5 Reasons

Manual Rewrites Fail

A publication of
Mobilize.NET
powered by Art in Soft

INTRO

Did you know that 70% of software manual rewrites fail?

Of 3,555 projects from 2003 to 2012 that had labor costs of at least $10 million, ***only 6.4% were successful****. The Standish data showed that 52% of the large projects were "challenged," meaning they were over budget, behind schedule or didn't meet user expectations. The remaining* ***41.4% were failures*** *-- they were either abandoned or started anew from scratch.*

Computerworld, Oct 21, 2013

TOP 5 REASONS MANUAL REWRITES FAIL:

1. Poorly defined requirements.
2. Cost overruns.
3. Software defects.
4. Unpredictable schedule.
5. Skill gap.

Poorly Defined Requirements

- IT projects often ignore strategy and stakeholders and focus on budget and schedule. This often results in late or ever-changing requirements. Every year thousands of projects get **derailed because of bad requirements**.

- What can you do to avoid this pitfall? Focus on the strategic planning before the project. Establish a clear view of the project's strategic value to the business. The plan MUST go beyond technical requirements. Build a robust business case and **maintain focus** on the business objectives.

Cost Overruns

Exceeding the budget is a major failure point for big software rewrite projects. A recent study showed that of 1400+ projects surveyed, on average **27% were over budget**. Even worse, they had a cost overrun of over 200% and were late by almost 70%. These were important enterprise projects which could cause huge failures and bring down companies.

What are the main causes of cost overruns? It all goes back to the planning phase…. Again. When you're thinking about large technology projects, you need to thoroughly (and honestly) articulate your organization's **strategic business goals**, readiness, and stakeholder willingness to engage.

"Why Your IT Project May Be Riskier Than You Think", HBR, Sept 2011

Software Defects

- A project that has 100K lines of new code will have between **2000-5000 bugs** to find, fix, and test. Most of the bugs won't be discovered prior to project delivery. The later a bug is discovered, the higher the cost. For a medium bug fix (requiring 100 development hours), it takes seven times the effort to fix versus getting it right the first time. That means a defect costs $40K to fix in production vs $1,500 in development.*

- How to avoid bugs? Software defects are a fact of life. However, the earlier you identify and fix them, the more likely your project will succeed. Include business, IT and development in one project team to ensure smooth planning and do testing in early stages.

Cost of Rework Models for Agile and Non-Agile Projects, voke Research, July 11, 2012

Unpredictable Schedule

- Software scheduling is notoriously unreliable. Over one-third of software projects experienced time overruns of 200 to 300%. The **average overrun is 222%** of the original time estimate. For large companies, the average is 230%; for medium companies, the average is 202%; and for small companies, the average is 239%.*

- How do you build a reliable schedule? Start with smaller time frames and deliver against frequent milestones. Shorter time frames result in an iterative process to design, prototype, develop, test, and deploy small elements. Set clear and precise objectives for each milestone. Small projects tend to be **less complex** and you're much more likely to deliver. Making the projects simpler is a worthwhile endeavor because complexity causes confusion and increased cost.

*Standish Group, 2010

Skill Gap

- When a team lacks the knowledge and skills needed to do the work properly, quality levels and productivity suffer and the risk of serious errors or omissions rises fast. Skill issues are consistently cited as a **failure point** for software development projects and account for a significant portion of the failure rate.*

- If there is one ingredient that most effectively increases the chance of project success, it is expertise.

*McKinsey-Oxford study on reference-class forecasting for IT projects

What Should You Do?

- You have alternatives to manual rewrites. 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 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 and no re-training is required 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.

Better and better…

- 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 get a full-functioning application that runs on the new platform.

- We've developed a calculator where you can calculate your development costs. You can use real numbers from your projects:
  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/