



Proper Software Engineering An Executive Primer

Choosing the best development firm for your software project can be a daunting task, so it's important to understand how great software is actually developed. Like any complex product, software should be engineered to meet business requirements, built according to best practices, and tested to ensure quality. This white paper describes proper software engineering methods and suggests a framework for selecting the right software development firm.

Introduction

Need special software for your business? No problem. Just hire a local programmer to copy some code samples and stitch them together into the perfect app. Right?

Modern technology makes it easy to build good software, so find the cheapest techie out there and start coding. Or even better, send the requirements to an offshore code “factory”, then sit back and wait for the new software to arrive. Make sense? Sort of. If you want to buy a bridge, you’ll always find someone who’s willing to sell.

Computer programming is a profession with very low barriers to entry. To get started, all you have to do is buy a computer, learn HTML and some other basic languages, put up a website, and voila! you’re in the software business. An instant expert. But the romantic notion of irreverent, all-knowing geek prodigies who write killer apps “on the cheap” is a Hollywood fiction that’s best left in the movies. When it comes to production, software is no different than any other product: teamwork and clear communication are paramount—and yes, you get what you pay for.

Software systems now play such a critical role in business that bad software (or even a lack of adequate software) can seriously impact the bottom line. Just ask folks at the European Space Agency, who watched a billion dollars go up in smoke when their first (unmanned) Ariane 5 rocket was destroyed by a software glitch 40 seconds after liftoff.²

Software projects are notoriously difficult to implement properly and many fail. Some, like Ariane 5, fail spectacularly. In 2005, the director of the FBI appeared before a U.S. Senate subcommittee to explain the failure of the Virtual Case File software project—and its \$104 million price tag.³ In 2008, Overstock.com had to restate five and half years of earnings because their ERP system wasn’t implemented properly. According to CEO Patrick Byrne, they “didn’t hook up some of the accounting wiring.”⁴

And consider the mission critical software that underpins financial trading on world markets: would it be acceptable if the software failed to process one percent of trades? Point one percent? Ask yourself, how effective do I need my business systems to be? The answer is usually obvious.

Software development is a complicated endeavour and business leaders are right to be cautious about handing over projects to unqualified developers. Indeed, software is too crucial to business success for it to be treated casually or as anything less than a complex product that must be carefully developed to meet specific needs.

But how can you tell if a software development firm is qualified to handle your project? How do the best developers produce great software? What approach do they take? What principles do they follow? What distinguishes great software developers from hacks? One word: engineering.

This white paper describes the basic principles and characteristics of proper software engineering, and then explains how to choose the right development firm for your project.

A software system is like a city—an intricate network of highways and hostleries, of back roads and buildings...Some software systems are lucky, created through thoughtful design from experienced architects. They are structured with a sense of elegance and balance...Others are not so lucky, and are essentially software settlements that grew up around the accidental gathering of some code...What kind of software city would you rather construct?

- Beautiful Architecture¹

The 'root cause' of the loss of the spacecraft was the failed translation of English units into metric units in a segment of ground-based, navigation-related mission software...

- Arthur Stephenson
Chairman of the
Mars Climate Orbiter
Mission Failure
Investigation Board⁵



...Wal-Mart invested in the infrastructure—the systems technology and business processes—to really drive cost efficiencies in its business and drive prices down for consumers, Kmart just didn't make that same type of investment. So, for Kmart, I think a key issue is the lack of competitive infrastructure, systems technology, and supply chain management that would give it the ability to play on a level field with Wal-Mart.

- Sandra Skrovan
Retail Forward⁶

Due to a wide range of options now available for how a system may be configured and connected, carefully designing the architecture becomes very important. It is during the architecture design where choices like using some type of middleware, or some type of back end database, or some type of server, or some type of security component are made.

- Pankaj Jalote
An Integrated Approach
to Software Engineering⁷

Businesses Need Great Software

Good software is “table stakes” but great software is a competitive advantage. To remain competitive in their industry, companies need:

- High quality, dependable software that users want and can easily use.
- Software that powers business processes, improves productivity, and enables competitive advantage.
- Software systems that meet enterprise-grade requirements, like performance and scalability.

These are the business drivers that have spurred on development and adoption of proper software engineering practices.

Consequences of Sloppy Software Development

Developing software without applying sound engineering principles can have many negative consequences, including:

- Blown budgets and unforeseen expenses.
- Missed deadlines.
- Buggy software that hampers your ability to do business.
- Software architecture that doesn't support current and evolving business needs.
- Decreased customer satisfaction.
- Higher maintenance costs.
- Higher risk.

Software Engineering

There is a huge difference between *engineering* software and merely *writing* software. This distinction can be illustrated by an analogy to the construction industry.

Working alone, a skilled tradesperson can perhaps build a small, simple house but is unable to construct a larger, more complex building without help. An entire team of professionals is needed to build a large structure: tradespeople, civil engineers, an architect, and a project manager. In addition, building permits must be obtained, machinery must be leased, and good quality, highly specialized construction materials must be sourced.

Similarly, there are limits to what a single, though talented, programmer can accomplish. Great software is actually built by multidisciplinary teams, not by any one person.



Maintain a “big picture” view of the whole project (the technical and business sides) and keep it in front of the team.

- Gary Chin
Agile Project Management⁸

Creating a simple (and efficient) design of a large system can be an extremely complex task that requires good engineering judgment.

- Pankaj Jalote
An Integrated Approach to Software Engineering⁹

Flexible systems are needed because we cannot predict the future. Because organizations change over time, complete knowledge of a system’s requirements, in the traditional sense, is necessarily imperfect. Some requirements lie in the future, and they are unknowable at the time the software system is designed or being built.

- Johnson, etc
Flexible Software Design¹⁰

As a rule, a software development team must include three core disciplines:

- **Engineering**
Conceptualization, specification, and architecture, where software engineers determine *what* needs to be built and then describe *how* to build it.
- **Programming**
Implementation, where skilled programmers write code according to best practices and testers validate it by using a proven quality control process.
- **Management**
Collaboration, where a skilled project manager keeps the team on track and in close communication with business leaders and users.

How It Works

Software engineering is such a complex field that it’s impossible to capture all of its diversity in just a few words. However, successful development firms draw upon certain key principles, proven techniques, and helpful tools to engineer great software for their clients. Here are some key indicators that proper software engineering methods are being used on a project:

- **Iterative development methodology**
For most business systems, teams should use Agile development to quickly produce releasable software while staying in tune with changing business priorities. Exceptions are projects with fixed requirements, like nuclear power plants.
- **Proper project management**
Likewise, project managers should use Agile to keep their projects within budget and on schedule, improve software quality, and respond to changing business needs.
- **Transparency and communication**
Development teams should work closely with businesspeople to ensure that software is always aligned with current business strategy and market forces. By speaking the language of the business, the development team can avoid the “software first” trap, a narrow fixation on technical aspects of the software that excludes an awareness of what the business is all about.
- **Proper architecture**
Software engineers should evaluate architectural concerns based on experience, best practices, and project constraints. For simplicity, these concerns are often divided into several categories or views: a logical view (end-user functionality), a development view (components, services, interfaces), a process view (performance and integration), a physical view (servers, networks, storage), and scenarios (use cases).
- **Proper evaluation/use of open source & proprietary vendor products**
Software engineers should investigate existing software products that can be incorporated into a design faster, easier, and cheaper than if code were developed from scratch. Why reinvent the wheel?



The Google User Experience team aims to create designs that are useful, fast, simple, engaging, innovative, universal, profitable, beautiful, trustworthy, and personable. Achieving a harmonious balance of these ten principles is a constant challenge. A product that gets the balance right is "Googley" – and will satisfy and delight people all over the world.

- Google.com¹¹

The only practical way to manage regression testing is to automate it. People become numb from running the same manual tests many times and seeing the same test results all the time. It becomes too easy to overlook errors...

- Steve McConnell
Code Complete¹²

As you increase the load, whether it's measured by the number of users or by the size of the data sets, you may see completely different behaviors with your software...For this reason, it's very necessary to conduct some level of scalability testing...

- Henry Liu
Software Performance
& Scalability¹³

- **Proper design**

Software engineers should have a good understanding of the domain and technology before planning the software. This way, they can make informed choices about methods like object-oriented design, service-oriented architecture, and data modelling.

- **User experience (UX) design**

Development teams should incorporate UX design into every aspect of a software project. Most people equate user experience with the user interface but the look and feel of the software is only one component of UX. User experience design ensures that software is as simple as possible and works the way you'd expect it to.

- **Development tools**

Development teams should use tools to speed the production of error-free software. Tools include Web and application frameworks that reduce the need for authoring new code, data modeling tools that tailor software to business processes, rapid prototyping tools, and tools for testing code coverage, performance, and Web service.

- **Automated builds**

Development teams should have infrastructure in place to automatically build code and generate tests, repetitive development tasks where human error can have a significant negative impact on software quality.

- **Test-driven development (TDD)**

Development teams should employ TDD, a rapid iterative process for generating simple code that works. In TDD, the team creates a test that captures the intent of the business requirement and then writes just enough code to pass the test—no more, no less.

- **Automated regression testing**

Development teams should be able to automatically verify that they've fixed existing bugs without introducing new ones. Automated regression testing makes development predictable because changes can be made with the assurance that errors will be detected.

- **Quality assurance (QA) metrics**

Similar to quality assurance in a manufacturing environment, software QA processes should be used to evaluate the product based on meaningful criteria. Software should be tested for a number of characteristics, including:

- **Security**
Are there any flaws in the code that would permit unauthorized access to the system?
- **Scalability**
How well does the software handle increasing numbers of transactions, users, etc?
- **Testability**
Is it easy to run meaningful tests on the software?
- **Performance**
How quickly does the software run?



The principle of testing for robustness is to be sure that the system is solid, and that unexpected events or input data will not cause it to perform in a way that is unacceptable. The definition of unacceptable behavior will differ from system to system...

- Guy Lecky-Thompson
Corporate Software
Project Management¹⁴

- *Extensibility*
How easily can software functionality be expanded or modified?
- *Flexibility*
To what extent can the software be adapted by the users themselves (without the need for IT intervention)?
- *Robustness*
How reliably does the software run in the real world?
- *Maintainability*
How easily can the software be maintained?

Does Proper Software Engineering Cost More?

No! Properly engineered software is a necessary investment and is ultimately cheaper than improvised solutions, which incur a “technology debt” that sooner or later comes due. When you consider that most of an application’s cost is in maintenance, it becomes obvious that poorly developed systems, though initially cheaper, actually end up costing much more in the long run.

Choosing a Software Development Firm

Great development firms are magnets for gifted software engineers and programmers, who are lured by the opportunity to work on a variety of challenging projects—in contrast to the vanilla corporate experience offered by most large companies. Hiring a nimble firm with a smart, passionate team translates into higher quality software (lower maintenance costs) and faster development (faster ROI) for clients.

If your company intends to commission a software project, you should look for a development firm that:

- Follows proper software engineering practices.
- Uses Agile development and project management techniques.
- Operates locally and knows the value of face-to-face service.
- Understands *your* big picture.
- Selects appropriate technology solutions, regardless of brand or affiliation.
- Makes your project their top priority.
- Has a track record of success for your type of project.
- Is passionate about technology.

...the time spent in coding is a small percentage of the total software cost, while testing and maintenance consume the major percentage. Thus, it should be clear that the goal during coding should not be to reduce the implementation cost, but the goal should be to reduce the cost of later phases...

- Pankaj Jalote
An Integrated Approach
to Software Engineering¹⁵



References

1. Spinellis, Diomidis, and Georgios Gousios (editors). *Beautiful Architecture: Leading Thinkers Reveal the Hidden Beauty in Software Design*. (Sebastopol, California: O'Reilly Media, 2009), 25.
2. Nuseibeh, Bashar. "Ariane 5: Who Dunit?", *IEEE Software* 14 (3): 15–16, May 1997, doi:10.1109/MS.1997.589224.
3. Shore, James. "Guest View: It's Not Too Late to Learn", www.sdtimes.com, August 15, 2005.
4. Kanaracus, Chris. "Update: Overstock.com restates earnings, cites ERP implementation", Computerworld.com, October 27, 2008.
5. Isbell, Douglas, and Don Savage. "Mars Climate Orbiter Failure Board Releases Report, Numerous NASA Actions Underway in Response", Release 99-134, mars.jpl.nasa.gov, November 10, 1999.
6. Turner, Marcia Layton. *Kmart's 10 Deadly Sins*. (Hoboken, New Jersey: John Wiley & Sons, 2003), 126.
7. Jalote, Pankaj. *An Integrated Approach to Software Engineering, Third Edition*. (New York: Springer Science+Business Media, 2005), 159.
8. Chin, Gary. *Agile Project Management: How to Succeed in the Face of Changing Project Requirements*. (New York: AMACOM, 2004), 78.
9. Jalote, *An Integrated Approach to Software Engineering*, 249.
10. Johnson, Bruce, Walter W. Woolfolk, Robert Miller, and Cindy Johnson. *Flexible Software Design: Systems Development For Changing Requirements*. (Boca Raton, Florida: Auerbach Publications, 2005), 13.
11. Google, www.google.com/corporate/ux.html.
12. McConnell, Steve. *Code Complete: A Practical Handbook of Software Construction*. (Redmond: Microsoft Press, 1993), 618.
13. Liu, Henry H. *Software Performance and Scalability: A Quantitative Approach*. (Hoboken, New Jersey: John Wiley & Sons, 2009), 75.
14. Lecky-Thompson, Guy W. *Corporate Software Project Management*. (Hingham, Massachusetts: Charles River Media, 2005), 299.
15. Jalote, *An Integrated Approach to Software Engineering*, 391.





Architech Solutions is a Toronto-based technology consulting and software development firm. We design and build powerful, user-centred systems that work. We're agile, disciplined, and passionate about delivering for our clients.

Architech Solutions
3 Church Street
Suite 602
Toronto, Ontario
M5E 1M2

Phone: (416) 607-5618
Fax: (416) 352-1768

To book a free on-site Discovery Workshop led by our team of consultants, e-mail info@architech.ca or visit us at www.architech.ca

