What successful conversion/rewrite of software have you done? [closed]

Asked 15 years, 10 months ago Modified 5 years, 3 months ago

Viewed 788 times

8

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 5 years ago.

Improve this question

What successful conversion/rewrite have you done of software you were involved with? What where the languages and framework involved in the process? How large was the software in question? Finally what is the top one or two thing you learned from being involved with the process.

This is related to this question

refactoring code-migration

Share

Improve this question

Follow

edited May 23, 2017 at 11:55

community wiki
6 revs, 4 users 62%
RS Conley

17 Answers

Sorted by: Highest score (default)

3

I'm going for "most abstruse" here:

- Ported an 8080 simulator written in FORTRAN 77 from a DECSystem-10 running TOPS-10 to an IBM 4381 mainframe running VM/CMS.

Share Improve this answer

Follow

answered Feb 26, 2009 at 20:47

community wiki
anon

2

I rewrote 20,000 lines of Perl to use "use strict" in every file. I had to add "my" everywhere it was needed and I had to fix the bugs that were uncovered during the process.

The biggest thing I learned from doing this is, "It always takes longer than you think."

I had to get it done all at once overnight so that the other coders would not be writing new, unfixed code at the same time. I thought it would go quickly, but it didn't, and I was still hacking on it at 6 AM the next morning.

I did get it complete and checked in before everyone else started work though!

Share Improve this answer Follow

answered Feb 26, 2009 at 20:47

community wiki
Zan Lynx

1

I rewrote a large java web application to an ASP.Net application for a realty company for various reasons.

The biggest thing I learned is that, no matter how trivial the feature the original system had, if it's not in the second system, the client thinks the rewrite is a failure. Expectation management is everything when writing the new system.

This is the biggest reason rewrites are so hard: it seems so easy to the client ("Just re-do what I already have and add a few things.").

Share Improve this answer Follow

answered Feb 26, 2009 at 20:52

community wiki
Mark A Johnson

1

The coolest one for me, I think, was the port of MAME to the iPod. It was a great learning experience with embedded hardware, and I got to work with a lot of great people. Official site.

Share Improve this answer Follow

edited Feb 27, 2009 at 20:00

community wiki
2 revs, 2 users 82%
Alex Fort

0

I am doing a rewrite of an Inhouse Project managment system to a more standard MVC model. Its in the LAMP stack (PHP) and i am close to the 1st milestone.

The things i have learned from that currently is how simple the program feels at the beginning and i tried to not add complexity until i have to.

Example is that i programmed all the functionality first (like i was an admin user) and then when that is sorted out, add the complexity of having restrictions (user levels etc)

Share Improve this answer Follow

answered Feb 26, 2009 at 20:50

community wiki
Ólafur Waage

Not successful yet then... all the best! – MarkJ Mar 9, 2009 at 13:31

0

I ported/redesigned/rewrote a 30,000-line MS-DOS C++ program into a similar-length but much more fully-featured and usable Java Swing program.

I learned never to take another job involving C++ or Java.

Share Improve this answer Follow

answered Feb 26, 2009 at 20:50

community wiki
Ken

How's your new Cobol assignment going? – AShelly Feb 27, 2009 at 0:01

You misread. I said I would *not* take another job where I had to muck with crappy programming languages. :-) And it was a fantastic decision. – Ken Feb 27, 2009 at 2:23

0

I ported a client server Powerbuilder app, a couple of hundred screens worth, into an ASP.NET app (C#).

Due to performance and maintainability issues, I had over the previous year moved a ton of embedded SQL out of Powerbuilder scripts and into stored procedures.

Although this would make a lot of you wince, having a lot of business logic in the database, it mean the Powerbuilder app was relatively "light" and when we built the .Net front end, it could take advantage of the SQL codebase and have a lot of functionality already built and tested.

Not saying I'd recommend building apps that way, but it certainly worked to our advantage in this instance.

Share Improve this answer Follow

answered Feb 26, 2009 at 20:51

community wiki
MikeW

0

We had a code generation tool in our application framework that was used to read in text-based data files, About 20 other applications made use of it.

We wanted to make use of XML data files instead of structured text-based files. The original code was quite outdated and difficult to maintain. We replaced this tool by a combination of XSLT scripts and a utility library. For the utility library we could make use of some code in the old tool.

The result was that all 20 applications could now make use of either the obsolete text based file format or the new XML based format. We also delivered a conversion-generation tool that converted old data files to new XML data files.

After bringing out one or two release we have now decided that we will no longer support the old text based format and everybody is able to convert their data to XML.

We did hardly have to do manual conversions,

Share Improve this answer

Follow

answered Feb 26, 2009 at 20:51

Roalt

8,420 ● 8 ● 43 ● 57

0

Converted the main company app from pre-standard C++ to standard C++. We had a multimillion dollar sale contingent on making it work on AIX, and after looking at it we decided that converting to standard C++ was going to be just as easy as converting to IBM's traditional C++.

I don't know the line count, but the source code ran to hundreds of megabytes.

We used standard Unix tools to do this, including vi and the assorted compilers.

It took a few months. Most of the fixes were simple ones, caught by the compiler and almost mechanically fixed.

Some of them were much more complicated.

I think my main takeaway was: Don't get too awfully clever with code in a language that hasn't been standardized yet, or is likely to have things change in unexpected ways. We had to do a lot of digging in some of the ingenious adaptations/abuses of C++ streams.

Share Improve this answer Follow

answered Feb 26, 2009 at 20:54

community wiki
David Thornley

0

Ten years ago I managed a team that converted a CAD system from DOS into Windows. The DOS version used home-brew libraries for graphics drawing, the Windows version used MFC. The software was about 70.000 lines of C code at the time of the conversion. The most important thing we learned in the process is the power of abstraction. All device-specific non-portable routines were isolated in a few files. It was therefore relatively easy to substitute the calls to the DOS-based library that would draw by directly accessing the frame buffer with Windows API calls. Similarly, for input we just substituted the event loop that checked for keyboard and mouse events, with the corresponding Windows event loop. We continued our policy of isolating the non-portable (this time Windows) code from the rest of the system, but we have not yet

found this particularly useful. Perhaps one day we will port the system to Mac OS X and be thankful again.

Share Improve this answer Follow

answered Feb 26, 2009 at 20:56

community wiki
Diomidis Spinellis

0

Several. But I mention one.

It was a performance modeling tool. Part delphi 1, part turbo pascal. It needed a rewrite else it was not going to survive. So we started as a team of 2, but only me survived to the end. And I was ready before the deadline ;-).

Several things we did:

- Make it multimodel. The original had lots of globals. I removed them all and multi model was easy to adapt.
- Extended error messages. Click on a message and get the help.
- Lots of graphs and diagrams. All clickable to drill down.
- Simulation. Change parameters over time and see how long the current configuration was enough.

We really made this one clean and it paid back heavily in the end. Such a big learning experience.

Share Improve this answer Follow

answered Feb 26, 2009 at 21:09

community wiki
Toon Krijthe

0

Re-wrote a system for a company that processes legal invoices - the original system was a VB monstrosity that had no idea of good OO principles - everything was mixed together. The HTML did SQL, and the SQL wrote HTML. A large part of it was a custom rules engine that used something like XML for the rules.

Two teams did the re-write, which took about 9 months. One team did the web front end and the backend workflow, while the other team (that I was on) re-wrote the rules engine. The new system was written in C#, and was done test-first. Adding new rules to the system when we were done was dirt simple, and it was all testable. Along the way we did things like convert the company from VSS to SVN, implement continuous integration, automate the deployment, and teach the other developers how to do TDD and other Scrum/XP practices.

Managing expectations was crucial through the project. Having a customer that was savvy about software was

very helpful.

Having a mix of large scale (end-to-end) tests along with comprehensive unit and integration tests helped tons.

Share Improve this answer Follow

answered Feb 27, 2009 at 6:43

community wiki
SteveDonie

0

Converted vBulletin which is written in PHP into C#/Asp.NET. I'm pretty familiar with both languages, but PHP is the hands down the winner for building that software. The biggest pain in the rear was needing to do a C# equivalent of PHP's eval() for calling the templates.

It was my first challenge in trying to do a conversion. I learned that I need more experience with C# and that writing it from scratch is just the easier route sometimes.

Share Improve this answer Follow

answered Feb 27, 2009 at 6:57

community wiki
William Holroyd

I converted a dynamical build-process completely written in Perl to a C#/.Net solution using a workflow-engine a

0

co-worker had developed (which was still in beta - so I had to do some refinements). That gave me the oppertunity to add fail-safe and fail-over functionality to the build process.

Before you ask - no - the microsoft workflow-foundation could not be used since you cannot dynamically change a process during its runtime.

What I learned:

- to hate the Perl-developer
- process-optimization using a wf-engine
- fail-safe and fail-over strategies
- some C# tweaks ;)

In the end it covered about 5k - 6k (including the wf-engine) LoC origin from 3 200 LoC Perl-files. But it was fun - and far better in the end ;)

Share Improve this answer Follow

answered Feb 27, 2009 at 9:34

community wiki
Gambrinus

0

Converting theoretically portable C code into theoretically portable C code across architectures to support a hardware change that saves the company X dollars per unit.

The size varies - this is a common need, and I've done small and large projects.

I learned to write more portable C code. Elegance is great, but when it comes right down to it the compiler takes care of performance, and the code should be as simple and portable as possible.

Share Improve this answer

Follow

edited Feb 27, 2009 at 20:05

community wiki
2 revs, 2 users 92%
Adam Davis

0

Ported a simulation written in Fortran 77 (despite being written in the 90s) to C/Java because the original only worked on small data sets. I learned to love big O notation after several times of explaining why just moving the entire data table into memory at the start of the program was not going to scale.

Share Improve this answer

Follow

answered Feb 27, 2009 at 20:14

community wiki
Kim Reece

0

Migrating the B-2 Stealth Bomber mission software from JOVIAL to C. 100% fully automated conversion. Seriously!

Main lesson: *using configurable automated conversion tools is a huge win*.

See DMS Software Reengineering Toolkit.

Share Improve this answer Follow

answered Nov 25, 2009 at 7:05

community wiki
Ira Baxter