened, fake a phone call and exit -- before the trautonium music starts.

Cauterizing each head after you sever it is the key to preventing more from growing back.

NIGHTMARE CLIENT NO. 3

The Hydra

The Hydra is a pernicious little nightmare of a client: Cut off one of its multiple heads, and two more grow back. The only way to defeat it is to cut off all its heads at once.

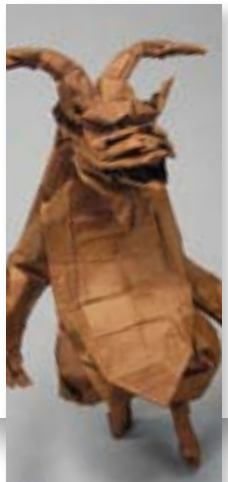
Hydras make themselves known in one of two ways: as a never-ending stream of request and conditions before commitment, if you're lucky, or a never-ending stream of also-must-haves and changes on a fixed-bid contract with insufficient boundaries for completion.

Cauterizing each head after you sever it is the key to preventing more from growing back. While I cannot condone carrying a torch into a business meeting, the next best thing is a contract with specific deliverables and a precise process for handling changes.

NIGHTMARE CLIENT NO. 4

The Minotaur

Half-man, half-bull, and hidden at the center of a complex labyrinth, the Minotaur is a powerful executive who remains behind the scenes, unknown, until you are close to commitment, then suddenly appears to throw the entire project into disarray.



There is no known defense against this creature, which is often armed by a pathological ego that cannot be assuaged, but you may be tipped off to its existence by oblique references made by others: "That won't get past Jim," or "Let's wait to loop in Jim before he puts the project off track." Any attempts on your part to involve "Jim" early the process will be met with terrified stares.

If "Jim" signs the checks, pull the plug on the project; you are doomed.

NIGHTMARE CLIENT NO. 5

The Dragon

This may come as a surprise, but Dragons do not make good clients. They hoard their gold, burn consultants to a crisp on a whim, and are

HYDRA: WIKIMEDIA, MINOTAUR: FLICKR/GERWIN STURM, DRAGON: FLICKR/VILLE MIETTEN

Deep Dive

generally disagreeable. And if you're looking for a knight in shining armor to slay them -- you're on your own. Dragons are quite vain and may be susceptible to flattery, but they are also treacherous and unlikely to keep any promises they make. It's much easier for them to devour you.

You can recognize a Dragon by its unreasonable demands, by the way it is never satisfied, by its reluctance to pay, by the fire that spews forth from its mouth and the terrible rending of its claws when it is upset. You most likely will not have to end a contract with a Dragon; the Dragon will fire you.

The best way to avoid a Dragon is to talk to those who worked with a client before you. Recognize the behavior patterns above, note the scorched earth and bones strewn about the entrance to its lair, give the job a pass, and live to code another day.

NIGHTMARE CLIENT NO. 6

The Unicorn

You may not think it, but the Unicorn is a nightmare. It manifests as a perfect opportunity … that never happens. You get a tantalizing glimpse, and theoretically the Unicorn can be ridden, but mostly it appears and disappears, and you end up exerting an exorbitant amount of effort trying to track it down and tame it. If you get close, getting impaled on its horn is an option, but not a good one.

If an opportunity seems too good to be true, if it appears and disappears at random, if it can be glimpsed only at a distance, if you must complete quest after quest to prove that you are virtuous enough to deserve the project -- it's a Unicorn.

Move along, nothing to see here!

NIGHTMARE CLIENT NO. 7

The Werewolf

Most clients aren't monsters. They come across as friendly, and they collaborate well. But every once in a while, a client suddenly turns into a ravenous beast that tears you asunder. This is the Werewolf.

Worse, when the Werewolf changes back, it doesn't remember anything and doesn't believe you -- assuming you survived the assault. In the software world, it's not only the full moon that triggers this change. Questioning some sacred cow technology or pointing out an obvious less-than-ideal business practice can result in a sudden, gutting transformation.

You can survive a Werewolf client if you know what sets the Werewolf off. Prior consultants may tip you off: "Don't try to talk with him during month-end closing." "He's fine to work with unless you question his devotion to spreadsheets."

UNICORN: FLICKR/ZOOMAN. WEREWOLF: FLICKR/NATHAN RUPERT

Deep Dive

The Sphinx is difficult to spot in advance, but if you hear (or overhear) comments like "pop the Pez dispenser, here's another one," then you may be heading for mysterious disaster.

NIGHTMARE CLIENT NO. 8

The Sphinx

The Sphinx confronts developers and consultants with riddles and devours those who do not answer to its liking (not unlike the interviewers at certain pretentious tech companies). The riddles may not be obvious; they may, in fact, appear to be ordinary business problems, like employee morale, poor production, quality problems, and excessive turnover. The "wrong" answer will get you shredded -- and as is often the case with the Sphinx in the software world, there may not even be a right answer!

The Sphinx is difficult to spot in advance, but if you hear (or overhear) comments like "third consultant this year" or "pop the Pez dispenser, here's another one," then you may be heading for mysterious disaster. If you think you may be engaging with a Sphinx, get paid in advance.

NIGHTMARE CLIENT NO. 9

The Undead

In the software world, there are many kinds of Undead, with varying appearances and deleterious abilities. Some seem like normal, even charming people, then suddenly turn on you when you're vulnerable and suck the lifeblood out of you or your work. Others are nebulous, noncorporeal entities that pass through walls and other barriers, haunting your project and scaring your developers. No matter what form they take, Undead clients are particularly frightening because they are relentless, unstoppable, and worst of all: No one believes you when you talk about them.

Mythological undead can be defeated by different means depending on their kind -- a stake through the heart for a vampire, an exorcism for an evil spirit, removing the head of a zombie, and so on -- but in the tech world, the safest recourse is to fire them and find new paying work.

NIGHTMARE CLIENT NO. 10

The Troll

Trolls lurk in isolated places and spring up on the unsuspecting to attack them. Anyone who has spent anytime in a comments section on the Internet is familiar with the tech world equivalent.

As a client, a Troll takes offense at everything you say, no matter how you say it, even if you are quoting their own words and agreeing with them. The Troll delights in extended email conversations, hijacking the original intent into netherworlds of pedantically misconstrued arguments -- because they think it's fun or because they have a technical hobbyhorse they can't get past.

You can't get anywhere with a Troll. You cannot trick them into the light of your wisdom as an experienced tech pro and consultant. You must do as the wisdom of the Internet tells you: Don't feed the Trolls; ignore them.

If the Troll is your client, fire them. If the Troll simply works for your client, exclude them from conversations. Cite their disruptive behavior to your client, and inform the client that the Troll will henceforth be excluded from all conversations.

It's not pretty, but if you're not careful, the Troll may try to eat you.

SPHINX: FLICKR/IRMAN, UNDEAD: FLICKR/VINCENT ANDERLUCCI, TROLL: FLICKR/DOUG WILDMAN

Deep Dive

NIGHTMARE CLIENT NO. 11

The Siren

Sirens are known for their beauty and enthralling songs, leading sailors off-course to be dashed against the rocks and reefs. It is rare to have a client that is an actual Siren, but it is not so rare for them to sing Siren songs -- with much the same effect. (Developers are also known to do this to themselves, by getting distracted by shiny new tech instead of finishing the job at hand.)

A Siren client may offer irresistible temptations, typically the promise of a lot more work in the future in exchange for something "simple" done today, for free. Chances are, the things they want today for free are not simple, and the future work will never materialize. Do not be tempted; instead, plug your ears and avert your eyes, and stay on course!

NIGHTMARE CLIENT NO. 12

The Ogre

Ogre clients have the unfortunate habit of killing ideas and projects out of reflex. They can't help themselves! Whether they are allergic to new ideas, are secretly committed to failure, or simply hate you, it doesn't matter -- you won't get anywhere with an Ogre.

If your client is behaving like an Ogre, preventing progress or constantly killing your fledgling efforts in favor of new directions, your only option may be to call for assistance from above. This may be done with a move as simple as sending a positively worded sent to the Ogre and the Ogre's boss, outlining some of your proposals or endeavors that have been devoured (don't say "devoured" in your email -- ogres are notoriously bad-tempered) and asking for help in understanding what they really want to achieve and how. Otherwise, you're wasting your time and theirs on a fairy tale.

Don't be surprised if the Ogre fires you; they're touchy like that.

NIGHTMARE CLIENT NO. 13

THE WILL-O-WISP

Will-o-wisps offer a guiding light through the foggy marsh when you need it most. But as you approach it, the light seems to move away, as if beckoning you to follow. Thinking you are saved, under the spell of a Will-o-Wisp you wander further into the marsh, straight to your doom.

Ever have a client like this? First, they create the fog: conflicting, vague requirements; multiple bosses and sign-offs; a series of free RFPs (requests for proposals) or low-paying POC (proof-of-concept) projects. But you never get a firm commitment, never a clear target. The end result is a slow, struggling death in a dark marsh.

Most clients don't intend to be Will-o-Wisps, but they end up that way, largely because they are unable to commit to or declare a goal or path. Many are rendered timid by technology. Others are simply wandering around in a fog themselves, so bound by their own bureaucracy that they cannot escape. But you don't have to join them on their death march! Whatever you do, don't get dragged into the muck and fog with them.

OGRE: FLICKR/IAN. WILL-O-WISP: FLICKR/MAY

Deep Dive

WIKIMEDIA

Has your client spouted morally ambiguous principles, eternally emphasized results-now and don't-care-how approaches with zero remorse or responsibility for cleaning up the resulting mess? **If so, you might be working for Dr. Frankenstein and unwittingly building a Monster.**

Intentional Will-o-Wisps are more nefarious. Often they simply use you for information because they already have another vendor in mind or are window-shopping. The way to escape is to escalate. Recognize you are talking with the wrong person, and you need be referred to the person who has actual authority, vision, and purpose in the organization. Chances are it is not this person's boss, but the boss's boss's boss, maybe even higher up the chain. It's OK if your contact is unwilling to go with you on your escalation; you must climb a tree, get above the fog, and find firm ground, or you will both fail. Do nothing, and you both stay lost.

NIGHTMARE CLIENT NO. 14

Frankenstein's Monster

Frankenstein's Monster was misunderstood. Cobbled together out of pieces and bits of others, he tried to be good, but failed -- disastrously.

One encounters Frankenstein's Monster not so much in client form, but as the emergent result of a client who places no value on consistent methods, clean code, testing, refactoring, user experience, and so forth. Dr. Frankenstein was the real monster in that story, and that is what to watch out for in a client.

Has your client asked you to take questionable shortcuts? Has your client spouted morally ambiguous principles, eternally emphasized results-now and don't-care-how approaches with zero remorse or responsibility for cleaning up the resulting mess? If so, you might be working for Dr. Frankenstein and unwittingly building a Monster.

You, as an ethical software professional, must make provisions in your work to keep the code clean. Emergency fixes and features will happen, but don't simply fire and forget -- when negotiating to fix or create these things (which will, over time, add enough cruft to your code base to bring the Monster to life) you must include time and provisions for refactoring, testing, and when appropriate, reanalyzing and redesigning the offending/affected subsystems. ■

Steven A. Lowe *is a consultant, software developer, inventor, entrepreneur, author, musician, and lover of puns. He ran an innovative custom software development company for nearly a decade before joining ThoughtWorks as a Principal Consultant in 2014. He admits to being the author of "From Burnout to Bonfire" and a willing participant in the band Noise in the Basement, but neither confirms nor denies being the science-fiction author Steven Ayel.*