How Do You Tell a Client Their Baby Is Ugly?

Choosing the wrong development partner can lead to poor design, technical debt, and costly rebuilds. So, what happens when you have to tell a client the solution they invested in needs to be rethought?

At Trinity, we often hear from clients whose existing software solutions are in distress.

Maybe the application has become painfully slow as the business has grown. Maybe it was never fully implemented. Maybe years of quick fixes have made even simple changes difficult. Or maybe the solution was built in a way that simply doesn't support what the business needs today.

Whatever the reason, these conversations can be difficult.

Clients have often invested significant time, money, and energy into their applications. Their teams may have spent months or even years building processes around them. So, when we come in and identify fundamental problems with the design or architecture, we're not just critiquing a piece of software. We're talking about something they've invested in and relied on.

In that sense, an application can feel a little like someone's baby.

And nobody wants to be told their baby is ugly.

But avoiding the conversation doesn't make the underlying problems disappear. Poor design decisions, the wrong development approach, or an inexperienced partner can create technical debt that becomes more expensive and difficult to address over time.

Sometimes the right answer is a targeted fix. Other times, the best path forward is rebuilding part, or all, of the solution.

The challenge is to know the difference.

There may not be one universal "correct" way to build an application, but there **is** a right way to build it for the organization, its users, and its long-term goals. That requires

understanding the business first, designing with the future in mind, and choosing a development partner who knows how to turn those requirements into a solution that can grow with you.

So, how do you recognize when an application has gone off track? How do you have that conversation without dismissing the work that has already been done? And, most importantly, how do you avoid ending up in the same position again?

That's what we're going to explore.

The Questions We Need to Answer

When an application isn't working as intended, the technology is only part of the challenge. There are also some difficult conversations and decisions ahead:

1. **How do you give honest, critical feedback without alienating the client?**
2. **What can happen when you choose the wrong development partner?**
3. **Is there really a "correct" way to build an application?**

How Do You Give Critical Feedback Without Alienating the Client?

When a client comes to us with an application in distress, our first job isn't to point out everything that went wrong. It's to understand how they got there.

Lead With Empathy and Acknowledge the Investment

By the time a client reaches out for help, they've often invested a significant amount of time, money, and energy into their solution. They may have spent months working with another development team, training employees, changing processes, and trying to make the application work.

Recognizing that investment matters.

It's also important to acknowledge how difficult it can be to admit that something you invested in isn't working. It takes humility to stop, reassess the situation, and potentially revisit decisions that have already consumed significant resources.

That deserves empathy, not an "I told you so."

A conversation might start with something as simple as:

"You've invested a lot into getting this solution to where it is today, and we recognize that this isn't an easy position to be in. The important thing now is understanding what's working, what isn't, and what the best path forward looks like."

The goal isn't to make the client feel worse about past decisions. It's to help them make better decisions about what comes next.

Focus on the Opportunity, Not Just the Problem

There's a big difference between saying, **"This was built incorrectly,"** and saying, **"Here's what's preventing the application from doing what you need it to do."**

Critical feedback is much easier to hear when it's connected to a business outcome.

Instead of simply identifying flaws, explain what fixing them could accomplish: better performance, easier maintenance, improved adoption, greater visibility, or the ability to scale without constantly adding workarounds.

The conversation becomes less about assigning blame and more about answering a much more useful question:

How do we make this better?

Show Them Why

Clients shouldn't have to take your word for it.

If you're recommending a significant change, or even a rebuild, you should be able to clearly explain why. Walk through the underlying issue, show how it affects the application today, and explain what could happen if it isn't addressed.

Sometimes an analogy helps.

Think of an application like a house. You can repaint the walls, replace the flooring, and renovate the kitchen, but if the foundation isn't strong enough to support the structure, cosmetic improvements will only get you so far.

The same is true with software. Sometimes a few targeted improvements are enough. Other times, continuing to patch the existing solution will ultimately cost more than addressing the underlying problem.

Be a Partner in What Comes Next

Pointing out problems is easy. Helping solve them is what matters.

Once the issues are understood, the conversation should quickly turn toward the path forward. What can be salvaged? What needs to change? What should be rebuilt? And how can you make sure the organization doesn't end up in the same position again?

The client came to you because they need help moving forward… not a postmortem on everything that went wrong.

What Happens When You Choose the Wrong Development Partner?

Choosing a development partner is a bigger decision than it may initially seem.

A partner can deliver an application that technically "works" while still leaving the business with serious long-term problems.

Poor design decisions can make an application difficult for employees to use. Shortcuts taken early in development can create technical debt that makes every future change slower and more expensive. An application that wasn't designed for growth may perform well initially but struggle as more users, data, workflows, and integrations are added.

And then there's the cost.

Businesses can spend significant amounts trying to fix a poorly designed solution; sometimes investing even more than they spent building it in the first place. Teams lose time working around limitations. Employees become frustrated. Leadership loses confidence in the system. Eventually, the organization may realize that continuing to patch the application no longer makes sense.

At that point, a rebuild may be the most practical option.

That's why choosing the right partner isn't just about finding someone who knows how to build an application. You need a partner who understands your business, asks the right questions, thinks beyond the initial launch, and designs a solution that can continue evolving with you.

Is There a "Correct" Way to Build an App?

Not exactly.

There isn't one perfect architecture, platform, or development methodology that works for every organization.

But there **is** a correct way to approach building one.

Understand the Business Before Building the Technology

Development shouldn't start with features. It should start with understanding the problem.

Who will use the application? What are they trying to accomplish? Where does the current process break down? What systems need to communicate? What information does leadership need? And what could change as the organization grows?

If you don't understand those things first, you risk building a technically impressive application that doesn't solve the business problem.

Design for Today Without Ignoring Tomorrow

An application should solve the immediate need without creating tomorrow's problem.

That means thinking about scalability, security, integrations, data structure, maintainability, and future changes from the beginning.

You don't need to build every feature you might someday need. But you should avoid decisions that make those future needs unnecessarily difficult.

Keep Users at the Center

An application only creates value if people actually use it.

The people doing the work should have a voice in how the solution is designed. Their feedback can uncover bottlenecks, unnecessary steps, and real-world requirements that may never appear in a technical specification.

The best solution isn't always the one with the most features. Often, it's the one that makes someone's job noticeably easier.

Communicate Throughout the Process

Clients shouldn't disappear after kickoff and reappear when the application is finished.

Regular communication, feedback, testing, and iteration keep everyone aligned and allow problems to surface while they're still relatively easy to fix.

It also prevents one of the most frustrating outcomes in software development: spending months building something only to discover that it isn't what the business needed.

Treat the Application as Something That Will Evolve

Businesses change. Processes change. Technology changes.

A successful application should be able to change with them.

Rather than treating launch day as the finish line, organizations should continue evaluating performance, listening to users, reviewing business needs, and improving the solution over time.

That's how an application goes from being another piece of software to becoming something the business can genuinely rely on.

Sometimes, You Have to Have the Difficult Conversation

Telling a client that their "baby is ugly" isn't really about criticizing their application.

It's about being willing to have an honest conversation when something isn't working, and doing it with empathy for everything the client has already invested in.

There may be parts of the existing solution worth keeping. There may be problems that can be corrected without starting over. And sometimes, the most responsible recommendation is to rebuild.

The important thing is having a partner willing to tell you the truth about which situation you're in.

At Trinity, we believe good development starts long before anyone begins building. It starts with understanding the business, asking the right questions, and designing a solution around what the organization needs.

And when an existing application has gone off track, our role isn't to criticize how you got there.

It's to help you figure out where to go next and build it better this time.

For more information about Trinity and a demonstration of our capabilities, contact Nathan Hise at nathan.hise@trinityis.com or (540) 850 – 6987.