Important Topics in Software Strategy For Business

By Justin Tocci - August 27, 2015

www.workflowproducts.com
Workflow Products, LLC
7813 Harwood Road
North Richland Hills, TX 76180
817-503-9545

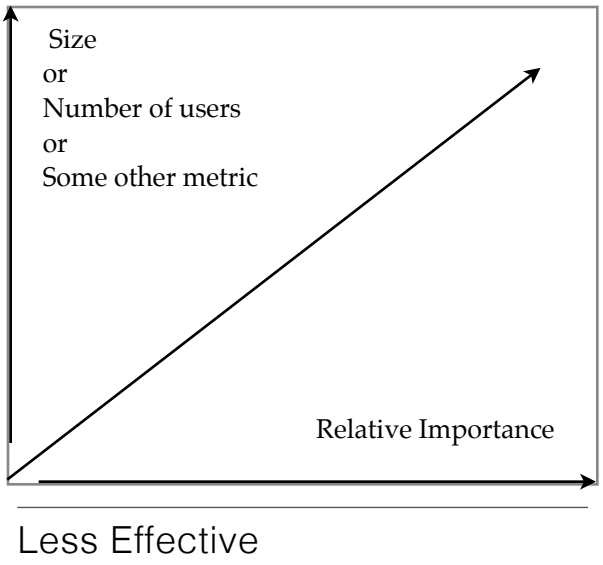
Table of Contents

Assessing the software landscape in an organization

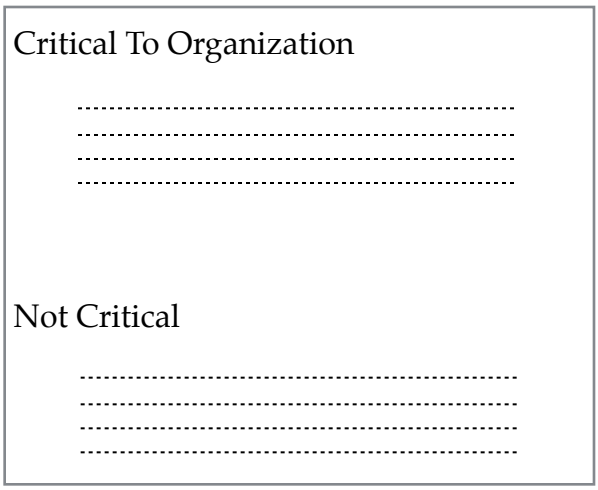
When looking for activities that should be handled by an application, we look for just one thing. **Documents that are updated regularly.** Any document that is updated regularly is not a simple document. Its a document that is being used as an application.

You may be tempted to evaluate a document that is updated regularly as not critical to the organization. This is normal. A document that is updated frequently may in fact be unnecessary or used to manage some trivial activity. This should be a judgement made by management based on the importance of the activity. Frequently, organizations will try to use other metrics to gauge importance and as a consequence, whether the activity should be brought into an application. This is discouraged because attempting to use metrics to determine importance does not reflect reality.

If you have two activities and one is managed by the whole company and the other by one department, it can easily seem as if the former is more important. In reality, every system either meets the bar of "critical" or it doesn't. As a result, its best to have management class activities as either "critical" or "not critical".



Less Effective



More Effective

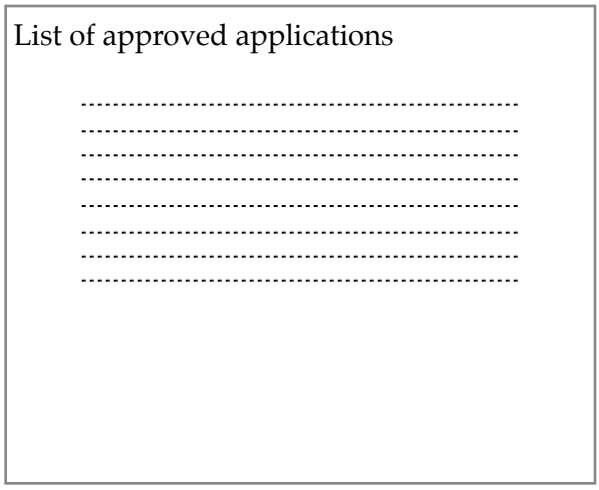
The Evolution Of Applications

New activities are introduced into an organization due to change. For the purposes of software management whether the source of change is internal or external is not important. What is important is to acknowledge that change will occur and with it new needs will be created and fulfilled. If your organization has a plan then the solutions will grow like a well kept garden. If not then they will grow according to the motivation and skills of users.

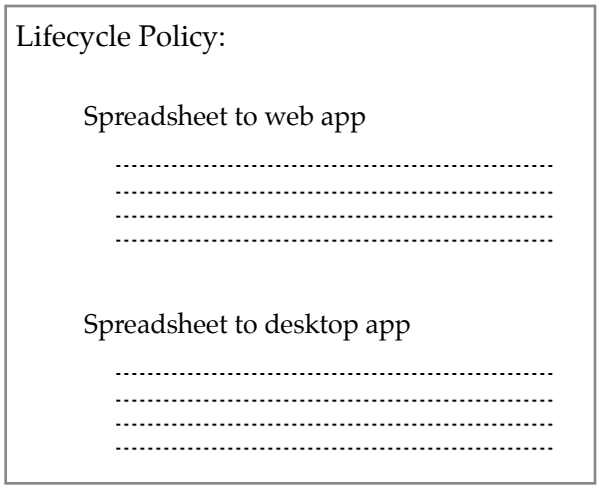
Management in collaboration with IT should take the time to approve applications for use by non-IT staff. Additionally, security and backup policy should be posted prominently. A clear policy stating where documents must be stored will prevent document loss.

All organizations will have a changing landscape of skill sets. Significant risk and waste can be avoided with a "lifecycle policy". A lifecycle policy is a simple document that enumerates organization policy for application activities. With approval, documents can be used as applications for non-critical activities. Other activities will start as a document and be reviewed periodically to determine if its time for the activity to be moved to the organization's chosen application platform.

For example, an employee is given a critical task which is new for the organization. The employee consults the Lifecycle Policy and possibly the IT department. As a result, a spreadsheet is designed. In the beginning, the design may not be completely suitable. Over time it is improved. Once the design is working and updating it has become routine, its time to reevaluate. Is the need still critical? If so, resources are applied and the spreadsheet is upgraded to a simple application. The app is now part of the organization's application suite and can take advantage of the organizations chosen application platform(s).



Minimum



Recommended

Application Platform Types

Applications can be classified into three broad categories, desktop, web and mobile. The differences are numerous and it can seem daunting to attempt evaluation. In order to start we'll make some broadly accurate generalizations.

Desktop applications excel compared to web applications in richness of user experience. Copy/paste, import and export, and other user interface tasks are typically superior on the desktop side. The reason is because web applications have a primitive toolkit. Developers have to exert great effort to provide rich interfaces and that is expensive. Technology is advancing in this area and desktop applications will begin to lose their advantage eventually.

Desktop applications are available in a wider range of solution cost profiles compared to web apps. The least expensive websites are more expensive than small desktop apps. Conversely, the most expensive web apps are much less expensive, on average, than the most expensive desktop solutions. Generally websites do tend to get more expensive over time. Part of this is due to the numerous tools needed to create a web app. Many web tools go out of fashion and finding appropriate developers can increase cost. Desktop application development is more robust over time due to a less complex architecture and less tools needed. A desktop application development tool will also typically have an upgrade path whereas many web technologies do not.

Web applications do not need to be installed and require less support than desktop applications. If your users are remote, a website can be lot easier to support. Sometimes your user profile dictates that you simply have to build a website. If you need to interface with customers directly a website may be your only option since remote support for a desktop application could quickly become prohibitively expensive.

Desktop Applications	Web Applications
• Solutions are typically easier to use per dollar spent. • Low cost compared to a minimal website. • Better for on-site solutions due to support, security and performance profiles.	• Generally more expensive for a minimal system. • Generally get more expensive over time. • Better security for remote users. • Better performance for remote users.

Mobile Applications

Mobile Applications are a category unto themselves. They can be classified into three types; native, web and hybrid. There are a large number of mobile platforms and there are many considerations when evaluating. In order to start we will again make some broadly accurate generalizations.

Native applications offer the richest experience on a mobile device. The better experience costs more because native applications are more time consuming to write. In order to get a native application for each mobile device type you can use one of the many cross platform development kits available. These paid software programs allow you to develop one app and release it to multiple device types at once. The rich experience of native applications covers a wide area of application types and a typical business application does not utilize even a small fraction of what native applications were designed to provide. As a result, native applications are most applicable to public corporations or organizations with regulatory requirements such as the Sarbanes–Oxley Act, ITAR, HIPPA etc...

Web applications can be written to work for desktop and mobile. For typical businesses, a web application can generally be written to work well regardless of what device people are using. Often, to save cost, the website will be written to serve desktop users first, then selected parts of the desktop application will be re-written for mobile devices.

Hybrid applications utilize a native shell to display a web application. Hybrid applications are used to add some native functionality without increasing the cost of the solution to the level of a native application. The common reasons that an organization would use a hybrid app for a business oriented application involve branding and other marketing activities. They are not normally used for internal applications.

- Less expensive than native apps.
- Offers the ability to extend your app with some native functionality if you have non-standard requirements.
- More expensive than a mobile enabled web application.

Mobile Hybrid

- Generally the least expensive option.
- Excellent feature coverage for business applications.
- Wide compatibility means no need to create a version for each device type.
- Due to possible incompatibilities you will want to do some testing on each platform you support.

Mobile Enabled Web Application

Desktop Application Platforms

	Desktop Platform	Minimum Initial Cost	Ongoing Cost	Risks
	Spreadsheet	low	low	data integrity, confidentiality, data loss, security
	MS Access, Document Mode, Single User	low	low	data loss, security
	MS Access, Document Mode, Mult-user	low	moderate	data loss, security, performance issues
	MS Access, Application Mode, PostgreSQL	< $1000	moderate	low
	MS Access, Database Mode, SQL Server	six figures	moderate	low
Advanced Platforms	Visual Studio and SQL Server	six figures	high	low
	Oracle, IBM DB2	seven figures	negotiated	low
	SAP	eight figures	negotiated	low

The above chart is short on detail and there are always exceptions but the overall profiles provided are based on very wide experience. If you look at the chart closely you'll see there is a sweet spot between the high risk and high cost solutions.

MS Access and PostgreSQL
Microsoft Access is the most popular database software in the world and has been for over a decade. Connecting Microsoft Access to a database removes virtually all risk of data loss and performance issues. The differences are driven by connecting MS Access to a database. Once Access is connected to a database, we essentially convert an Access document to an Access application.

The PostgreSQL security model is exceptional and has a better track record than MS SQL Server. Initial cost approaches zero while ongoing development cost is very low compared to the more advanced platforms.

Evaluating Desktop Application Platforms

There are many different ways to evaluate platforms and a comprehensive comparison is beyond the scope of this document. For our purposes, the important considerations for each platform should help you get started and if you need further explanation please contact us.

Documents and Spreadsheets

Recommended Use Cases

- Single use data gathering and reporting.
- Storing historical data for infrequent use.
- One time reports.
- Report prototyping.
- One-off presentations and charting.
- Reference or work instructions.
- Records of correspondence.
- Data that needs to be sent to one person.
- Data that is edited by one person.

Time to upgrade

- Reports are created regularly.
- Data is being transferred from one document or application to another document or application.
- Data is being entered or updated regularly (daily, weekly or monthly).
- Data is for use by several people.
- Data is updated by several people.
- You've experienced errors in data entry.
- Data has been lost.

Note that solutions created in documents and spreadsheets are often created by non-IT professionals. Managers, analysts, and administrators create these solutions without budgets or guidance as part of their job. These solutions would not make economic sense if IT professionals fulfilled them. Although solutions created by non-IT professionals might suffer from poor design, users should not be discouraged from fulfilling their own ad-hoc and short term needs. Instead, IT should support users by providing assistance such as help desk and training resources.

MS Access Document Mode

Using Microsoft Access in Document Mode is not recommended. Over the years Microsoft has added so many features to Access that investments in document mode can lead to some difficulty when its time to upgrade. In order to avoid that, it's recommended that developers have experience using Access with a database. If you have a proper developer, then its only pennies more to move on to Application Mode. In light of the small difference in cost but large difference in benefits, Document Mode is not recommended.

You may encounter a developer who wants to create a database in Document Mode as a first step toward Application Mode. This is very useful and can help keep costs down.

Evaluating Desktop Application Platforms

Microsoft Access and PostgreSQL

- Make data entry easier/faster.
- Validate data upon entry.
- Eliminate double entry.
- Create reports on demand from a single click.
- Display real-time metrics.
- Create, store and retrieve documents.
- Automatically distribute documents on demand or on schedule via email.
- Prompt employees to action.
- Any data-driven application, such as accounting software, can be recreated and then customized for your company.

Recommended Use Cases

- Data is covered by regulatory or other considerations such as the Sarbanes–Oxley Act, ITAR, HIPPA etc...
- Due to regulatory or other considerations, Employees must be treated as untrusted users.
- The organization is a public company and regulatory or other considerations require data to be treated with special requirements.

Time to upgrade

Access' popularity can be attributed to its speed, low cost and versatility. Professional developers can use Access to create very sophisticated solutions with maximum return on investment.

Access solutions require significantly less code, but more importantly, partially complete applications are always able to be run and tested. As a result a developer can create solutions more quickly than when using an advanced platform. In fact, advanced platform developers sometimes use Access for prototyping.

Access is sold with Microsoft Office but you don't need to purchase Access to use it for your organization. Only the developer is required to purchase a license. Meanwhile, Access solutions can easily create Word documents or Excel spreadsheets at the click of a button.

Access solutions are as rich and easy to use as any other application. Using Microsoft Access to create an application often causes lots of rich behavior to be included without the developer having to do any additional work. This keeps costs down and offers users a higher quality experience.

Advanced Platforms

Visual Studio, Microsoft SQL Server and other advanced platforms are all very expensive. If you're considering these platforms it should be because you have problems that advanced platforms were designed to solve. Obviously, if it isn't necessary to use an advanced platform then it makes fiscal sense to avoid it.

Although the usual evaluation strategies apply, when consulting with developers there are some unexpected pitfalls worth avoiding.

Not every developer is sufficiently experienced in Microsoft Access to answer questions. One can use Access for many years without ever becoming knowledgable about Application Mode. Consequently, a person can feel qualified to give information when in fact they are objectively unable to do so.

Developers that work with advanced platform may have insufficient experience with Access. It often happens that developers use Document Mode as a stepping stone to an advanced platform. Despite their advanced skills they may still be uninformed about Application Mode.

It can be harmful for a developer to spend significant time working with a different platform. Despite the lower productivity and higher cost, once a developer has become skilled in an advanced platform there are strong reasons not to 'go backwards' and use Access for a project. Advanced platforms require special skills acquired through lots of discipline and experience. Those skills can fade over time if left unused. A developer can have very strong motivations to bias the evaluation process.

If a consultant is equipped to serve $200K+ solutions, it cannot profitably manage opportunities worth far less than that. When consulting outside contractors as to whether requirements can be met by Microsoft Access and PostgreSQL, be sure to consult with an appropriate contractor. If the organization handles advanced platforms at all then they are unlikely to provide unbiased advice during the evaluation process.

Evaluating Web Application Platforms

A quick search of Wikipedia will result in over a dozen common languages and over a hundred web 'frameworks'.

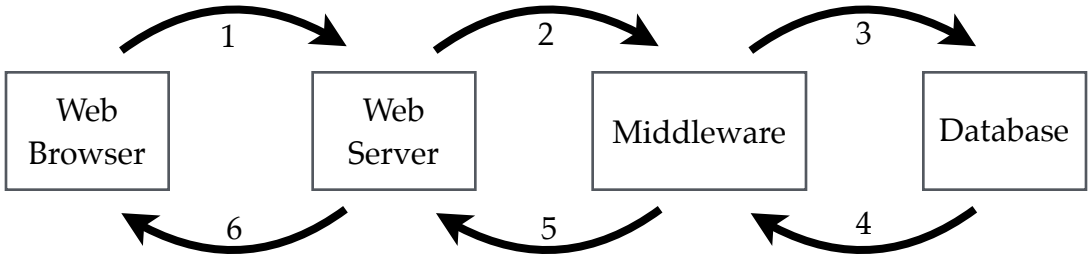
https://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

The above list is not comprehensive but instead reflects the platforms chosen by Wikipedia editors. It also doesn't give any indication of popularity. Less popular platforms don't offer a sufficient pool of experienced developers and are not recommended.

Up until very recently there was no qualitatively objective way to evaluate web platforms. This is because all web application design is based on the same architecture regardless of language.

A web application has four parts; the web browser, the web server, the middleware and the database. Developers can put code in all four places and they usually do. In the list of platform choices you'll see Javascript. That goes in the Web Browser. The rest of the choices are what we've labeled 'Middleware' below.

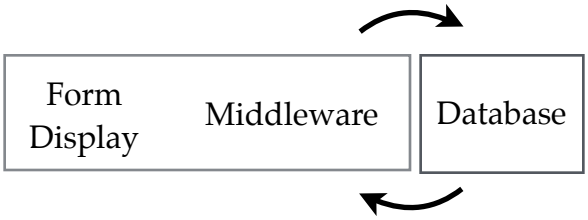
When a request is made by the user at the web browser, the web browser starts a chain of events that follows the arrows below.



For our purposes it would help if we name this chain of events "the request-response cycle".

Web Application Platforms(continued)

The web application request-response cycle is much more complex than the desktop request-response cycle. For comparison, here is an example of a desktop application request-response cycle.



When a request is made by the user, the form consults its own code. If it needs to contact the database it does so. Note that the form display and middleware are tightly integrated; to the point where its appropriate to understand them as one entity. On some platforms the database is also tightly integrated to the rest of the platform.

The difference is clear. The next question is "Why do we need so many pieces to make a web app?".

Web Browser

The reason we need Javascript is because its all we have in the web browser to enable rich experiences. Other technologies were produced to compete with Javascript. They used the browser plugin architecture. If the user wanted to view "rich" content they had to download software into their browser to enable it. Unfortunately, the developers of these rich experiences tied their technology to large screens, keyboards and mice. As a consequence, they fell out of favor once the smartphone was born.

Web Server

The reason we need the web server is because web applications have domain names. Web servers give you the ability to serve more than one domain name on the same server. Although they constitute a net increase in the number of pieces used to create our application they don't typically increase complexity. Web servers are worth the effort for the added capabilities they offer.

Middleware

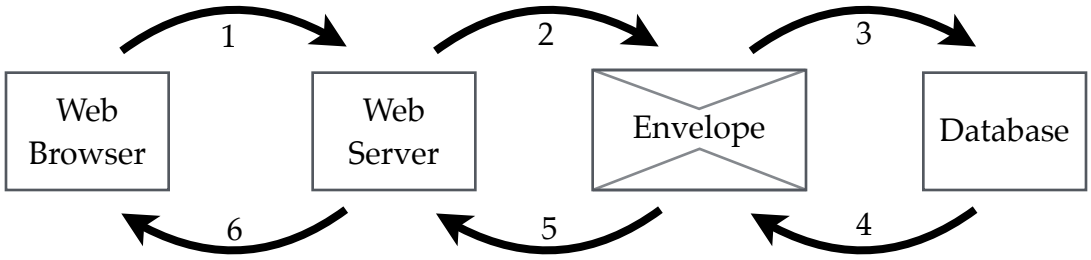
The reason we need the middleware is because web browsers can't talk directly to the database.

Database

The reason we need the database is because databases are the only safe way to store and get at data quickly. Data safety is important but if a web application is slow it will be difficult to use. Databases are the fastest way to get at or summarize large quantities of data.

Introducing "Envelope"
The first truly new web platform since the beginning of the web.

Envelope is a completely new platform that prevents the developer from writing code in the middleware by replacing it. This forces all code to be written for the web browser or the database.




```
<script>
// lots of code here
</script>
<body>
  <div id="combo"></div>
</body>
// On the server:
// lots of middleware code
```

Old Paradigm

Lets look at a simple example.

An organization wants a web page with a combo box, otherwise known as a drop down box. When the user clicks the drop down it should display a list of choices from the database.

To make this happen under the old paradigm, the developer has to write code in several places.

```
<script>
//Link to Greyspots Framework
</script>
<body>
  <gs-combo src="db.table"
     column="project_name">
     </gs-combo>
</body>
//On the server: Envelope
//(No code in Middleware)
```

Envelope Paradigm

In order to more closely mimic the desktop architecture, Envelope ships with a complete Javascript toolkit for building database applications. This Javascript toolkit, called "Greyspots", is based on a new technology called "Web Components".

Greyspots is very different from other frameworks in that it provides HTML tags that can be used to display rich interface objects on a web page and connect those objects to the database without the developer having to write Javascript or Middleware code.

Introducing "Envelope" (continued)

The Greyspots web components are highly integrated with Envelope and PostgreSQL. This reduces development time down to desktop application levels. Additionally, much work has been done to the components to incorporate desktop-like application richness and mobile compatibility.

Unlike many frameworks that came before it, web component technology is capable of parity with desktop applications. The goal of the Greyspots framework is to reach absolute parity but a great deal of work has been completed and the goal is nearly accomplished.

For the first time a small web application can meet a small desktop application budget. This is accomplished by reducing total lines of code. When migrating an existing website to Envelope, the code savings easily exceeds forty percent.

Due to the discrete nature of components in Web Component technology, anyone with a thorough understanding of Javascript can create new rich user experiences. And when testing new experiences, developers won't experience an exponential increase in effort as under previous Javascript frameworks.

Envelope has additional advantages that exceed the scope of this document. Visit the Envelope website at www.envelope.xyz for more information. Envelope is fully documented, free and open source. Paid support is available.