mobilize.NET

# A CFO's Guide to Legacy Applications

Risks, costs, and options.

Your App. New. Again.

# Do you know the state of your soft assets?

You can see the wear down of hardware and other equipment - but you can't see software eroding.

Which leads us to mistakenly believe that legacy software is an "invisible cost".

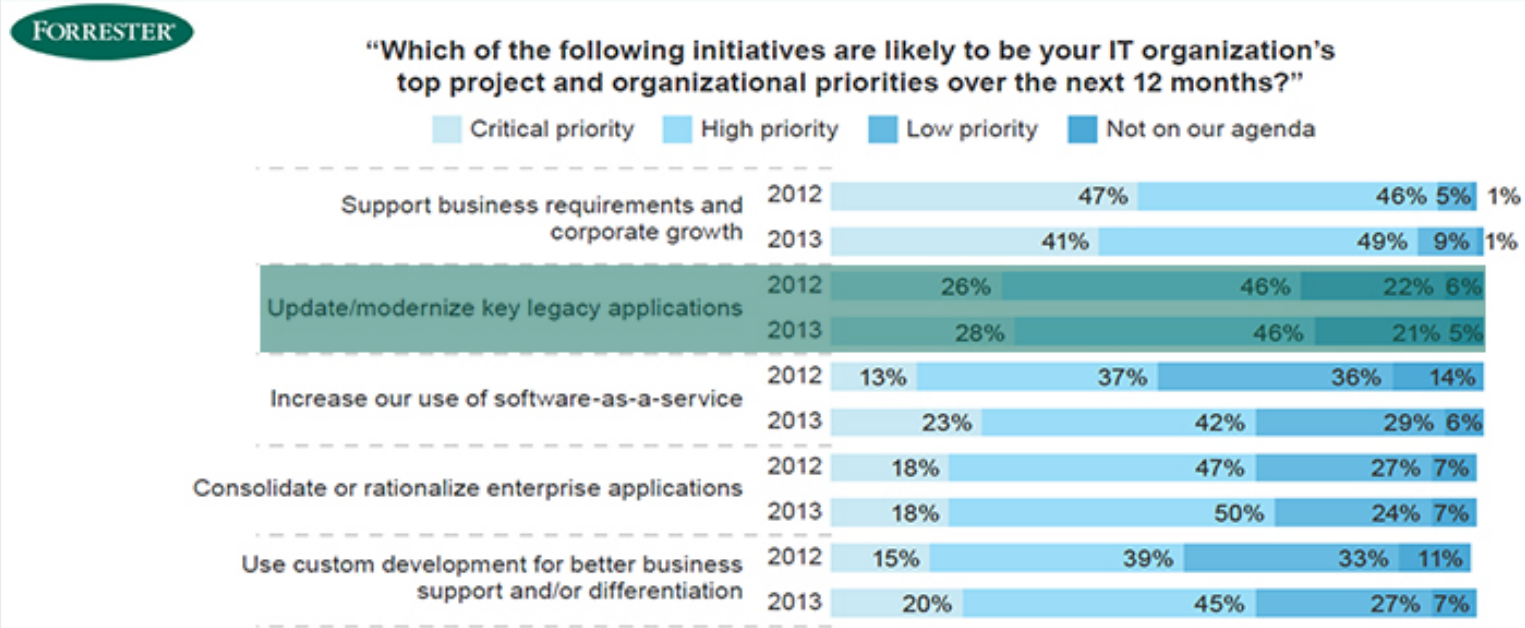
Did you know that 70% of IT budgets are spent supporting legacy software?
**That's pretty visible.**

In fact….
Check out the numbers on legacy software…

# Among enterprise IT, governments and ISVs greater than 15 yrs old…

## Few problems more ubiquitous than legacy apps

FIGURE 1
Custom Development And SaaS Are On The Rise As IT Organizations Seek To Meet Business Requirements

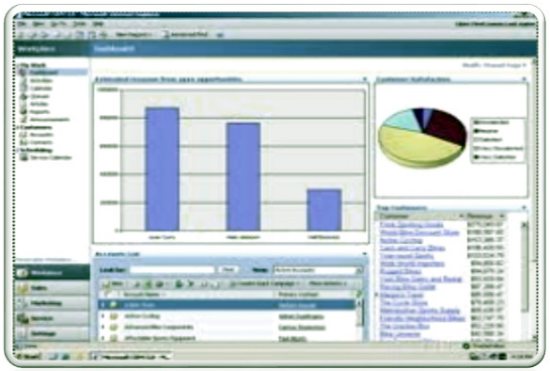



Base: 798 (2012) and 605 (2013) IT respondents from enterprise-size organizations in the US and Canada
Source: Forrsights Software Survey, Q4 2013, Forrester Research, Inc. and Forrsights Software Survey, Q4 2012, Forrester Research, Inc.

* 74% (and growing) of enterprise IT leaders say "updating/modernizing key legacy applications" is a "high" or "critical" priority.

# Software obsolescence is a **huge and growing** problem. Technology platforms are constantly changing

## IT spends massive resources maintaining legacy software



- 74% of enterprise IT leaders say "updating/modernizing key legacy applications" is a "high" or "critical" priority. It's the number 2 priority.
- 59% of IT leaders identify this as the top software issue.
- Client/server apps have overtaken COBOL and mainframe apps as their greatest legacy modernization challenge.
- Mobile and cloud are the dominant modernization target platforms.
- 84% of IT orgs plan to seek external help to accomplish this.

## Client/server legacy is huge, starting with Visual Basic…

- From 1993 to 2003, 5 million pro software developers wrote custom business Visual Basic applications for IT.
- In that time, they wrote 550K projects per year.
- Do the math. That's over 50 billions lines of VB code.
- Even now, roughly 250,000 developers still use VB to maintain those apps, costing IT $25 BILLION a year.
- That number doesn't include C/C++, PowerBuilder, Java, .NET (WinForms, WPF, Silverlight), Delphi, and the list goes on…

# Greatest Legacy Challenge Today

## Moving client/server apps to mobile/cloud



### Does your firm have legacy application challenges?

| | Very serious legacy application challenges | Somewhat serious legacy problem | Mild legacy problems | We don't have a legacy application problem | Rating Average | Response Count |
|---|---|---|---|---|---|---|
| Legacy challenges | 22.6% (177) | 34.4% (270) | 31.4% (246) | 11.6% (91) | 2.32 | 784 |

### What types of legacy systems present the biggest challenges for your firm?

| | In great need to modernize | Some what serious legacy problem | A few of these to modernize | None of these to modernize | Rating Average | Response Count |
|---|---|---|---|---|---|---|
| Client Server Apps | 19.4% (152) | 25.6% (201) | 26.8% (210) | 28.2% (221) | 2.64 | 784 |
| COBOL | 16.2% (127) | 15.6% (122) | 13.1% (103) | 55.1% (432) | 3.07 | 784 |
| Mainframe | 20.4% | 19.3% (151) | 15.9% (125) | 44.4% (348) | 2.84 | 784 |
| Early web apps | 11.7% (92) | 21.0% (165) | 30.0% (235) | 37.2% (292) | 2.93 | 784 |

### How do the following technologies factor into your legacy modernization roadmap?

| | Already implemented | Plan to implement within 12 mos | Plan to implement beyond 12 mos | Never plan to adopt | Not sure yet | Response Count |
|---|---|---|---|---|---|---|
| Mobile | 26.8% (199) | 29.2% (217) | 21.3% (158) | 7.1% (53) | 15.6% (116) | 743 |
| Cloud | 19.4% (145) | 25.7%(192) | 20.2% (151) | 11.4% (85( | 23.3% (174) | 747 |
| Virtualization | 57.9% (432) | 14.6%(109) | 9.2% (69) | 5.8% (43) | 12.5% (93) | 746 |
| Rich Internet Apps | 28.9% (213) | 19.6%(145) | 13.4% (99) | 9.5% (70) | 28.6% (211) | 738 |
| Open Source | 39.4% (294) | 13.5%(101) | 9.0% (67) | 14.2% (106) | 23.9% (178) | 746 |
| SaaS | 30.1 (224) | 20.5%(152) | 15.1% (112) | 10.1% (75) | 24.2% (180) | 743 |
| SOA | 39.8% (296) | 21.7%(161) | 11.3% (84) | 7.9% (59) | 19.2% (143) | 743 |

1. **57% of IT reports** having "serious" legacy
2. **Client/server legacy** is most common
3. **Mobile and cloud** most popular technology choices

Forrester Research, 2012

# What is “legacy software?”

- **Somebody wrote it years ago**
- **It is still being used; it still has value**
- **Years of accumulated technical debt**
- **Someone has to maintain it**
- **Visualize duct tape and superglue**

In computing, a legacy system is an old method, technology, computer system, or application program, “of, relating to, or being a previous or outdated computer system.” Often a pejorative term, referencing a system as “legacy” often implies that the system is out of date or in need of replacement.

Legacy system - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/Legacy_system

# Legacy code is expensive to keep around.

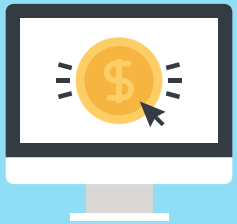
Staffing is expensive.

Compliance is expensive.

Maintenance is expensive.

Agility is non-existent.

# Higher staffing costs for specialized talent.



**Legacy applications require legacy skills.**
Skills that are hard to find so they're more expensive. When there are few resources with the required skills, you can bet you're going to pay a lot.

That's why current open systems and platforms not only have an abundance of competitive talent, but also have readily available support in terms of wiki's, expert blogs, web sites and more.

Plus, many legacy applications are built around proprietary, single-vendor platforms. Modern applications have transitioned to open source tools and programming languages, and open APIs allowing IT departments to hire employees with generalized experience and skills to work on a wide variety of applications.

# Compliance can kill you.

Have you heard of HIPAA? Sarbanes Oxley? Those are two examples of (many) US regulations that require extensive work to maintain compliance.

Applications written 10, 15, 20+ years ago do not conform to current requirements.
Even worse, industry and government regulations change all the time. Five years ago, no one heard of Meaningful Use, FLSA, ERISA, etc. – now they're law – and they're being enforced.

Each year, between 3,500 and 3,800 new government regulations are passed – in the US.
That number doesn't include all of the regulations being passed by the EU, by individual states and countries.

## Legacy software is the enemy of compliance.

# High maintenance fees. $$$$$.

## High maintenance fees for legacy vendors

(i.e. high support contracts for Windows XP) can be crippling. What happens when the vendor stops supporting your software version? As support expires and warranties end, costs increase — embedding another hidden cost into supporting a legacy application.

You can try to support it yourself, or upgrade to the vendor's latest edition.
But, what happens if you can't (or don't want to) upgrade? It's the problem you'll run into repeatedly with legacy systems: You lack options. You pay more for staffing because your options are limited. You'll pay more for support for the same reason.

## Higher maintenance costs for supported obsolete hardware are common.

Old infrastructure is expensive and older applications generally require an older technical environment, which might include old operating systems, databases, libraries, and specific hardware. Just as applications grow more expensive to maintain as they age, so does the underlying infrastructure. Modern applications that are designed for cloud, SaaS and SOA platforms don't have this problem.

# No agility.

Cloud has changed the economics of software maintenance but you'll never know because you're stuck on old apps that don't support it.

Cloud computing has revolutionized corporations that were once dominated by large and expensive legacy applications.

Not having to host huge datacenters takes a massive capital expense off the balance sheet for the CFO and alleviates the need for huge teams of consultants or specialized engineers reporting to the CIO.

**Legacy is difficult to mobilize**

Web and mobile devices have transformed enterprise computing. Old desktop apps not only tether users to a PC, but they're hard to deploy. By definition, apps that are hard to deploy will be out-of-date and stale.

# Legacy code is risky to keep around.

We've already talked about the risk of non-compliance with standards like HIPAA, SOX, PCI.

**There are a lot more risks beyond compliance, including security.**
How do you keep an app that was written in 1995 secure from malware that is written in 2016? The answer is you can't. Malware writers are notorious for exploiting unpatched vulnerabilities.

You're also always one OS update away from having major application functionality blocked.

A legacy app usually requires a major re-architecting to get around issues like that.

Most likely the legacy platform won't support what you need.

# You could lose customers.

What happens when you give your customers an outdated solution that doesn't meet their needs?

Chances are, they'll find a new solution.

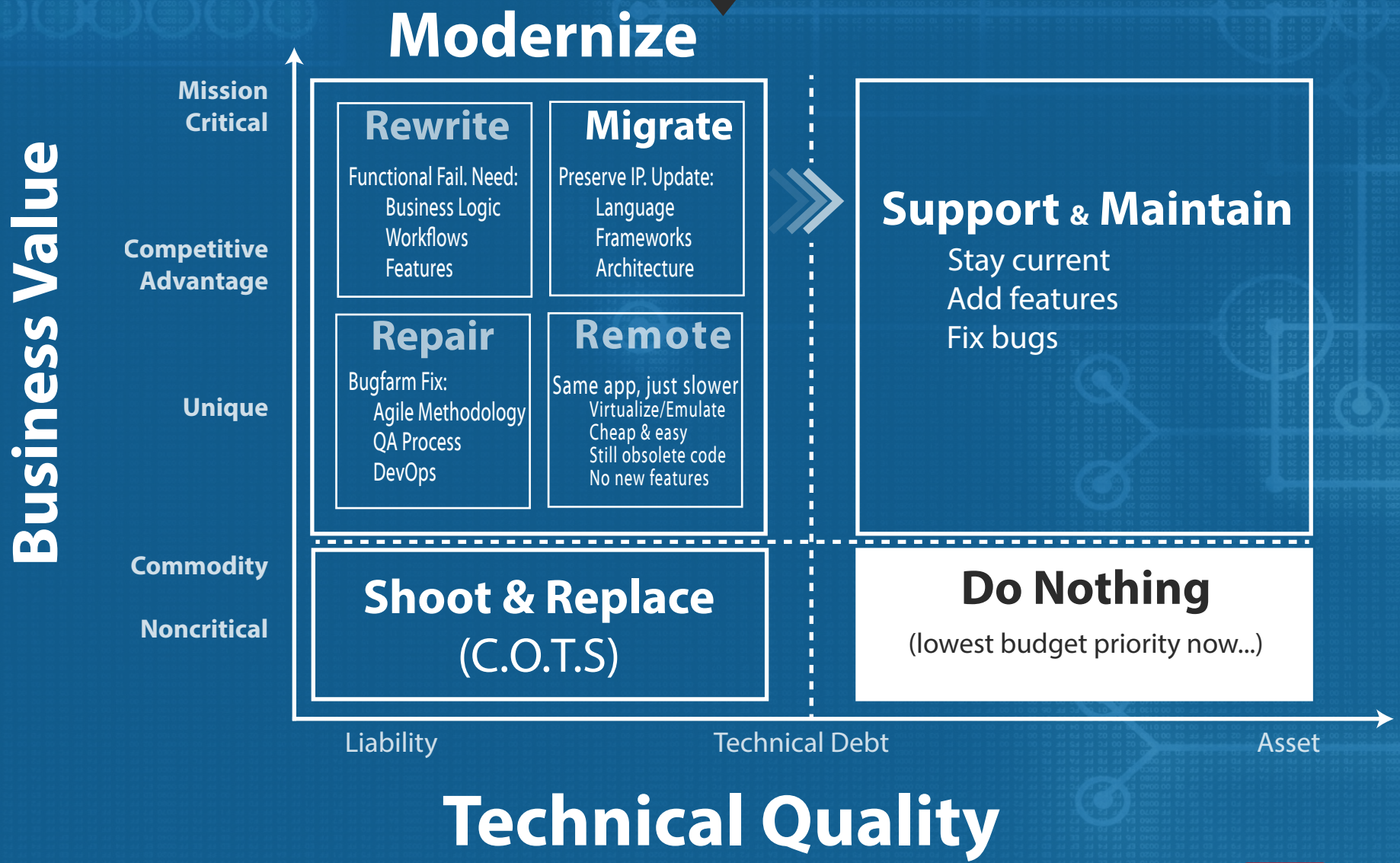
Entire industries are going through shakeups right now as a result.
This applies to both software vendors and IT departments.
If an IT department delivers outdated solutions, their users will begrudgingly use them.

But, as soon as they find a better solution to meet their needs, they'll use it in a heartbeat. This is the problem fueling the "Shadow IT" trend–a big issue in the business world.

# You do have options for legacy code


1. **Rewrite**
2. **Repair**
3. **Remote**
4. **Migrate:**
   **Modernize** - lift and shift valuable code off legacy platforms
5. **Do nothing** (kick the can down the road)
6. **Replace** with commercial off-the-shelf software
   (eg Salesforce, Dynamics, SAP) aka COTS
7. **Stay current:** add features, fix bugs

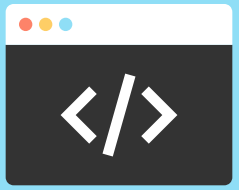
# Here's a **model** for how to think about your legacy apps - **The process and framework** for analyzing, prioritizing, and modernizing legacy application portfolios.



# Your CIO knows what's up.

- 74% of IT leaders say updating/modernizing key legacy apps is high or critical priority
- Over half view this as the TOP software issue
- Client/server legacy is now larger than COBOL
- $25B annually to maintain legacy VB apps
- Plus: PowerBuilder, Silverlight, classic ASP.NET

# Sometimes you have to rewrite from scratch.



**For strategic reasons:**

If your app IS your business (e.g. if you're Facebook or Amazon).

If you want to add complicated or complex features.

**For technical reasons:**

The code is bad, poorly written, unsalvageable.

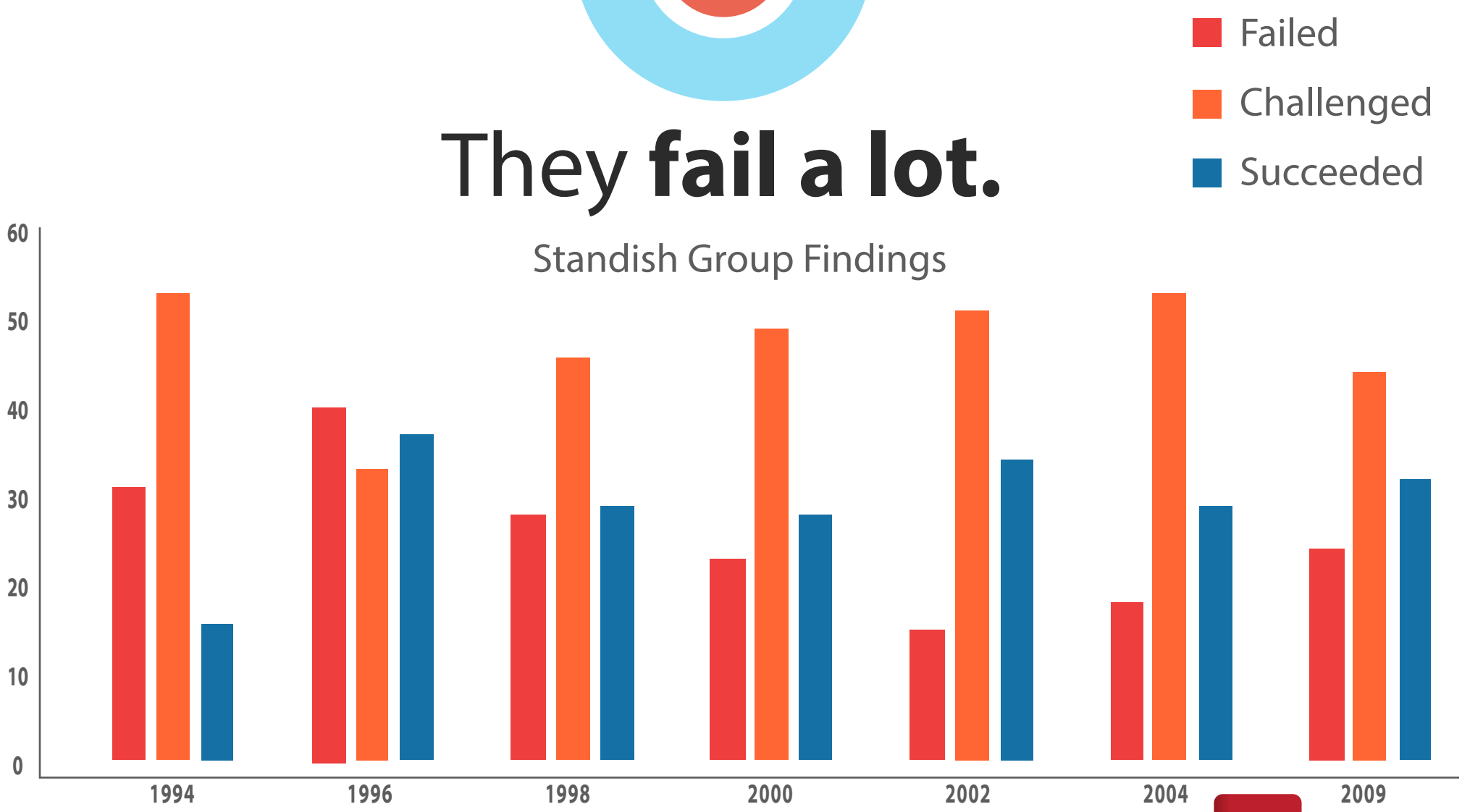
Your code depends on obsolete libraries that are no longer available.

# BUT...

before you decide to
rewrite from scratch,
you should understand
the **risks and costs.**

# The problem with manual rewrites.

## They **fail a lot.**



# But mostly, manual rewrites are expensive.

It’s going to cost between $6-$23 per line of code written.

And you’re going to get a bunch of bugs.
And **bugs are expensive.**

# Even the best developers create bugs

- Typically 20 – 50 new bugs per thousand lines of code (KLOC).
- A 100KLOC project will have 2000 – 5000 bugs to find, fix, and test.
- Many of those bugs will not be discovered before the product is delivered.
- What are the risks created by post-deployment defects?

# Where do the bugs come from?

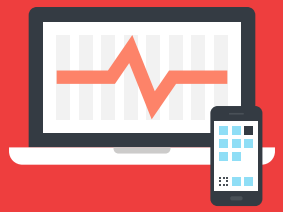
- Design errors make up 30-60%.
- Coding errors make up 25-40%.
- Design errors take an average of 8.5 hours to fix.
- Coding errors take an average of 3 hours to fix.
- Data errors take an average of 6.5 hours to fix.

# What conclusions should you draw?

- Manual rewrites are risky
- Software projects = Russian roulette
- It  might be ok, but the odds are terrible
- Bigger projects = more risk
- Too many unknown unknowns
- It can happen to anyone
- No one starts believing they will fail miserably
- A rewrite should be the last recourse, not the first.

# There **IS** a better way.

# Consider automated migration



Automated migration preserves functioning code without introducing new bugs and feature creep.

After migration, you can refactor poor code.

This keeps the system running, on a new platform, while you clean things up.

## You get:

- Predictable budget
- Predictable schedule
- **Covered in glory for a well-managed project**

# Pros and Cons

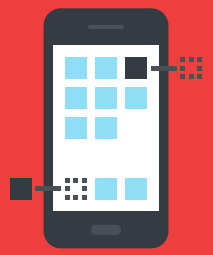
## Automated Migration

- Costs 75% less than rewriting
- Keeps business logic intact
- Does not introduce new bugs
- Completed in months, not years
- Removes feature creep from equation
- No requirements specs needed
- Creates real code as if you wrote it
- Preserves comments and names

## Rewrite

- 70% chance of failure or missing key goals
- Opens the door to feature creep
- Poor requirement specs are common
- 10-50 new bugs per KLOC
- Many bugs are never discovered before release
- Recoding key business logic can create critical errors

# New custom app development



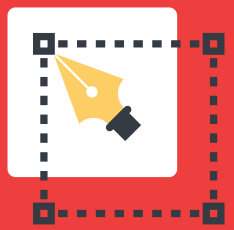
In developed (i.e. onshore*) markets, an average fully burdened developer costs $150K per year, writes 5000 lines of code per year at a cost of $30 per line of code.

**How did we do the math?**
($150K per year avg fully burdened developer cost) divided by (20 debugged lines of code / workday  times 250 workdays/year = 5000 LOC/year)  = $30 / line of code

There are also some big downsides to this approach but the main one is that 70% of new app dev projects fail.

*US, Canada, Europe

# Manual rewrite using developed **market resources**



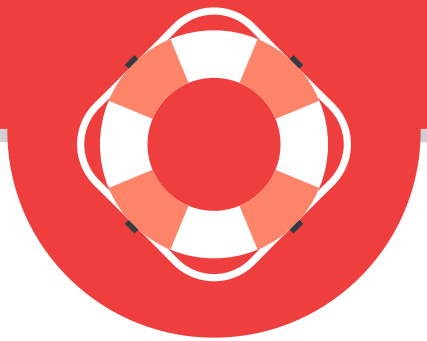
Using your existing application as a spec can cut the cost of a rewrite by half but it's still $15 per line of code.

**How did we do the math?**
Using your legacy application as the spec can double the productivity compared to greenfield app development = $15 / line of code

Hope you like bugs, because you're going to have a bunch of them: Developers write between 10-50 bugs per 1,000 lines of code. On 1M line app, you're looking at 10K to 50K regressions introduced in working business logic.

What's the cost of finding and fixing them all, and what's the risk to the business if you don't? Over time, up to 80% of the cost of software development is finding and fixing defects.

# Manual rewrite using offshore resources



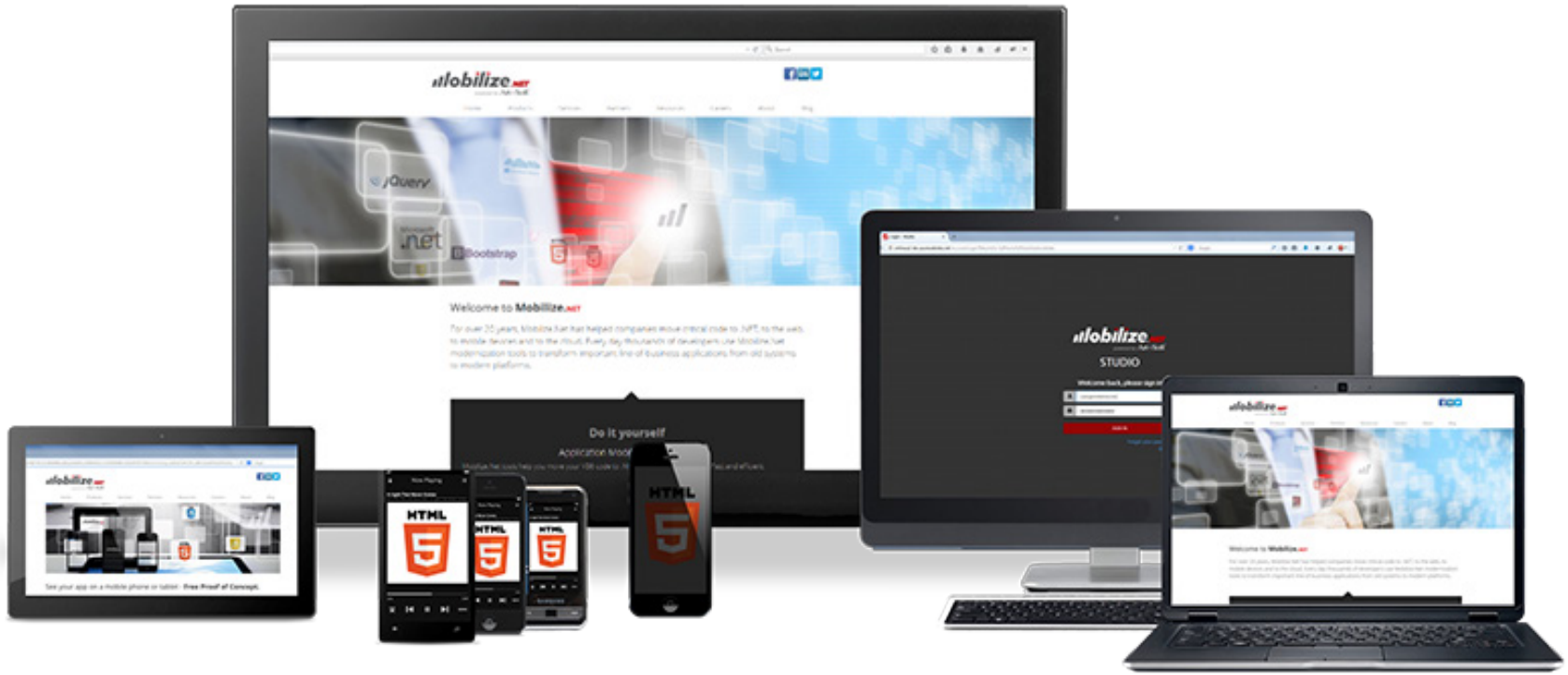
Using your existing application as guidance can cut the cost of a rewrite by half* and using offshore resources cuts costs too, but it's still $7.50 per line of code.

**How did we do the math?**
Using your legacy application as the specification, offshore resources at half the cost of US/Europe can at most cut cost in half again) = $7.50 / line of code

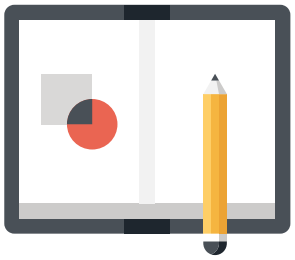
There's big offshore project risk.  Body shops of large numbers of young coders with less migration experience, language and time zone communication challenges have project failure rates around 80%…

*Careful analyses put the real number at 20-25% savings (source: Aberdeen Group and United Technologies)

Once the code conversion is complete, you can enhance the app with **new features, updated UI and other improvements**.

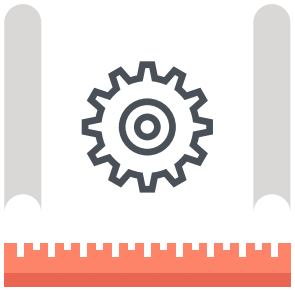


**The application can be re-factored and re-architected** via automation tools to make the new application multi-tier and cloud-enabled



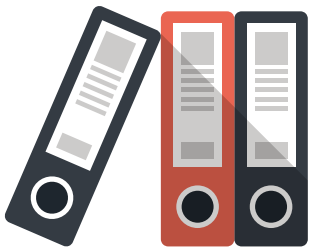
**Presentation Tier**

(User Interface)



**Logic Tier**

(Business rules and processes)



**Data Tier**

(Database storage and retrieval)

# The beauty of using an automated code conversion tool is you are guaranteed to succeed.

You will **get a full-functioning application** that works on the new platform.

# Every day thousands of **developers use Mobilize.Net** modernization tools to transform important line-of-business applications from old systems to modern platforms.

"We estimated our project, and Mobilize did it for 20% of our estimate for a ground-up rewrite. We got state of the art architecture that we can build on going forward."
**Chief Architect, F1000 company**

"When the Mobilize team walked me through the generated code in my new application, I was really impressed by the way they solved tough problems."
**Development Manager**

"We tried to rewrite this old VB6 app as a modern Web app twice and failed both times. Mobilize got it done in 6 months—I wish we had gone with them first."
**Project Manager, F500 company**

# Rewrite

# Migrate

High Risk

High Cost

Time-Consuming

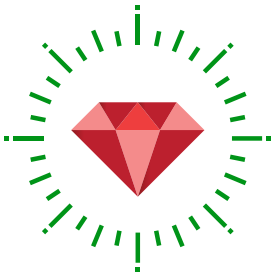
**BAD IDEA**

Low Risk

Low Cost

Speedy

**GOOD IDEA**



# Find your estimated costs

We've developed a calculator where you can **calculate** your development costs for a **rewrite vs a migration.**

**You can use real numbers from your projects.**

**See your savings here:**
http://mobilize.net/solution/rewrite-calculator

# Who's chosen automated migration over manual rewrite?

## Chances are good you know someone….


**Check out the full list at:**

http://www.mobilize.net/resources/customer-list

# Need more help?

You can also use our **assessment tool** to **help you figure out costs**:
http://mobilize.net/modernization-assessment-tool/

**Let a Mobilize.Net migration engineer help you** figure out how to convert your legacy application:
http://mobilize.net/talk-to-an-engineer/


**Migrate** to web, mobile & cloud

# Want to go to the movies?

We've got some stellar youtube videos to help you learn.

**Watch and learn about this customer's project:**
https://youtu.be/HcXc6KdBV4A

**Webifying Windows apps with Keith Pleas:**
https://youtu.be/CD-uVgwiBr8

# Mobilize.NET

## Your App. New. Again.


**www.mobilize.net**
**+1.425.609.8458**
**info@mobilize.net**

Mobilize.Net
10500 NE 8th St., Ste 725
Bellevue. WA 9804