mrc

(New) Rules for Modern Web Application Development

Strategic principles for building future-proof web applications for the modern business

Introduction

Driven largely by recent trends, web application development is undergoing a massive shift. According to Gartner:

"Application development (AD) is going through a metamorphism. We are seeing the birth of a new AD that spearheads business innovation. Technologies such as mobile and cloud, innovations in the web, large changes in methodologies like agile and continuous release, and the growing importance of a user experience platform are driving renewed interest in AD. We need strategies and capabilities that balance architecture innovation with legacy integration so that application architects can deliver powerful and flexible systems."

The fact is, web application development standards are rapidly evolving, yet many companies are stuck developing applications based on outdated standards and technology. They build applications on proprietary platforms. They build applications that don't integrate with web services or adapt to mobile devices. They build applications that limit their company's flexibility. In doing so, they unwittingly hurt their company's ability to compete in an increasingly competitive business world.

"Application development (AD) is going through a metamorphism. We are seeing the birth of a new AD that spearheads business innovation."

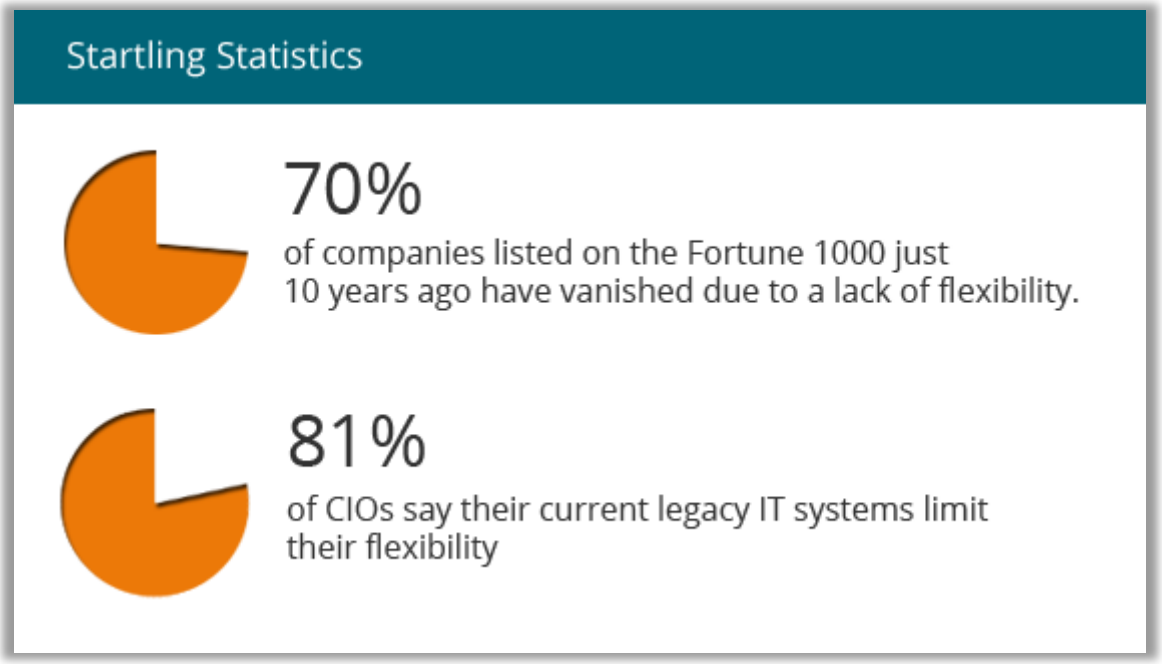
Why are modern web applications and development strategies so important in today's business world? Here are two big reasons:

1. **They bring flexibility**
   According to **Forrester Research**, 70% of the companies listed on the Fortune 1,000 list ten years ago have now vanished. The reason: an inability to adapt to change. As modern technology evolves, we're seeing a shift in the business world. In the age of the web, business agility trumps size. More and more, the ability to adapt to change is critical to a company's success.

2. **They give you a competitive advantage**
   A recent Forrester study claims that custom software development will set companies apart in the future. As software continuously takes a larger role in business, the ability to create custom applications that adapt to an ever-changing technology landscape gives companies a massive strategic advantage.

However, while the importance of modern web application development is growing, many companies are stuck in the past. According to a recent survey, 81% of CIOs say their legacy IT systems pose the biggest hurdle in moving to the cloud. In a world where flexibility trumps size, they're tied to inflexible systems that limit their options and keep them from adopting new features and capabilities.

Companies like these will struggle to adapt, unless they bring flexibility to their current systems and adopt modern development methods.



Inflexibility kills companies, yet most companies remain inflexible

The big questions: What new rules must developers follow when building modern web applications? What strategic development principles help you deliver modern, sustainable web applications that prepare your company for the future?

This paper will help you answer those questions. It outlines modern web application development guidelines you must follow for developing successful, modern web applications that adapt to the ever-changing tech landscape. To start things off, we'll focus on the most important point of all: application architecture.

1. Separate your architecture

In the past, old procedural programming methods often resulted in large, monolithic applications. They were independent of other applications, and every function of the application was programmed into the code itself. This led to maintenance nightmares, as the code became more and more convoluted over time…often resulting in what's commonly known as "spaghetti code."

These days, modern web applications avoid these issues using tiered architecture, where the database, business logic, and presentation are loosely coupled. While each tier works together to make a single application, they also operate independent of the other tiers. Building web applications in this manner may require more upfront effort, but offers multiple business advantages. Here are a few examples:

1. **Security**
   Separating your architecture lets you enforce security differently on each tier. For instance, the database tier may require more security than the presentation tier. Separating your architecture gives you full control over security on each tier.

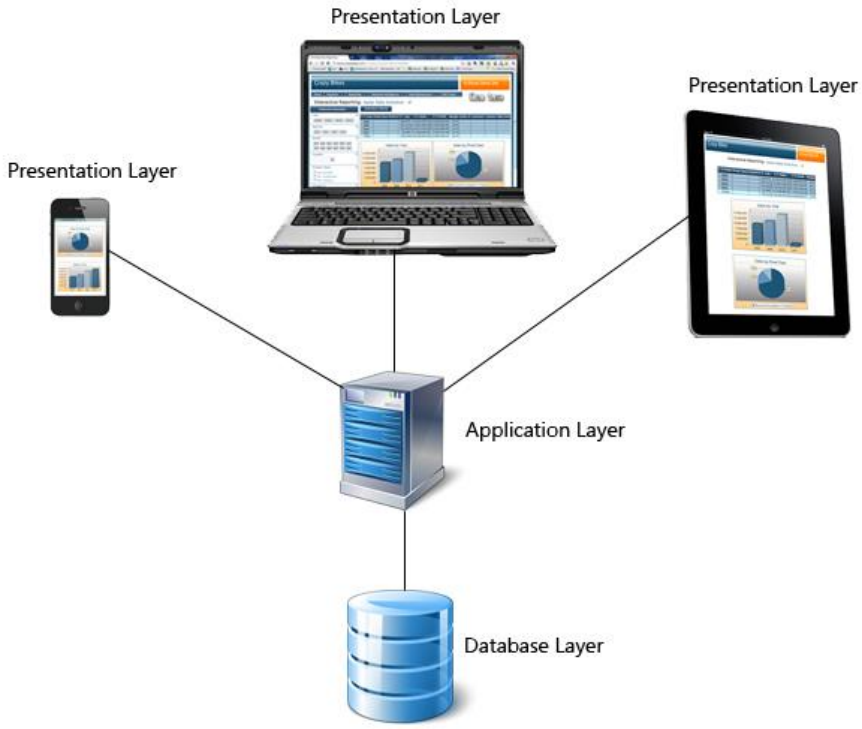
   How important is application architecture to overall security? In a word: Essential. "Application architects probably have more impact on an application's security than anyone else," says Rohit Sethi, VP of **SD Elements**. "A single architectural decision can have major ramifications on how secure an application is inherently, and perhaps more importantly how expensive it is to fix a vulnerability."

   "A single architectural decision can have major ramifications on how secure an application is inherently, and perhaps more importantly how expensive it is to fix a vulnerability."

   Sethi goes on to share a cautionary story on the value of good architecture: "I once reviewed a web-based accounting application where the developers decided to implement their own JavaScript presentation framework with a service-based backend. In order to speed up performance, the JavaScript framework was responsible for making all authorization decisions. The development team was horrified to find out that it was trivial for attackers to bypass JavaScript security controls and access the backend services directly. This effectively meant anyone who could reach their web server could perform any accounting activity – even without credentials. Their architecture made it very difficult to fix because it effectively meant changing the way every transaction occurred and sacrificing at least some of the JavaScript performance benefits on which they marketed their software. The fix was a very expensive, multi-year effort."

2. **Scalability**
   This approach lets you scale individual tiers as needed, meaning your applications will grow with your company. For instance, you could scale up your database tier with database clustering without touching the other two tiers.

3. **Flexibility**
   This approach lets you easily modify or enhance individual tiers, without touching the others. For instance, the image below illustrates how an n-Tier architected application could be adapted to meet growing mobile demands. Without replacing the application, a company could simply develop separate presentation layers for different devices—complete with logic to identify the user's browser and display the appropriate layer. This approach lets a company address mobile devices without rebuilding their apps, or building separate apps.



Example of how tiered architecture might apply to mobile

Tyler Wassell, Software Development Manager at **mrc**, explains the importance of application architecture: "When IT leaders look at their portfolio of business applications, they should be asking a few important questions: Can we move these applications to the cloud if necessary? Are they secure? Can they be accessed from mobile devices? Can we cost effectively maintain, modify, and customize them as our business needs changes? Do they scale? If the answer to any of these question is "no", attention should be given to the underlying software architecture."

Key take-away: Separating your web application architecture into separate tiers is a key element in creating modern web applications. It dramatically improves the application's security, its ability to adapt over time, and leaves you with applications that grow with your business.

2. Plan for mobile platform fragmentation

Despite their relatively recent addition to the marketplace, smartphone adoption in the U.S. has already surpassed 50%. This rapid growth makes smartphones the fastest growing trend in history.

This rapid growth also puts an added strain on developers. In an increasingly mobile world, applications must now work across personal computers, tablets, and smartphones. As if that's not difficult enough, these devices vary in screen size and span multiple platforms.

Jason Salsiccia, Project Manager at **OnSIP** shares a past story that highlights the risks of ignoring mobile: "Prior to OnSIP, I worked on a website redesign project, and developing a mobile solution was not a part of the plan," he says. "The reason for this was budget and time constraints. When launch time came, almost all of the feedback we got was that the site was unusable. It presented a truly awful user experience on smart phones and tablets, which now more than ever are the devices from which most people experience the Internet. The question I should have asked was, 'What is our mobile strategy? Is what we're building useful if people need a large monitor to use it?' We chose to ignore the necessity of mobile devices to satisfy our constraints, but in the end we realized we didn't have anything viable. Fixing the mistake cost even more time and money than it would have if we had a mobile strategy planned from the beginning."

However, while developing for mobile devices is absolutely critical, it doesn't stop there. Not only must your applications support all current platforms and screen sizes, they must support all future ones as well. The problem: No one knows what the future holds, and the mobile landscape is incredibly volatile. A popular mobile platform today may not exist in three years, while a platform that doesn't exist today may very well be popular in 5 years.

"We chose to ignore the necessity of mobile devices to satisfy our constraints, but in the end we realized we didn't have anything viable. Fixing the mistake cost even more time and money than it would have if we had a mobile strategy planned from the beginning."

How can you develop web applications that adapt to an ever-increasing number of screen sizes and platforms? The best approach: Use cross-platform design methods like responsive or adaptive design. Here's the difference between each one:

- **Responsive design** uses fluid grids and CSS3 media queries to adapt the app to various screen sizes and types. You can find examples of responsive design across the web, as it's widely used in web site design. For instance, **TheNextWeb** site is a great example of a responsive site. You'll notice that if you resize your browser, the site changes to fit the new size.

- **Adaptive design** delivers a different HTML page depending on the device. For instance, a single web application may have 3 separate presentation (HTML) layers: One for PCs, one for tablets, and one for smartphones. Logic placed in the application identifies the user's device and displays the correct layer. For example, visit this **web app** from different devices and notice how it looks and acts different.

Key take-away: Before you begin any modern web application development project, ask yourself, "How will this work on different screen sizes and platforms?" This question may make the initial development project more complex, but it will position your company for the future.

3. Design for usability

While always important, usability has now become an absolute necessity. After all, user expectations have changed. They expect the well-designed, intuitive interfaces they receive on their smartphones and consumer applications. These days, if the application confuses the user, they'll find another option.

"My approach has really had to change since I am building a consumer-facing set of apps," says Alex Genadinik, founder and developer of **Problemio.com**. "Instead of focusing on the server and database, and optimizing them to scale, I had to shift focus on first, observing user behavior as much as possible, and making sure all the buttons and screen elements are A/B tested to ensure the best usability. So I had to shift focus from the backend technologies to focus very intensely and with great details on the front-end and usability. Because if there is no usability, the backend tech becomes useless."

Now, usability is a vague term. What does it look like? More importantly: How can you build applications with good usability? Here are three steps to take that will help you improve application usability:

1. Talk to users early and often

The first step towards usability: Understand what the users want. Talk to them throughout the entire process, not just at the beginning.

"Even though the time to launch has decreased dramatically, the competition in the web space has increased even more dramatically so the need to be efficient with development time is more important than ever," says Michael Granados, Co-founder and the head of Product Design at **Medko**. "Talking to your users early is a very important step; it helps you understand what they want so you know what to actually develop. And talking to them often throughout the initial development process is equally important so you understand if you are interpreting their "wants" correctly in your product."

"Talking to your users early is a very important step; it helps you understand what they want so you know what to actually develop. And talking to them often throughout the initial development process is equally important so you understand if you are interpreting their "wants" correctly in your product."

2. Develop in small, agile cycles

Miscommunication combined with long development cycles leads to disasters. Developers often spend months building what (they think) the users want, only to realize the users wanted something completely different. In short, it's a big waste of time.

"In past decades, development cycles could be months, sometimes even years," explains Granados. "Today, it's not uncommon for new code to be pushed every couple of weeks. Quick development cycles help ensure that the team is focused on only the most important priorities so you can quickly begin testing features with your early adopters. Fast development cycles also prevent teams from wasting months developing a feature that nobody ends up using."

3. Measure Everything

Usability goes beyond simply asking your users what they want. You must also understand how they use the completed app. In this way, you will understand how to improve and modify the app going forward.

"Talking to your users is a great place to start development but once you understand what they told you they want, you need to use analytics and cold hard numbers to see if their behavior matches their words," explains Granados. "There are many free and affordable analytics solutions available today and they can be configured into an app in just minutes. Measuring user behavior will give you indisputable data on whether or not your app is being successful."

Key take-away: The user experience is becoming more and more important to web application development. Users now have other options, and are less likely to use an application that's confusing. Focus your application development on usability, and solving your user's problems.

4. Build for future integration

In the past, business applications typically stored and accessed data on a single platform. Now, with the rise of mobile and the growing use of Software as a Service (SaaS), platforms have exploded. Modern applications must now interact with an ever-increasing number of software platforms--moving the need for integration to the forefront.

According to Dave West, former Forrester Research president and research director, "Application integration problems are a top reason why businesses -- and their enterprise architects and project managers -- can't deliver business innovation at the speed demanded by customers using all these application platforms."

Unfortunately, integration is often ignored if it's not an initial requirement for the project. This is a mistake. "Most software apps become Frankensites/Frankenapps because they get so many tools and functions added to them down the road that their appearance and usability suffers dramatically," explains Jeff Kear, Owner of **My Wedding Workbook, LLC**. "When you are first scoping out your application, try to brainstorm all the potential uses that the software may have in the next 5-7 years. You probably won't add many of these tools upfront, but if you have figured out a potential home for them in the application and built the framework so you can somewhat easily integrate them after the fact, then you won't be kicking yourself down the road."

The fact is, you don't know what new application or service you'll need to integrate with in a few years. However, if you plan for future integration when developing your applications, you'll save time and effort in the future, when you actually need the integration.

Key take-away: Building applications with integration in mind starts with the right foundation. Avoid proprietary tools or languages and build applications on open foundations. This will prepare your company for future integration and won't lock you to any single platform or vendor.

5. Prepare for the cloud

Maybe your company hosts your own applications in-house, and has no plans for the cloud. Maybe you prefer to keep your applications and data in-house and manage everything yourself. Don't worry, this guide won't try to convince you that one path is better than the other.

However, as cloud hosting becomes more reliable, the cloud cannot be ignored. Who knows what the future holds? Maybe a new manager will come in and decide that everything must move to the cloud. Maybe a current manager will change their tune on the cloud. When/if that happens, your applications must be ready. They must deploy anywhere—on premise or in the cloud.

"The fact is, new management, vendor changes, and technological innovation can quickly shift IT priorities and the technologies used to accomplish them," explains Wassell. "But while deployment platforms and databases may change, the right software architecture can greatly reduce the need to repurchase and/or rewrite legacy software. Is the software being introduced into the organization today ready to meet the changing needs of tomorrow? Can it be moved to a new platform in-house or to the cloud with relative ease?"

The danger of not designing your applications for portability? As Wassell explains, it further locks your company down to outdated systems, limiting your future flexibility. "In other situations, a lack of forward architecture planning may keep an IT organization stuck on outdated systems and unable to meet these changing organizational priorities. Staying the course and developing with outdated technologies on legacy architecture essentially means investing further in a legacy system that is now incrementally more difficult to move away from when the business need arises in the future."

"A lack of forward architecture planning may keep an IT organization stuck on outdated systems and unable to meet these changing organizational priorities."

Key take-away: "Using an n-tier architecture to separate out the various components of software applications as well as a language that can be ported to a new platform is key," explains Wassell. "Doing so can increase the likelihood that when your application needs to be moved off premise to the cloud, or adapted to use a new database, resources can be spent on the shifting business priorities and not on rewriting/repurchasing existing software."

6. Use well-established technology

Developers love trying new things. They love experimenting with new libraries, frameworks, languages, tools, etc… (and they certainly have plenty to choose from these days). There's nothing wrong with that.

However, it becomes dangerous when developers carry that over into business application development. Here are two reasons why this is a big risk:

1. **It might not take off**
   If you build web applications using brand new technology, you run the risk of that technology fizzling out. What if it never takes off? Now you're stuck with an application built with little-used technology that may or may not work well with existing technologies.

2. **It complicates maintenance**
   Building an application with a little-used language limits a company's future options. If the developer ever leaves, they'll have trouble finding another developer to support that application.

Key take-away: While experimentation with the latest and greatest technology is fine, stick with established technologies when building business applications. This insulates your company from getting stuck with applications that are difficult to support, or that don't work well with existing technologies.

Summary

Chances are, you don't want to replace your business applications every few years. You'd like to build applications that last. You want to build applications that grow with your company and adapt to changing technology.

However, technology is evolving faster than ever, which makes business application development even more challenging. If built incorrectly, a modern application today might be outdated in just a few short years. Obviously, businesses can't afford to replace their applications every few years.

How can you build applications that remain relevant for years to come? These six guidelines will put you well on your way to building long-lasting, flexible applications that position your company for the future:

1. **Separate your architecture**: Long-lasting applications start with good architecture. Separating your web application architecture into separate tiers dramatically improves the application's adaptability, and leaves you with applications that grow with your business.

2. **Plan for mobile platform fragmentation**: The mobile platform landscape is incredibly volatile. The truth is, nobody knows which platforms or devices will exist in the next few years…but, you must prepare for them. While planning for different screen sizes and platforms may complicate the initial development project, it will position your company for the future.

3. **Design for usability**: Driven largely by the rise of well-designed mobile and cloud-based consumer applications, user expectations are rising. A confusing or poorly designed interface will not only frustrate users, it may even drive them away. Focus your application development on usability, and you'll save yourself the time and expense of re-creating an app that nobody uses.

4. **Build for integration**: As application platform options expand, integration becomes a key element of web application development. These days, if you want applications that last well into the future, they must communicate with applications across an increasingly diverse number of platforms.

5. **Prepare for the cloud**: While your company may not plan on ever moving to a cloud host, it's good practice to prepare for any scenario. Build applications that easily port anywhere—on premise or to the cloud.

6. **Use well-established technology**: When building business applications, stick with established technologies. Choosing well-supported and widely used technologies simplifies future maintenance and insulates your company from future compatibility issues.

As technology evolves at an ever-increasing rate, flexibility becomes absolutely critical to a company's success. Companies that follow the above guidelines lay the foundation for modern, flexible applications that easily adapt to future changes—whatever they might be.

About mrc

Michaels, ross & cole, ltd. (mrc) is a global software company which specializes in web application development software. Headquartered in Lombard, IL, and established in 1981, mrc has offices in the U.S. and the UK. mrc offers award-winning development software, as well as consulting, mentoring, and training services.

mrc's development platform, m-Power, automates web and mobile web application development. Using a unique build process and application architecture, m-Power develops applications that automatically look and feel like native applications across all platforms (PCs/tablets/smartphones). Visit www.mrc-productivity.com to learn more about mrc or m-Power. Visit mrc's Cup of Joe Blog to read more about a wide range of development and IT leadership topics.

References

Page 6 image
Gonzalez, Francisco. *Clouds on Northeastern Paris*. Digital image. *Flickr*. N.p., 21 Apr. 2012. Web. <http://www.flickr.com/photos/franciscojgonzalez/7099881781/>.

Page 8 image
Patterson, Blake. *My IPhone Family Pile*. Digital image. *Flickr*. N.p., 8 July 2010. Web. <http://www.flickr.com/photos/blakespot/4773693893/>.

Page 9 image
Varlan, Horia. *Scattered Puzzle Pieces next to Solved Fragment*. Digital image. *Flickr*. N.p., 23 Oct. 2008. Web. <http://www.flickr.com/photos/horiavarlan/4273913228/>.