



The Four Reasons Software Modernizations Fail

(and Twelve Strategies for Success)



EXECUTIVE SUMMARY

A modern, intuitive customer experience that is delivered digitally is a prerequisite for maintaining brand/service relevance, growing customer relationships, and acquiring new ones. We all demand user-friendly technology in every transaction. The financial services companies that provide the best digital customer experiences will earn the loyalty of their customers and attract others fed up with complex, outdated, and ineffective websites and mobile apps.



5%

Retention

=

25%+

Profit Increase



According to a study conducted by [Bain and Company](#), a financial services firm that increases its customer retention by just 5% will increase its profits by 25% or more.

EXECUTIVE SUMMARY (CONTINUED)

Modernizing or replacing legacy customer-facing software is easier said than done. Poorly formulated strategy, design flaws, and unchecked technical complexity can make the success of a large development initiative a remote possibility, even before the first line of code is written.

The key to success is a strong modernization foundation built on three pillars: strategy, design, and technology. Within each pillar are four strategies that build upon one another to ensure success.



PILLAR	STRATEGY	DESIGN	TECHNOLOGY
FIRST	Clear Problem Statement	Application Documentation	Evolutionary Architecture
SECOND	Objectives	Usage Context	Software Delivery Pipeline
THIRD	Phased Roadmap	Design Objectives	Reusable API Layer
FOURTH	ROI Analysis of Phase 1	User Experience Framework	User Interface Framework

You can do it! We can guide you.

Background

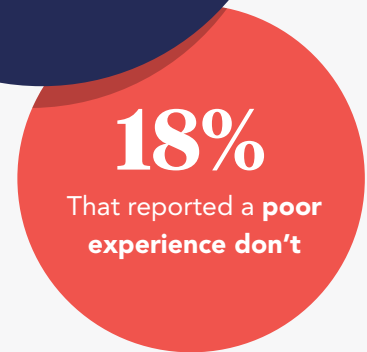


Does Customer Experience Matter?

The advantages of delivering a great customer experience (CX) have never been greater. In a recent survey of 10,000 US consumers, 87% that reported a very good customer experience said they plan to repurchase from the same company, compared to just 18% that reported a poor experience.

This is, of course, obvious. When you have a great experience with a company, your loyalty to its brand increases. This is particularly true in the case of financial services, where the products are complex and where customer experience has become one of the few opportunities available to many firms to differentiate through service.

The message is simple and clear. Good customer experience pays off. According to a study conducted by Bain and Company, a financial services firm that increases its customer retention by just 5% will increase its profits by 25% or more.



Does Your Software Support Customer Experience?

Increasingly, customer experiences are being delivered by websites and mobile apps. However, in a survey of 171 organizations only 28% believe they deliver a 'good' or 'very good' customer experience through their website and only 25% believe the same about their mobile app.



25%

Believe they deliver a **good customer experience** through their **mobile app**



28%

Believe they deliver a **good customer experience** through their **website**

It's all in the Delivery...

Why are established companies struggling so mightily to deliver a great digital customer experience? The answer for many mid-market financial services firms (and perhaps for you) is that they are caught between a rock and a hard place, or more specifically, an irresistible force and an immovable object.

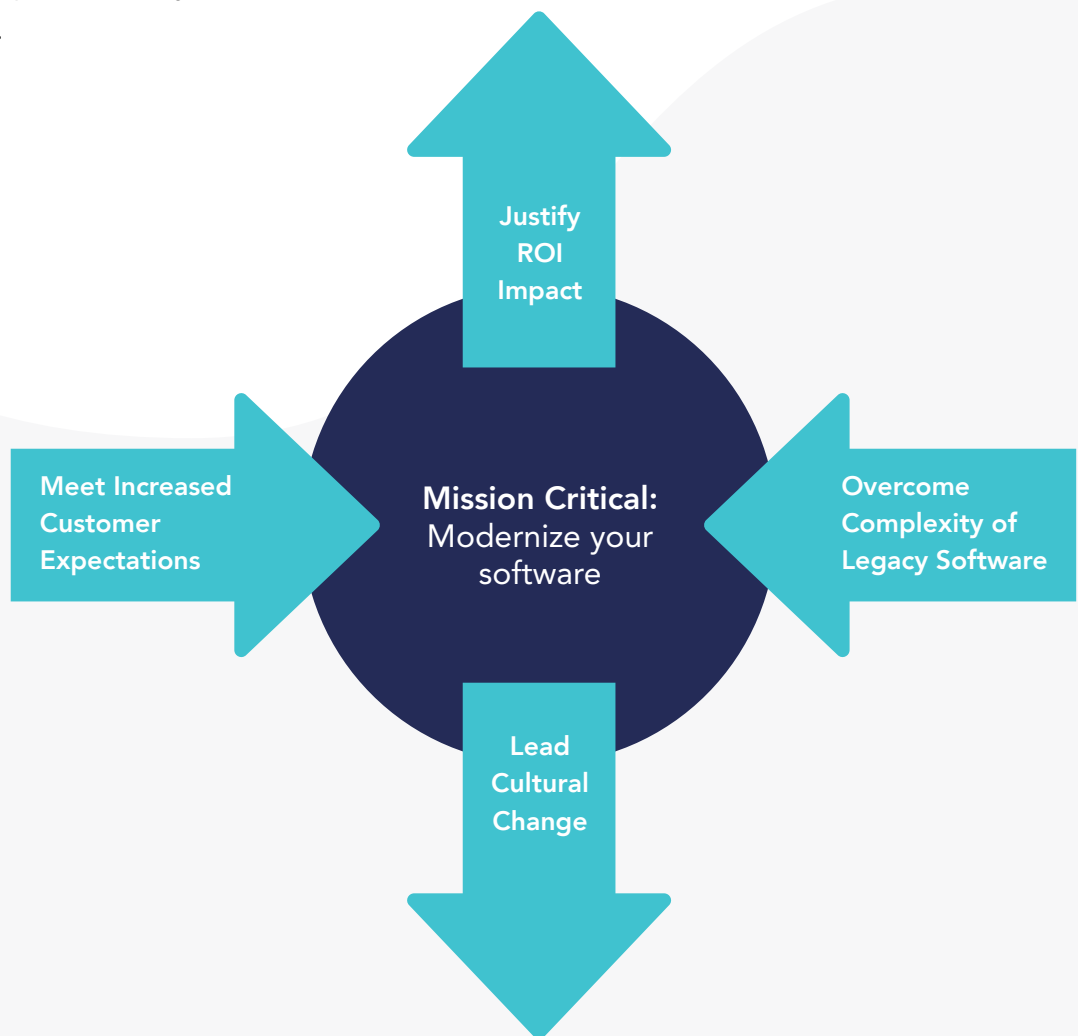
The irresistible force is the ever-increasing set of customer expectations for convenience and simplicity, accelerated by the example set by tech giants (you know who) and upstart, born-digital competitors. The immovable object is the suite of custom software applications that got you to where you are today; your business model depends on them. However, truly understanding your dizzying web of legacy systems is a tall order--let alone modernizing or replacing them.

Legacy codebases routinely exceed one million lines of source code, the equivalent of 20% of the data contained in the entire human genome--and they can be just as hard to decipher.



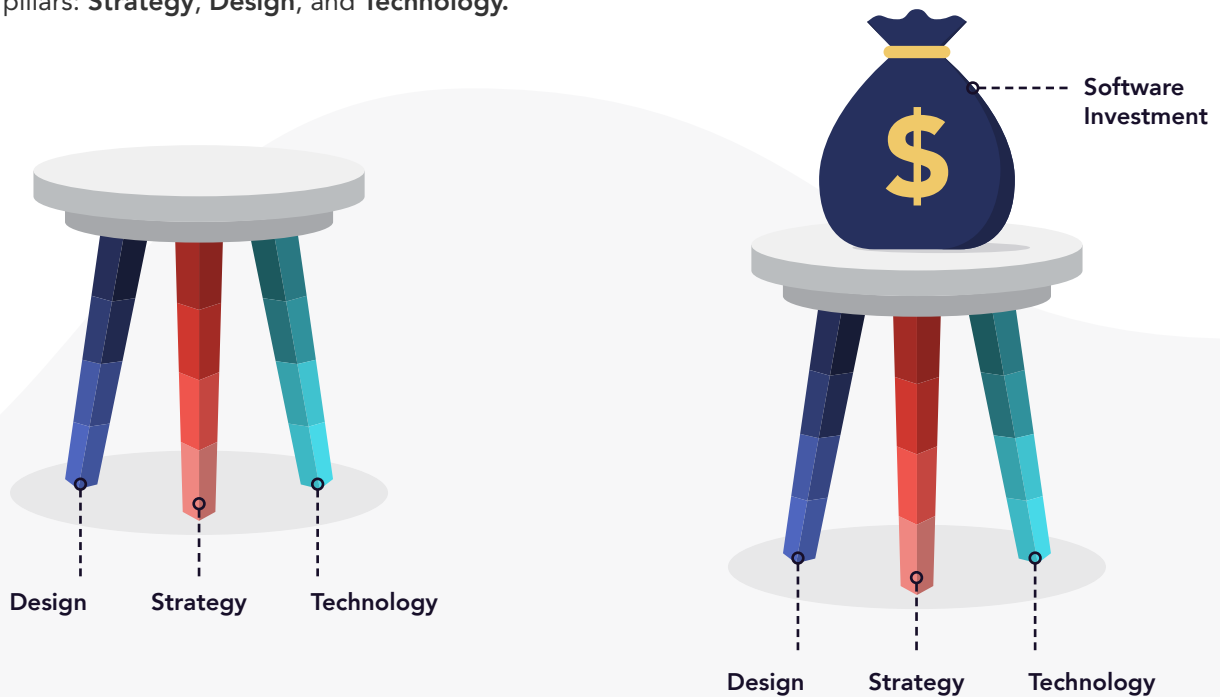
Mission Critical

The bottom line: to remain relevant to your customers, you have no choice but to replace or modernize your applications. However, justifying the cost and leading the necessary cultural change might feel like the next installment of Mission Impossible. But you know it's actually Mission Critical.



A Model For Success: The Modernization Foundation

The key to success with software modernization projects is to prepare by building a strong foundation on three pillars: **Strategy**, **Design**, and **Technology**.



To extend the metaphor, a strong foundation can safely support the weight of a major new investment in software modernization.

An unbalanced or insufficient foundation will put your investment at risk, not to mention your sanity.

The Four Reasons Software Modernizations Fail



The Four Reasons Software Modernizations Fail

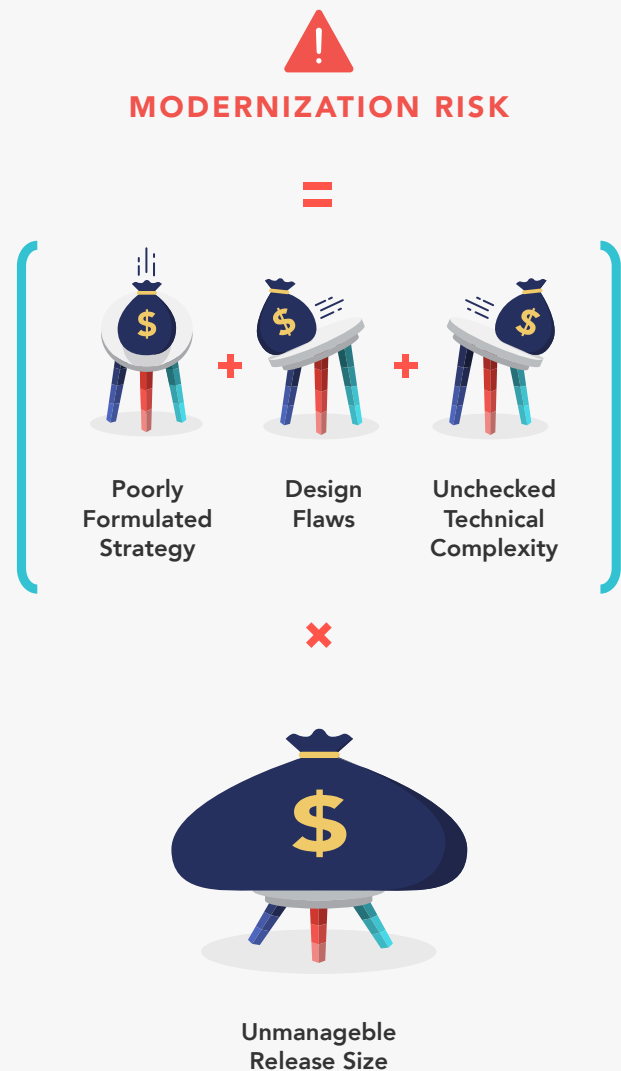
Over my 25-year career in custom software development, I have had the opportunity to work with more than 300 companies on their initiatives, and have consulted with thousands. While I have led the development of net new software applications, the majority of my experience has come from modernizing, enhancing, or replacing existing software systems.

With the benefit of this experience, it is clear that software modernization failures can always be attributed to one or more of the following three causes:

- **Poorly Formulated Strategy:**
Misaligned business and technology strategy.
- **Design Flaws:**
Inadequate requirements and user interface design.
- **Unchecked Technical Complexity:**
Introduced during software development.

In addition, there is a fourth factor that acts as a risk multiplier to the first three:

- **Unmanageable Release Size:**
Long release cycles that delay valid user feedback.



Poorly Formulated Strategy

As with most organizational endeavors, the biggest obstacle to success with software modernization is the lack of a clear, mutually understood definition of success. In addition to securing buy-in from all relevant stakeholders, your objectives must also make sense financially. If they do not, support for your effort will wane when you need it most.

In addition to delivering a better customer experience and improved market differentiation, software modernization initiatives often provide benefits across an organization such as:

- **Marketing:** Streamline content publishing, offer upsells to customers, and drive renewals.
- **Technology:** Reduce technical debt and train developers on new technologies and methodologies.
- **Project Management Office (PMO):** Trial a new set of program management best practices.
- **IT:** Migrate infrastructure to the cloud, enhance security, increase uptime, and decrease hosting costs.
- **Operations:** Streamline existing processes to save time and money, while also reducing customer service errors.



While these goals are all worthwhile, if they are not clearly outlined and agreed upon in advance, the allocated budget may not be sufficient. This, in turn, could cause stakeholders in these different departments to work at cross-purposes, proverbially rowing in opposite directions as they contend for scarce project resources.



Design Flaws

When I work with business leaders planning to modernize or enhance their software, they often center our conversations on the disruptive technologies we could employ to achieve their vision. For example:

- Can we use blockchain to manage our transactions?
- Would a combination of computer vision and machine learning enable us to eliminate time-consuming data entry processes?
- Could a chatbot handle the majority of our routine customer service inquiries?

In reality, and counter-intuitively, the success of a modernization initiative rarely hinges on the integration of these more recent innovations. First, these technologies often play a limited role relative to the overall software implementation. Second, there is almost always an open source software library, or third-party product/service that will provide 80% of the solution, right out of the box. In fact, beware if there's not.

Where initiatives do fail is in the definition of business requirements and user interface designs. When designing custom software, the details really matter. Missing even a few details can doom your project schedule and budget, or result in an application with low utilization. When customers choose to place a phone call or send an email, rather than use the software you have developed, the value of your software investments is greatly reduced.

Incorrect assumptions, missing requirements, and ineffective user interface designs aren't readily apparent and this can create a false sense of confidence in your team's progress. Project teams often report they are 80-90% complete with a software release, only to find that when real end users begin testing it, the software is too confusing or its functionality completely misses the mark. Design flaws results in costly rework and a seemingly endless series of delays that, in about 25% of all cases, results in the cancellation of the project before the software ever launches.

In short: it's not the development of an augmented reality investment advisor that will risk your modernization effort; it's remembering to develop and test the compliance disclaimer shown only to New York residents that reside in Florida for more than 40% of the year.

“
When designing custom software, the details really matter. Missing even a few details can doom your project schedule and budget, or result in an application with low utilization.

Project teams often report they are 80-90% complete with a software release, only to find that when real end users begin testing it, the software is too confusing or its functionality completely misses the mark.

Unchecked Technical Complexity

The final root cause of software modernization failures is unchecked technical complexity. Software systems composed of multiple applications can be as complex as a machine with a hundred thousand or more moving parts, intricately constructed over the course of several years (or decades) by a team whose membership changes frequently. The challenge can bring to mind the famous Sir Walter Scott quote, “What a tangled web we weave...”

This level of technical complexity can overwhelm the progress of even the most senior software development teams, if not carefully managed. In fact, the term ‘technical debt’ is often used to describe the concept of incremental additional cost with increased technical complexity, because, like financial debt, even strong organizations can fall victim to it if it’s not managed carefully.

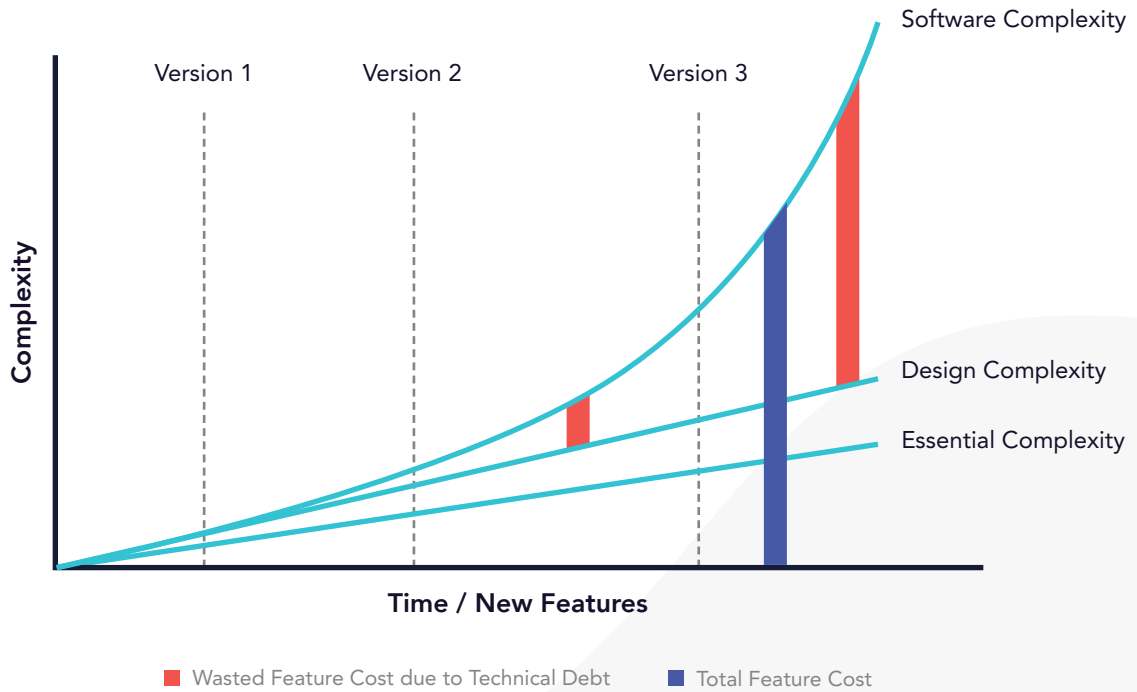


“

What a tangled web we weave...

SIR WALTER SCOTT

THE COMPOUNDING COST OF TECHNICAL DEBT



One goal of every software development team is to use software design and development best practices to keep the Software Complexity line (shown above) as close as possible to the Essential Complexity line. Essential Complexity is inherent in the non-negotiable business requirements for the software application. Design Complexity stems from the decisions made in the user experience design and requirements step for each feature.

Keeping Software Complexity under control allows a development team to continue serving the business well by moving quickly to cost-effectively add new

software features. However, even with the best of intentions, technical debt can begin to balloon as a result of unexpected requirements changes, inexperienced development team members, and the realities imposed by budget and timeline constraints.

All software applications have some level of technical debt and it can even be useful to accrue it in the short term to accelerate development. You just have to be mindful of it because, like financial debt, its exponential growth could stop your forward momentum in its tracks.

Unmanageable Release Size



When developing software, you are forced to make countless assumptions (consciously or unconsciously) about how the software should function and how it will actually function under the stresses of real-world usage, in the hands of actual end users. The longer these assumptions go untested, the greater the risk that, to use an analogy, your boat won't float when you put it in the water.

While there are many approaches to simulate real-world usage to validate your assumptions, there is no equivalent to the feedback you receive when you actually launch a software product. The next best thing is the feedback you can gather on a prototype, which can prevent costly development rework.

Adopting a large project strategy, where your team works on a software release for six months or more before launching it, acts as a risk multiplier to the other three failure causes: poorly formulated strategy, design flaws, and unchecked technical complexity. Data from the Standish Group's Chaos Report bears this out:

	LARGE PROJECT	SMALL PROJECT
Percentage of projects that are challenged or cancelled	90%	24%
Percentage of projects that are cancelled prior to launch	38%	4%

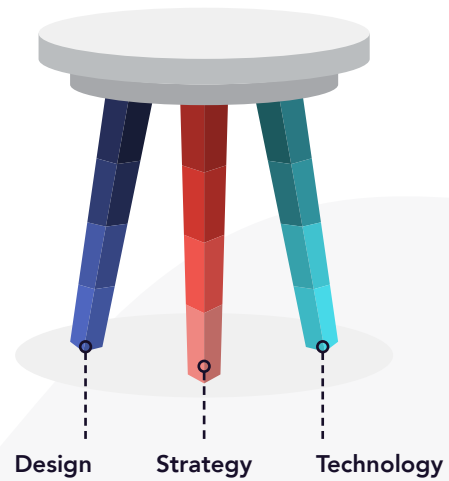
12 Strategies for Software Modernization Success



12 Strategies for Software Modernization Success

We've now covered why companies modernize their customer-facing software and the pitfalls that sometimes derail those modernization efforts. What can you do to ensure success? As previously mentioned, the key to success is preparing a strong Modernization Foundation, which rests on three pillars: Strategy, Design, and Technology.

Within each pillar, there are four strategies that build upon one another. In other words, you cannot effectively move on to the second strategy until you have addressed the first:

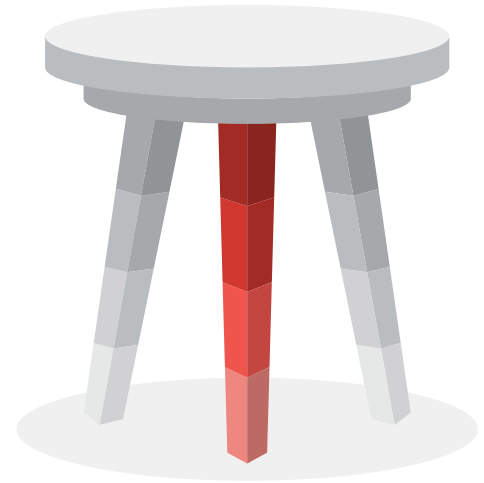


PILLAR	STRATEGY	DESIGN	TECHNOLOGY
FIRST	Clear Problem Statement	Application Documentation	Evolutionary Architecture
SECOND	Objectives	Usage Context	Software Delivery Pipeline
THIRD	Phased Roadmap	Design Objectives	Reusable API Layer
FOURTH	ROI Analysis of Phase 1	User Experience Framework	User Interface Framework

STRATEGY

The Do's and Don'ts of Modernization

Head to the [appendix](#) for an in-depth guide to each modernization strategy, along with example deliverables and diagnostic questions.



STRATEGY	DO	DON'T
Clear Problem Statement	<ul style="list-style-type: none"> ✔ Identify one or more compelling problems with the status quo 	<ul style="list-style-type: none"> ✘ Forget to quantify the impact of the problem
Objectives	<ul style="list-style-type: none"> ✔ Collaboratively set measurable goals and decide on your approach 	<ul style="list-style-type: none"> ✘ Get into the details yet - keep it big picture
Phased Roadmap	<ul style="list-style-type: none"> ✔ Break down your goals into a phased approach to produce quicker wins 	<ul style="list-style-type: none"> ✘ Fall into the trap of addressing every need in your next software release
ROI Analysis of Phase 1	<ul style="list-style-type: none"> ✔ Demonstrate and document how Phase 1 will pay for itself 	<ul style="list-style-type: none"> ✘ Seek funding for all phases before beginning Phase 1, if at all possible

DESIGN

The Do's and Don'ts of Modernization

Head to the [appendix](#) for an in-depth guide to each modernization strategy, along with example deliverables and diagnostic questions.



DESIGN	DO	DON'T
Application Documentation	<ul style="list-style-type: none"> ✔ Educate your team on the current business processes and technology 	<ul style="list-style-type: none"> ✘ Assume an understanding of the status quo will prevent innovation
Usage Context	<ul style="list-style-type: none"> ✔ Talk to your end users and observe them using your technology 	<ul style="list-style-type: none"> ✘ Assume you fully understand your customer's experience of your software
Design Objectives	<ul style="list-style-type: none"> ✔ Identify principles you will return to with every design decision you make 	<ul style="list-style-type: none"> ✘ Forget to tie your design objectives to your underlying business goals
User Experience Framework	<ul style="list-style-type: none"> ✔ Create a design language you can reuse across your application(s) 	<ul style="list-style-type: none"> ✘ Rely solely on long, written documents to convey requirements

TECHNOLOGY

The Do's and Don'ts of Modernization

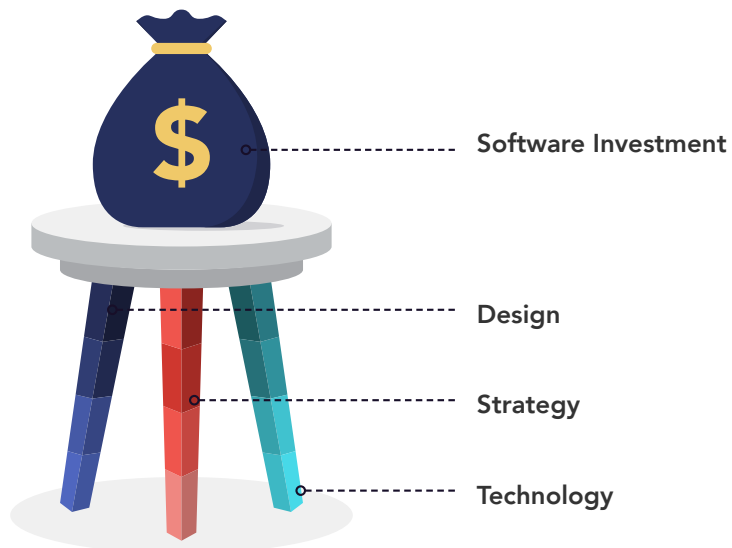
Head to the [appendix](#) for an in-depth guide to each modernization strategy, along with example deliverables and diagnostic questions.



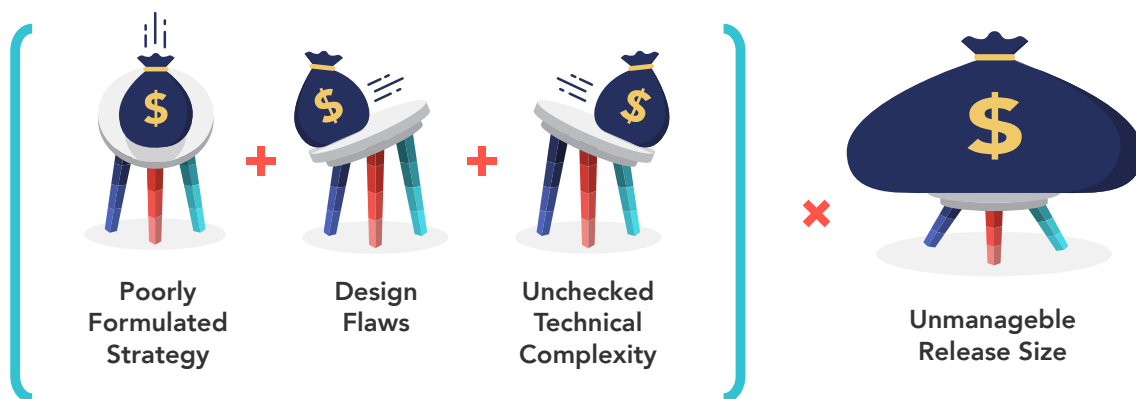
TECHNOLOGY	DO	DON'T
Evolutionary Architecture	<ul style="list-style-type: none"> ✔ Create a plan to incrementally roll out your new software 	<ul style="list-style-type: none"> ✘ Attempt a hard, irreversible cutover from your legacy system
Software Delivery Pipeline	<ul style="list-style-type: none"> ✔ Automate as many aspects of software testing and deployment as possible 	<ul style="list-style-type: none"> ✘ Tolerate manual, error-prone activities that delay feedback cycles
Reusable API Layer	<ul style="list-style-type: none"> ✔ Plan ahead on your API design and development to prevent delays 	<ul style="list-style-type: none"> ✘ Begin developing your front-end before deciding on your API contracts
User Interface Framework	<ul style="list-style-type: none"> ✔ Leverage a modern UI framework that enables reuse of common components 	<ul style="list-style-type: none"> ✘ Leverage outdated UI technology solely because of your team's familiarity with it

Reference Card

MODERNIZATION FOUNDATION



MODERNIZATION RISK



Reference Card



STRATEGY	DESIGN	TECHNOLOGY
Clear Problem Statement	Application Documentation	Evolutionary Architecture
Objectives	Usage Context	Software Delivery Pipeline
Phased Roadmap	Design Objectives	Reusable API Layer
ROI Analysis of Phase 1	User Experience Framework	User Interface Framework

Implement these 12 strategies in your next software modernization project, and I guarantee you'll see results.

If you prefer to fast-track your way to success, hop on a free call with one of our software modernization experts, who will personally walk you through each of the 12 strategies and provide you with a customized readiness report you can share with your team.

[Book my Readiness Assessment](#)

About Praxent

At Praxent, we know you want to be a savvy, pragmatic innovation leader. In order to do that, you need a modern, intuitive digital experience to serve customers. The problem is your current applications are too complex to easily rebuild and are becoming more outdated by the day. Most likely, that has left you feeling overwhelmed by the complexity of your legacy systems and unable to align stakeholders around the need for change.

We believe there should be a way to improve your digital customer experience without betting the farm on an all-or-nothing rewrite. We understand your software applications aren't getting any younger, which is why we have assembled a fast-moving team of 70+ software designers and engineers that have successfully delivered over 300 software transformations over the past 20 years.

Here's how we do it:



1. ASSESS

Together, we assess your situation and present a proposal with three investment options.



2. DESIGN & BUILD

We design and build a modern user experience, supported by a robust full-stack architecture.



3. LAUNCH

You launch a modern, intuitive software application that lives up to the promise of your brand.

Schedule a call today, so you can stop losing customers to born-digital competitors and, instead, enjoy the feeling of accomplishment as you compete with confidence.


[Schedule a call](#)

Appendix: 12 Strategies for Software Modernization Success





Strategy

STRATEGY	CLEAR PROBLEM STATEMENT
	<p>There is a reason that most marketing experts will encourage you to sell aspirin instead of vitamins. For better or worse, current pain is a far better motivator for change than future pleasure, even if you are just convincing yourself to make an investment.</p> <p>The first step to effectively pursuing a modernization effort is to clearly define the problem with the status quo. Quantify the problem in dollars and cents, even if you need to make a few assumptions to do so. These initial calculations will prove critical in subsequent steps. At a minimum, it is critical to quantify the problem using metrics that eventually drive financial outcomes.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Sales Funnel Conversion Analysis: What percentage of late-stage opportunities are you losing to competitors due to an outdated customer experience? • Customer Retention Analysis: What percentage of your customers renew? How many more might renew if you offered a better customer experience? • Marginal Cost of Delivery Analysis: How much manual labor could be eliminated for your next sale through customer self-service and automation? • New Opportunity Analysis: Are you currently shut out of a new market opportunity due to your customer experience? How much is the missed opportunity worth?
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✓ What inputs do you need from departments across the organization including executive leadership, marketing, operations, and technology? ✓ How will the investment increase sales, increase retention, create new market opportunities, drive word of mouth, and reduce write offs?


STRATEGY	OBJECTIVES
	<p>If you have successfully defined the problem, congratulations! That truly is the hardest and most important step. Now it's time to start defining your goals and approach. In what ways could you enhance your website to create a better customer experience? At which points in your customers' journey do they experience the most friction? How will you measure success after you launch the new version of your website?</p> <p>To spur your creativity around your approach, there are a number of strategy tools that can help.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Strategy Canvas: An analysis of the value drivers in your industry and how your firm competes on each. • Buyer Utility Map: An analysis of the systemic customer pain points at different stages of the buyer's journey in your industry. • Customer Journey Map: What touchpoints does a customer have with your brand? Where are you excelling? Where are you struggling? • Generative User Research: 5-10 conversations with your current and prospective customers will yield invaluable insights on the challenges they face working with you and your competitors. • Feature Prioritization Matrix: A collaborative exercise to rank the business value versus cost for a series of potential CX enhancements.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✓ Can you map each of your objectives to one or more of the challenges identified in your problem statement? ✓ Have you solicited input and sought buy-in from key stakeholders across your organization?

STRATEGY	PHASED ROAD MAP
	<p>Hopefully at this point you have a solid list of ideas for how you can modernize and enhance your customer experience, as well as, a clear and measurable definition of success.</p> <p>This next step is critical. How should you prioritize the development and launch of your exciting new CX improvements? Even if you had the time and funds to pursue all of your objectives, it is best to phase them out over time so you can start earning a return on your first round of investment while your next round of enhancements is being developed. Organize your ideas into the following four categories:</p> <ul style="list-style-type: none"> • First Release • Next Release • Eventually • Maybe/Someday
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Product Roadmap: A high-level outline of the next 2-3 releases with the capabilities that will be included in each. Note, the road map will have to be revisited periodically as software estimates are produced and as business priorities change.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✓ For clarity, can you organize each release in your roadmap around a theme (i.e. ‘policyholder self-service’ or ‘streamlined underwriting’)? ✓ Are there significant dependencies or obstacles that could prevent the launch of your first release? ✓ Are your releases focused/prioritized by value to the end user? Grouping in this way can lessen the change management required for each release.

STRATEGY	ROI ANALYSIS OF PHASE 1
	<p>A well-designed road map puts you in the enviable position of a cost-justified first release. This will enable you to be agile and iterative using the new lessons learned during development of the first release.</p> <p>Unfortunately, many organizations tend to view customer experience optimization as once and done, meriting attention every 5-10 years, instead of an ongoing business function. This thinking causes the scope of these less frequent initiatives to be much longer, more expensive, and more risky. The goal, instead, is to break up these initiatives into several smaller efforts that each produce their own ROI.</p> <p>The most impactful decision you can make to succeed with custom software development is to adopt a small project strategy; it's the closest thing to a 'silver bullet' you have in your arsenal to stack the odds in your favor.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • ROI Analysis: Spreadsheet demonstrating how improvements to metrics such as customer retention and sales conversion rate will produce an ROI.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✓ What can be deferred from your first release to the second while still earning an ROI on your first release? ✓ What budget threshold should you stay under for your first release to streamline approval? ✓ Can you include a 20-30% contingency in your budget to reduce the likelihood of requiring a second budget request to complete your first release?

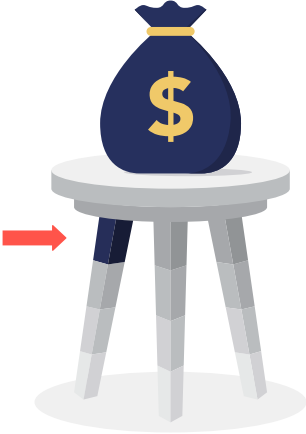


Design

DESIGN	APPLICATION DOCUMENTATION
	<p>One of the single greatest mistakes leaders can make is to disregard the strengths and lessons learned from your current software when considering a rewrite or modernization. It is usually done to avoid the constraints imposed by what 'is', but like trees that have outlived the people around them, your software likely contains information not available in any other document or person's mind.</p> <p>Carefully documenting your current software will prevent you from overlooking key details that, once discovered, could delay or derail your project. Asking a new team member to create this documentation, if you do not already have it, is a great way to make them an expert on your current technology and business operations.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Feature Inventory: A list of each of the features included in a software application, as experienced by an end user. • Site Map: A diagram that outlines the different areas of an application and how they are connected. • Data Dictionary: The purpose and contents of the tables and columns in a database. • Software Architecture Diagrams: Physical and logical architecture diagrams that describe the technical layout of a software system. • API Specification: A description of the purpose, inputs, and outputs of API endpoints used to support a customer-facing software application.
<p>Diagnostic Question</p>	<ul style="list-style-type: none"> ✓ Do we understand the current software well enough to prevent a missed requirement for the next version (that could result in a costly and unexpected delay)?


DESIGN	USAGE CONTEXT
	<p>Building on that documentation will enhance your understand of the usage context of your current software.</p> <p>A data dictionary can begin to answer those questions, but there is no replacement for seeing for yourself how your customers actually use your software. By watching and listening to end users, you will identify bottlenecks, pain points, and unanticipated usage scenarios that you may never otherwise design for.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Usage Statistics: Data from a source like Google Analytics that show how your software is used. • User Persona: A document describing a specific set of end users, such as their job responsibilities, needs from the application, and current pain points. • Customer Journey Map: A chart visualizing how a customer interacts with your company over time. • Heuristic Analysis: An evaluation of the usability of your software from a UX expert’s perspective. • Usability Testing: User research study conducted to understand how usable your software is. • Contextual Observation and Inquiry: A series of meetings with end users, ideally in-person, where they demonstrate how they currently use your software and their frustrations, if any.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✓ Have you spoken with enough end users to begin noticing clear patterns in how they use your software and the obstacles they must overcome to do so? ✓ Are there any underutilized parts of the system? Who uses them? Are they still necessary?


DESIGN	DESIGN OBJECTIVES
	<p>What overarching goals should your software designers keep in mind in order to set design goals for the software that align with the business objectives?</p> <ul style="list-style-type: none"> • Minimize the steps to complete each process? • Increase end user productivity? • Provide clear direction at every step? • Educate customers on your products? • Highlight the availability of upsell options? <p>Collectively, these ideas comprise a “North Star” for your software designers to follow. In addition to design goals, other deliverables can help you set the high-level design direction for your new software.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Epic Backlog: A list of features and capabilities the first version of your software must include to launch. • To-Be Process Flows: Process flow diagrams that show how different end users, both internal and external, will collaborate through the software. • Security Roles: Different levels of security access that will be granted to different user types (i.e. personas). • Data Visibility Hierarchies: A rule for how much data different end users can see. For example, an HR manager can only see their data and that of their team.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✓ Where have you posted your design objectives? How can you make them more visible? ✓ Can you visually and operationally align your design goals with your business objectives?

DESIGN	USER EXPERIENCE FRAMEWORK
	<p><u>Stephen Covey wrote</u> that all things are created twice -- first conceptually (i.e. 'on paper') and then in real life.</p> <p>When it comes to software, the best way to build software 'on paper' is through a design prototype. This allows your team to quickly spot deficiencies and missed expectations from key stakeholders instead of using long written documents most often for this purpose that are skimmed, ignored, or interpreted in very different ways. Before beginning development, design the most important features and define your application's branding (i.e. look-and-feel), navigation (i.e. menu structure), and user interface components such as your dashboard elements, data tables, data inputs, report formats, notifications, and help messages.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Wireframes: Drawings of specific screens within the application that range from low-fidelity (i.e. rough sketches) to high-fidelity (fully-designed, branded screenshots). • Design Prototypes: A clickable prototype that demonstrates how an end user will move through key parts of the system, one screen at a time. • Prototype Validation: Interactive sessions where a UX designer asks an end user to perform tasks in the prototype and observe how user-friendly it is. • Digital Style Guide: A set of design choices such as colors, font styles, icons, buttons, and user messages that, when followed, will create consistency throughout your application.
<p>Diagnostic Question</p>	<ul style="list-style-type: none"> ✓ Will your stakeholders be surprised the first time they see a demo of your new software? (If so, you could be facing one or more costly rounds of rework)



Technology

TECHNOLOGY	EVOLUTIONARY ARCHITECTURE
	<p>Embarking on a large software rewrite can be like a long sea voyage. The longer you sail between safe ports of harbor (i.e. releases), the more likely you are to be blown off course or encounter a storm you cannot overcome. Fortunately, there is a better way.</p> <p>For small applications, it is usually manageable to rewrite them from the ground up. For complex software applications, a more incremental approach is necessary.</p> <p>By building new modules and integrating them with your existing software in iterations, you can remain relevant for existing users while attracting new ones with a fresh digital experience, delivered quickly and with less risk. This also allows you to replace critical features in a prioritized order for faster impact. To employ this strategy, you will need a game plan to connect your new features with your older software.</p>
<p>Example Deliverables</p>	<p>Strangler Pattern: A <u>best practice</u> conceived by software pioneer Martin Fowler to combine old software with new, often supported by the creation of a new service layer.</p> <ul style="list-style-type: none"> • Database/Service Reuse: Reusing a database and/or service layer between major versions of an application to support the integration of the two. • Single Sign-On: Enabling end users to move seamlessly between a newer and an older web application, without the need to sign on again.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✔ Can both your new and old software run side-by-side? ✔ Does your chosen approach provide flexibility in which features you modernize first?

TECHNOLOGY	SOFTWARE DELIVERY PIPELINE
	<p>Software developers are often among the mostly highly paid individual contributors at a company. However, in many companies the technical environments in which they work render them highly unproductive. Why?</p> <p>Software development is all about trial-and-error. When I developed software, it was not uncommon for me to attempt 5-10 solutions to a problem before finding one that worked well -- and I was far from alone. That's just how it works.</p> <p>The key to optimizing your productivity in software development is to compress your feedback cycles. Fast feedback empowers developers to deliver value more efficiently, but very few organizations are successful at this. A well-designed software delivery pipeline will help your entire development team move from requirements to completed software features with minimal delays and wasted effort.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Continuous Delivery Pipeline: Automated deployment processes to promote secure, quality software from local developer workstations to testing, staging, and production environments. • Source Control: Best practices for managing and integrating the source code of multiple developers to prevent defects and rework. • Automated Testing: A companion software application designed to automatically test the features in your primary application as developers make changes to it.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✔ Is your process of deploying your software to an environment where you can test it error-prone? Where is the process breaking down today? ✔ Do you only deploy your software during off hours, for fear something will go wrong?

TECHNOLOGY	REUSABLE API LAYER
	<p>In 2002, Jeff Bezos famously <u>issued an ultimatum</u> to every software team at Amazon: build APIs to access all of the software you are developing or be fired. This visionary mandate put Amazon in a position to modernize and extend its platform indefinitely, laying the foundation for its AWS business unit, which now generates \$35B in annual revenue.</p> <p>Fortunately, the development of APIs is a natural fit for how web and mobile-based applications are developed today. Because modern applications run directly on a user’s laptop or mobile device, APIs are required to simplify software development and enforce security. However, the first generation of web applications were mostly developed without APIs.</p> <p>Before proceeding on your modernization journey, it is imperative you plan out which APIs you have now and which you will need to support your new customer-facing applications.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Secure Design: While there are multiple dimensions to security, your API layer will serve as your most important bulwark against data breaches. • Cloud-Based Deployment: Cloud-based platforms like AWS and Azure help address the “ity’s” more cost-effectively than ever before: scalability, security, extensibility, business continuity, etc. • SOLID Design Principles: <u>SOLID design</u> is the antidote to technical debt. Taken together, the SOLID principles reduce the incremental cost of new feature development over the long term.
<p>Diagnostic Question</p>	<ul style="list-style-type: none"> ✓ Are the data included in your wireframes and prototypes available today via an existing API or one you plan to develop as part of phase 1?

TECHNOLOGY	USER INTERFACE FRAMEWORK
	<p>Digital customer experience has come a long way, and so have the software development technologies that support it. Perhaps the best way to appreciate the rapid evolution of these technologies is to scroll through screenshots of Amazon.com from 1995 until today. How far down did you have to scroll until you found a version of Amazon.com that resembles your own website today?</p> <p>Websites built to serve customers are no longer simple web pages with images, text, and battleship gray buttons. They are sophisticated, responsive systems unto themselves, built to support complex workflows and rich content types. They radically adapt their appearance based on the end user's preferences, screen size, spoken language, country of residence, and even disability such as reduced vision.</p> <p>Before moving into the development of a modern front-end, make sure your team is ready with the skills necessary to ensure you are not investing in yesterday's technologies today.</p>
<p>Example Deliverables</p>	<ul style="list-style-type: none"> • Front-End Frameworks: Frameworks, such as those developed in Javascript and CSS, streamline the development of rich user experiences and make your systems available to mobile users. • Accessibility: While often overlooked, it is critical to design for users with disabilities from the beginning. Doing so later will likely require a complete rewrite.
<p>Diagnostic Questions</p>	<ul style="list-style-type: none"> ✔ Will your user interface framework enable you to support the modern user experience envisioned by your designers? ✔ Can users easily access your systems using a mobile device?

Conclusion

A modern, intuitive customer experience, delivered digitally, is becoming a prerequisite for growing customer relationships and acquiring new ones. We all crave better, more user-friendly technology. As evidence, we replace our mobile devices on average every 1-2 years. The financial services companies that provide the best digital customer experiences will earn the loyalty of their customers and attract others fed up with confusing and ineffective websites and mobile apps.

Modernizing or replacing legacy customer-facing software is easier said than done. Poorly formulated strategy, design flaws, and unchecked technical complexity can make the success of a large development initiative a remote possibility even before the first line of code is written. The key to success is a strong modernization foundation built on the three pillars: strategy, design, and technology.

You can do it! We can guide you.

“
The key to success is a strong modernization foundation built on the three pillars: strategy, design, and technology.

AUTHOR PROFILE



Kevin Hurwitz

MANAGING PARTNER, PRAXENT

Kevin's passion for computers began in 1982 when his family bought him a Tandy home computer on which to play and eventually program games. Kevin's childhood hobby of computer programming led to a lifelong passion for turning code into tangible business outcomes, starting with internships and contract engagements throughout high school and college.

Kevin has consulted with more than 250 organizations to deliver proprietary technology solutions, ranging from startups to Fortune 500 companies such as ExxonMobil, GE, Sysco Foods, Flex, Verizon, and Dell. Kevin's recent positions include Headspring where he served as EVP of Client Success, Better Insights, and Praxent where he joined as Managing Partner in 2016. At Praxent, Kevin is responsible for business development and client success. Kevin holds a B.S. in Computer Engineering from Texas A&M University and an MBA from Jack Welch Management Institute.

“

Digital innovation can be a game-changer or a costly distraction. The difference lies in the people and processes you use to go after it.

Dramatically improve your customer experience with a modernized digital system built specifically for your business—with minimal risk—so you can compete with confidence.

[Book my Readiness Assessment](#)

Schedule your free call at praxent.com/contact today.



4330 Gaines Ranch Loop, Suite 230
Austin, TX 78735
(512) 553-6830 | hello@praxent.com

[PRAXENT.COM](https://praxent.com)