Mobilize.NET

Calculating the cost of manual rewrites

Know before you go.

Your App. New. Again.

You've got an old legacy application and you're faced with the dilemma

Should I rewrite from scratch?

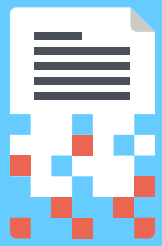
Should I keep trying to maintain it?

Before you take on a manual rewrite, you should understand the risks and costs.

Legacy code is expensive to keep around.

- Higher staffing costs for specialized talent.
- Higher maintenance costs for supported obsolete hardware.
- High maintenance fees for legacy vendors (i.e. high support contracts for Windows XP).
- Higher costs to add features or do required updates.
- Longer project cycles due to poor architecture, linear programming, bad data schemas, and more.
- Delays in responding to market changes and business cycles.

Sometimes it's good to rewrite from scratch



There might be strategic reasons:

- If your app IS your business (e.g. if you're Facebook or Amazon).
- If you want to do something fundamentally different with the app (add complicated or complex features).

And there might be technical reasons:

- The code is bad, poorly written, unsalvageable.
- Your code depends on obsolete libraries that are no longer available.

The problem with manual rewrites

Typical Project Results for Full Manual Rewrites*

HIGH FAILURE RATE

- 70% of manual software rewrites fail*

HIGH COST

- Rewrite costs 4 times more than migration

HUGE DEFECT RATE

- Developers write 10-50 bugs per KLOC
- In a 1 million line project, developers will write 10-50K bugs

FEATURE CREEP KILLS MANUAL REWRITES

- Projects overcome by too many features
- Requirements keep changing

But mostly, manual rewrites are expensive.

- It's going to cost between $6-$23 per line of code written.
- And you're going to write a bunch of bugs.
- You should figure out the costs before you attempt a full. manual rewrite.

Before you can figure out costs, you need to understand some definitions.

There are 3 common ways to predict the effort required to develop a software program.

Function Points (FP)

A unit of measurement to express the amount of business functionality a software program provides to a user.
The cost (in dollars or hours) of a single unit is calculated from past projects.

Source Lines of Code (SLOC):

Measures the size of a program by counting number of lines in the source code.

COCOMO (Constructive Cost Model)

Uses formulas to calculate costs based on SLOC.

Function Point estimating has pros and cons.

Pros:

Lots of support for FP because it requires thorough planning and results in a bulletproof plan.

Concentrates on functionality vs implementation details so it's focused on end user outcomes.

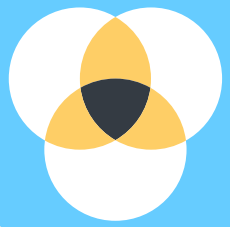
Helps to drive an accurate estimation of effort.

Independent of programming language or style/terseness.

Cons:

It's a lot of work, hard to do if you lack experience, and there's no way to automate it.

So does SLOC



Pros:

Automation tools make it simple to count lines of code.

It's an objective measure vs any other method.

Cons:

Doesn't take into account code complexity.

Can be misleading depending on the quality of code.

Varies by programming language (high level vs low level).

As does COCOMO

Pros:

Based on historical data.

Widely accepted.

Cons:

Based on analyzing waterfall projects from before 1981.

Doesn't factor in modern patterns or methodologies.

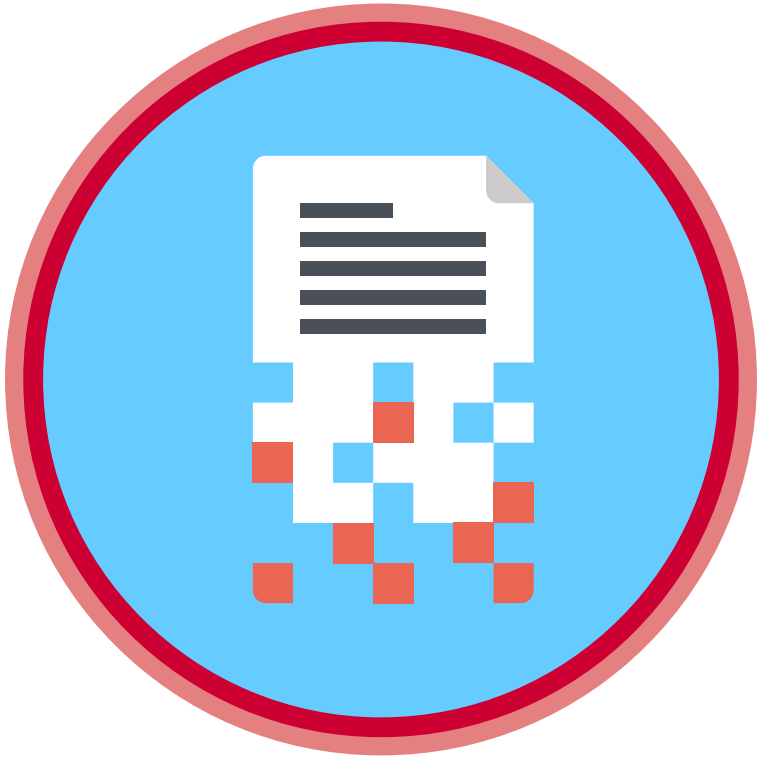
Which one should you use?

Bottom line: they're all great but flawed.

It's not possible to accurately predict software development costs for complex projects.

But you can at least get some order of magnitude estimates.

What's the relationship between FP and SLOC?



Low level languages (e.g. Assembly) require many lines of code for one Function Point.

High level languages (SQL, Powerbuilder) require few lines of code for one Function Point.

Ratio of lines of code to function points is a reflection of the level of language development (high or low).

For example,

The average lines of code* it takes to build a Function Point in:

Macro Assembler = 119

Visual Basic = 42

SQL Server = 21

One developer month:

A developer can develop 5 to 9.25 FPs per month.

1 Function Point = approx. 54 SLOC in C#

How do FP productivity numbers compare to SLOC?

Industry data by lines of code:

A world-class developer (e.g. Facebook or Google senior engineer) will write 50 LOC per developer-day.

A regular developer will write 10 LOC per developer-day.

This includes everything from requirements definition, specifications, coding, bug fixing, testing, and documentation.

Most work in programming is not actual coding.

What about COCOMO?

The COCOMO algorithm tells you three things:

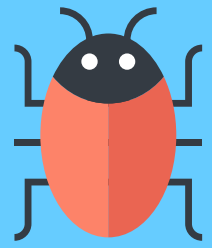
Person-months to complete (“effort applied”).

Calendar months to complete (“development time”).

Number of engineers (“people required”).

Uses SLOC and standardized calculations to complete.

Now let's talk about developer productivity and defects.



There's a lot of interesting data on this topic and it's pretty sobering stuff:

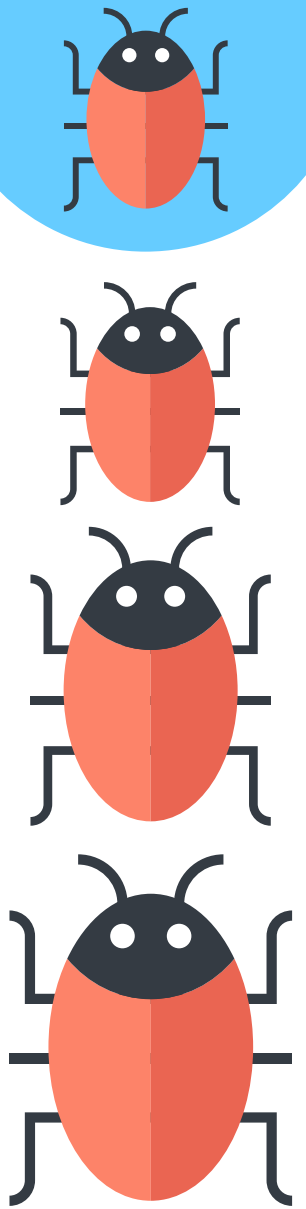
On average, developers write 4.5 defects per Function Point.

On average, 1.2 defects per FP make it into the final delivered application.

Best-in-class delivery is 1-3 defects per FP.

Unfortunately, US devs average 4-6 defects per FP.

You’re going to ship a lot of bugs

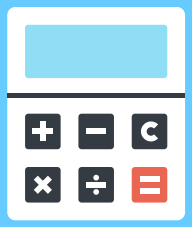


If you write a project with a thousand Function.

Points, you’re going to ship 1200 bugs.

Best-in-class developers will write fewer bugs, but not by much.

How does that break down?



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.

How many bugs will you write?

Typically, you'll write 20 – 50 new bugs per 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.

Whichever costing method you use, the bottom line is that everything takes longer than you estimate.

You’re going to write more bugs than you anticipate.

How do you figure out realistic costs for manual rewrites?

Here are some assumptions to get you started….

Inputs	Description and sources
Average LOC written or changed per developer per day	**20** Industry average from low of 10 to high of 50 (McConnell: Code Complete
Total developer work days per year	**250** Assumes 10 days off, no additional sick or vacation days)
Fully burdened cost of one developer per year	**$150,000** Include all taxes, vacation, insurance, office space, IT support, PC, software, etc.
Maximum potential savings of offshore development	**15%** Best savings is 40%, worst is less than 10%, likely is 15%: Source: Deloitte, multiple research reports
Potential cost savings of rewrite vs. new development	**20%** Reductions come from better requirements, existing use and test cases, existing staff experience
New defects introduced per KLOC touched	**25** Industry averages run from 10 to 50, Sources: Deloitte, Lotus, Microsoft
Developer hours per defect to find, fix, test	**5.00** Industry averages run from 3 to 9.
Total project KLOC	**100,000** lines of C# code would be considered a medium-large project
Total function points (FP)	**1,852** Based on C# ratio of 54 SLOC per FP. See: www.qsm.com/resources/function-point-languages-table
Average FPs per developer per day	**0.30** Industry averages run from 0.24 to 0.44
Average defects per FP	**5** US industry averages are 4-6. See: http://sqgne.org/presentations/2011-12/Jones-Sep-2011.pdf

…which result in these estimates

Calculations

Software Development using LOC estimates:

Lines of code per developer per year	5,000
Total developer-months to design / develop new application similar size	416.7
Estimated number of regression defects introduced by rewrite	2,500
Average cost per LOC for on-shore new development	$30.00
Best case scenario cost per LOC offshore development	$25.50
Rewrite cost per LOC domestic rewrite	$24.00
Rewrite cost per LOC offshore rewrite (optimum)	$20.40
Estimated rewrite cost on-shore	$2,400,000
Cost of defect mitigation before ship (on-shore)	$937,500
Estimated rewrite cost off-shore	$2,040,000
Cost of defect mitigation before ship (off-shore)	$796,875

Software Development using Function Point estimates:

FPs per developer per month	$6.25
Total person months to complete	$296.3
Estimated cost per FP on-shore	$2,000
Cost per FP on-shore with rewrite efficiency factored in	$1,600
Estimated regressions / defects introduced	$1,852
Estimated rewrite on-shore	$2,962,963.0
Cost of defect mitigation before ship (on-shore)	$694,444
Estimated rewrite off-shore	2,518,518.5
Cost of defect mitigation before ship (off-shore)	$590,278

And for COCOMO

Effort Applied (Person months)	302.14
Development Time (months)	21.90
People Required	$14
Cost	$3,776,776

Which should you believe?

None is perfect.

Bottom line: expect your total costs to be somewhere between the high and low in the different models.

What are your alternatives to starting from scratch?

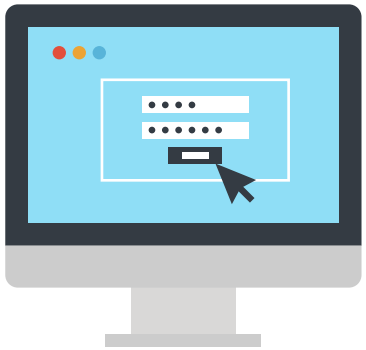
Automated software code conversion gives you the best of both worlds, costs 80% less money and is 4X faster.

Code conversion via automation tools enables you to reuse the existing functionality without starting over from scratch.

Because you don't have to re-invent the wheel:

- **Costs are lower**
- **You need less time**
- **Your risk is lower**

No new bugs are introduced.



Doesn't require re-training because UI is the same (or similar).

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

Three Tier Architecture

Presentation Tier
(User Interface)

Logic Tier
(Business rules and processes)

Data Tier
(Database storage and retrival)

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

Why use automated code conversion? Guaranteed success!

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

Calculate the real costs of your project

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

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

If you really want to go deep on rewrite statistics:

www.qsm.com/resources/function-point-languages-table

http://sqgne.org/presentations/2011-12/Jones-Sep-2011.pdf

www.compaid.com/caiinternet/ezine/bundschuh-est.pdf

www.ifpug.org


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 98004