In Search of the Perfect Piece of Software

What Businesses Need to Know About the Software Development Process

A good piece of software…

Solves some sort of business problem for the end user. It makes life easier for the business, for the customer, or both. It does this by combining a high level of functionality with aesthetics. It operates as a seamless and reliable solution, for the end user.

Once you understand the steps and processes for creating the perfect piece of software, you'll be able to make more informed decisions that will benefit your company in the process. There may be more than one way to develop software, but it's important to strive for the highest levels of productivity, on projects large and small.

When you're talking about a project that can range anywhere from $60,000 to $1 million, it's important to look at your development company's efficiencies and productivity strategies. It's important to step back and look at the entire process.

When you understand what happens from start to finish, you are better able to collaborate and find the product that you are looking for on the other side.

Below are some of the steps involved in creating the perfect piece of software for your business.

The Right People

Software development projects are nuanced. What ends up being a "small" project to your development team is still a significant investment to the end user. Depending on the complexity of what a business is looking to have done, it could take weeks, months, or even years to complete. Many projects involve dozens of people all contributing code to a single project.

It's important to have the right chemistry amongst the team, to keep the project moving in the right direction. Personality problems, or continuous miscommunications can have a negative impact. That means taking a good look at who is involved.

"First and foremost, it's about developing the trust," Praxent Director of Operations Chris Walker said. "We've had people here who have been able to step into a client meeting on their first day."

Walker said that development teams reach that point because of the detailed questions the company asks potential employees during the hiring process. Candidates often walk out saying they have revealed stories they never intended to reveal.

This has the effect of both parties understanding each other. Once a person is brought onto the team, there is a mutual understanding that they can be trusted, he said.

"When somebody new here starts, you can be as candid with them as you would be with someone who has been there for 3, 4, or 10 years," he said.

That tighter team chemistry leads to better code, and a better end product.

Victor Vu is the lead engineer at Grow Labs, a software as service company dedicated to helping B2B companies find new leads. Having the right people involved in development is a high priority, he said.

"It's definitely the chemistry on the team," he said. "It's not being afraid to ask questions, and having check-in points."

The Process

Good Software Starts With Good Communication

The sales team has closed the deal, and a contract has been signed. Work begins on a new client project. It'd be easy for the development team to immediately assume and dictate what needs to happen next in order to create the kind of software that will solve the client's problem. But it's not that simple. Jumping in too early with the "expert mindset" can be a mistake.

When development begins before there is a full understanding of the problem, the work may not meet client expectations. Instead, a good development team will practice active listening with the client to ensure they have a full understanding of the needs for the project.

Some development styles put a higher emphasis than others on client check ins. But these check ins are important for ensuring that the development company and the client remain in sync throughout the lifetime of the project.

At times it may help to have a go between to communicate between the client and the development team. Having this point person helps streamline communication and makes sure the client and the development team are on

the same page. This can be especially important as the size of the project increases.

Praxent Business Technology Analyst Pete Van Horn often serves as this go between. He breaks down the development team's progress for the client. He also brings the client feedback to the development team, ensuring that both sides are in sync throughout the lifetime of the project.

"I communicate the technical requirements in a functional way, that our clients understand," he said.

When both parties are communicating effectively about the situation at hand, it's easier for the development team to manage expectations and deliver a product that the client will love on time, and under budget. It often helps when a client has someone dedicated to a similar role on their side, he said.

The first thing that happens is that the development team and the client must come to agreement regarding the size and the complexity of the project.

Point Sizing

Software comes in all shapes and sizes. Some programs are large, some are small. Some require massive databases, and an intricate level of development and programming.

Sometimes, all a client might want a different kind of button added to a form – or a similar type of minor request. Developers need a way to measure the size of a project, and establish a strategy for creating a deliverable. One strategy used to manage projects in the development industry is point sizing.

Once the size and scope of a development project is understood, it can be assigned a certain number of points. The Fibonacci sequence is commonly used to assign the proper number of points to a project. On this scale, the next number is identified by adding the two numbers before it. The scale starts with these numbers:

1, 2, 3, 5, 8, 13, 21….

It increases from there. If a client wants a button added to a form, that might rank as a 1 on the point-sizing scale. This means a single developer can handle the task in a matter of minutes. If a development project ranks

higher on the timeline, it may require a team, and a longer timeline to complete.

The use of the Fibonacci sequence offers the development team a little flexibility. It may be difficult to see the full scope of a project at the onset. If the temptation is to rank a certain project as a 13, but there are several unknowns, it may instead be a better fit at 21. When you round up, there is wiggle room as the details of the project emerge.

Praxent Solution Architect Ryan Ostrom said that point sizing gives the company a consistent methodology for determining the size and complexity of a client development project.

"People are often optimistic about what they think they can do. It's more difficult to plan how long something might take."

It also gives a methodology for communicating to the client exactly what the company will deliver, and when, he said. Instead of launching straight into the build, there is a methodology for giving the client a realistic glimpse at what the functional software will look like, upon completion, before beginning.

Software Prototyping

The best way to explain a software prototype might be with an analogy to another industry.

Let's say you want to build your dream home. It makes no sense for you to meet with the builder once, and have him start building without any follow up conversations, or continued planning.

The builder in this circumstance would likely be making huge assumptions about what you want, and it would become increasingly difficult for both of you to remain on the same page. Instead, you'd continue to feed him information about your tastes and preferences. He'll ask a lot of questions and launch into blue prints and other forms of planning.

This gives you a chance to determine whether you like his interpretation of your vision, before you make the larger commitment of hiring him to build your dream home. It's easier for both of you to remain in sync through the project with this kind of communication and planning.

Another way to picture a prototype would be like you are walking through a movie set. For example, seeing the movie set version of the death star ship

on *Star Wars* might look like the real thing. It might conjure up images of the movie. But the controls won't work. It's not real or functional, like it appeared on screen. The actors assuming the controls of the ship aren't going to blow up any planets.

The same ideas hold true with a software prototype. It's built to look and feel real. It's built to give you the experience of your new software, before all the work goes into the development. A software prototype helps establish trust on both sides, and defines expectations, before the hard work goes into the rest of the project.

Having a prototype before advancing to the later stages saves you time and money. It also provides a realistic determination of whether the client and the development company are an ideal fit for each other before the project advances too far.

Ostrom said that creating a software prototype is almost a must-have at the onset of a development project. He strongly urges clients to proceed with a prototype before moving on to software development. The Praxent prototype is called ClickModel, and it is designed to give the clients and developers that shared vision of the project.

"It helps us to make better decisions in the long-run," he said. "The good news is we've spent very little time building our structural prototype."

It's also good at unearthing software requirements that the client hadn't previously stated, he said. Once that shared vision is established through the prototype, the development work can truly begin.

Agile Vs. Waterfall Software Development Styles

You don't need to be a computer programmer yourself, or a technical wiz to hire and work with a development company. But it does help to have a basic primer on the types of development styles, and the development process, so that you can ask the right questions that allow you to receive the best possible product at the end.

Two common development styles are waterfall and agile. While both can lend themselves to a quality product, they involve extremely different approaches to the process.

Waterfall Software Development

For years, waterfall remained the industry standard for how a piece of software was developed. It originated in the manufacturing and construction industries, and is named after the downward flow of progress, through the phases of conception, initiation, analysis, design, construction, testing, deployment and maintenance.

The linear path of development within the waterfall model provides a structured approach, with easily identifiable phases. It serves as a logical method when the requirements and scope are fixed, and the technology is understood. This is not always the case with software development, however.

A client may be attracted to the waterfall method because of the regimented approach, and the ability to plan the whole project out. The problem with this is it can delay any sort of deliverable, and the software world changes fast. Your plan may not be relevant in a year or two.

The development environment often contains more variables than manufacturing or construction (where waterfall originated) – and this can create an unpredictability in the practice that makes waterfall a challenging solution.

As we talked about at the beginning, communication is key when it comes to starting any development project. The requirements for a given piece of software may change somewhere along the line of development. Developers may not be fully aware of the complexities of a project until it is underway. These kinds of complexities are challenging in a linear system that allows for little client-developer communication.

Waterfall can become an inefficient model for software development that leads to the client not receiving any sort of software deliverable until the final stages of development. For large projects, this can mean that the client goes without a solution to their business problem for months, or sometimes even years. In the waterfall system, there is often very little contact between the client and the developer during this time. From a customer relations standpoint, this can present some challenging problems.

For the above reasons, other software development methodologies are often used to create more efficiencies in the process and deliver more value to the end client.

Large development firms such as Praxent would be ideally suited to a development system such as waterfall, Ostrom said. But there is a good

reason that they don't use it. There are other, more viable alternatives for the types of projects they work on.

Agile Software Development

Agile software development is more suited to the changing needs of the client. It's a way to maintain a workable system, and give the client a product that they can realize some benefit from, in a much shorter time span.

Agile is a more adaptive process that allows for the changing needs of clients through increased collaboration between client and developer. Software needs are given to change, and agile has a built-in methodology for being able to address those situations.

Agile development still requires planning, but that planning occurs in cycles, so that the client and the developer aren't locked into a long-term direction that may not be the most viable in the long term.

For Ostrom and Praxent, he said it's a matter of showing a client how an agile process can help give them the product they want. It allows them to create a minimum viable product and build from there. This means that a company can go live with a smaller version of the product, and then expand as more features are complete.

The frequent check-ins also allow for a better way to handle a client's evolving business model.

"This allows us to build a piece of meaningful, valuable software without wasting time or budget," Ostrom said.

Below are some of the common phrases associated with agile software development.

Sprints

In agile, work is done in "sprints" where the project is broken down into more manageable chunks. A software sprint typically lasts about two weeks. The developer is in regular communication with the client between sprints, so there is an up-to-date understanding of the project status.

Typically, both the developer and the client have dedicated point people. The development company shares the progress made on the project, and the

client communicates priorities, and any possible shifts in the direction of the project.

Stories

In the context of development, a story is a high-level way to describe a software requirement. The story contains enough detail for the developer to understand what needs to be implemented on the project.

Epics

Like in literature, an epic is simply an extended story that can often be broken down into several stories. An epic may even contain multiple projects. Depending on the point size, an epic may take several sprints to complete.

More on the Agile Development Process

The best way to break agile down may come from an original source. In 2001, 17 developers met in Utah. They drafted and signed the *Manifesto for Agile Software Development.*

In that document, they professed to value:

- ***Individuals and Interactions*** *over processes and tools*
- ***Working Software*** *over comprehensive documentation*
- ***Customer Collaboration*** *over contract negotiation*
- ***Responding to Change*** *over following a plan*

These values highlight the adaptive qualities of agile software development. Where waterfall is structured to provide the whole product at once, agile fosters a strategy where key components can be delivered over a fixed time.

This process allows the space for flexibility over the demand of a rigid schedule. It allows for an increased level of collaboration between the client and developer, giving the client more direct input over the project. It allows for a more natural communication cycle through the development process.

The differences between agile and waterfall are evident in the 12 agile software development principles laid out in the manifesto:

1. *Customer satisfaction by early and continuous delivery of valuable software*
2. *Welcome changing requirements, even in late development*

3. *Working software is delivered frequently (weeks rather than months)*
4. *Close, daily cooperation between business people and developers*
5. *Projects are built around motivated individuals, who should be trusted*
6. *Face-to-face conversation is the best form of communication (co-location)*
7. *Working software is the primary measure of progress*
8. *Sustainable development, able to maintain a constant pace*
9. *Continuous attention to technical excellence and good design*
10. *Simplicity – the art of maximizing the amount of work not done – is essential*
11. *Best architectures, requirements, and designs emerge from self-organizing teams*
12. *Regularly, the team reflects on how to become more effective, and adjusts accordingly*

There are some key differences between the development of software, and the industries of manufacturing and construction – where waterfall originated. Agile accounts for these differences by creating a flexible approach to client demands.

Within agile software development, there are different frameworks for output. Two frequently used frameworks are scrum and Kanban. Below is some additional information about both.

Scrum

Scrum is an agile framework designed to handle the flexibility of software development. It is structured to allow for the idea that the client's needs may change or evolve over the course of a given project. This can sometimes be referred to as requirement volatility.

It is also structured to allow for the fact that there will be unpredictable, or unforeseen challenges through the course of a software development project. Scrum is focused on the team's ability to deliver quickly, maintain a position of adaptability, and remain flexible to account for the forces of an evolving market.

Some characteristics of the scrum approach include developers working in the same physical space, or in close collaboration online. This allows them to focus on a common goal. It allows for daily interactions and meetings to determine the progress toward a goal.

Kanban

The kanban framework allows for improved efficiencies across systems. It aims to manage work by evaluating capacity and improving system workflows.

The process is made visual as participants view the process and progress through a visual board. This can aid the decision-making process and influence the direction of a project. The underlying method began with Toyota's manufacturing process, but is heavily utilized for development and technology work.

It's often combined with other work methods, such as scrum. While scrum is often used in feature development, kanban can be used for aspects of project maintenance.

Creating Efficiencies Within the Development Process

Agile can be used to build efficiencies into the development process. It's important to note that 7 out of 10 development projects end in failure. That's an industry standard, and one that we think we can beat.

Taking the time to make some simple decisions at the onset can improve your chances for success. For instance, it's important to consider questions like whether a software solution is the best approach to solving the problem in the first place. It's also a time to understand what the process looks like, and how to eliminate waste.

It's important that the development company you choose is not only capable of handling the technical nature of your project, but that they are capable of good communication, project estimation, and regular client check-ins.

That kind of flexible development style can make all the difference when it comes to the success or failure of your software development project.

For Vu and Grow Labs, he said that an agile approach allows the development team the ability to meet the needs of the sales team or adopt to customer feedback. The built-in check-in points help, he said.

"If there's a miscommunication, we're basically just adding cost."

Code Environments

It's a good idea to have a basic understanding of what is involved in the actual implementation of a software development project. Any given project

may have several developers working on it at one time. To accommodate this type of situation, multiple environments are set up to handle the code.

One reason that multiple environments are established is to keep developers from overriding the work of their team members. It's also important that the code works in conjunction with the other components of the final project. Developers should test all code, and make sure the software is of the highest quality before it is handed over to the client.

Below are a few common environments, but not every development company may use them the same way.

Some common production environments include:

Local – This is the individual developer's workstation. The code developed here is yet to be integrated into the larger project. At this point, the individual developer is working to create code that performs a specific functionality in isolation.

Sandbox – This is an environment to determine where/how untested code performs. A sandbox is an area to test the code's quality, while still protecting final versions.

Integration – This is the process of merging developers' code into a working copy of the software, to test how different components work together.

QA (Quality Assurance) – The purpose of quality assurance is to assure that the new code won't have any negative impact on how the current version of the product works. The QA environment is used to test the functionality of the system before the code is moved on to the next environments.

Staging – The staging environment is used to assure that all upgrades to the production environment will be completed without errors. It is an added step to assure the quality of the final product.

Production – This is known as the "live" version that users will directly interact with. The previous testing environments are used to assure the quality of the production environment. The production environment may be released in stages, to assure quality.

Again, these development environments may vary by company. But it does help to understand the various steps your software undergoes before you see a finished product.

Van Horn said he is often testing in the quality assurance environment. When he tests code in this environment, he is essentially looking to recreate the user experience, he said.

Having an established testing system helps ensure that the client receives a quality product every time, he said.

"It gives me great comfort knowing exactly where things are."

It's also important to find a developer that practices automatic deployment into the production stage. The alternative is manual deployment, which involves the unnecessary risk of data loss or corruption.

More on the Development Process

The team of developers assigned to a given project estimates how long the various aspects of the project will take, based on points and the complexity of the build. No one knows better than the developers themselves what may be involved in the creation of the code, Ostrom said.

The focus is then on creating the minimum viable product, instead of building the whole project out. This allows for quicker turnaround times, and it allows the development company to complete something they can deliver to the client at a much faster rate.

"Agile involves the same amount of time and the same amount of product," Ostrom said. "It's how you prioritize the work. Waterfall won't give you the important features as you need them."

The business technology analyst and the project manager check in with the developers as they work to complete the software. They will discuss in meetings what will be involved in upcoming sprints, and how to make the work more efficient.

Praxent maintains a 97 percent delivery rate when it comes to completing the points on the schedule they have committed to, Ostrom said.

They are regularly reviewing the production timeline and the budget for a given project, Ostrom said. They are also giving the clients a CommandView Report, where the project metrics are laid out in a way that keeps the client

well informed of progress. The report reviews the amount of software delivered, against the timeline and budget.

Version Control

Software routinely needs to be updated to reflect new features, changes in the database, and other factors. These updates occur within a process called versioning, or version control.

It's similar to the process that your cell phone software goes through. Changes and updates are made, and then the user downloads the new version.

In this way software development can be considered an ongoing process. The software is maintained to ensure that it remains compatible with new technology and operating systems.

Putting it All Together

The agile process is designed to create an efficient methodology for delivering a quality product to clients on a consistent and regular basis.

Laura Alattore Kvalheim is the Director of Project Management at Praxent. She oversees a variety of project large and small. The company often has around 25 active projects taking place at any given time, she said.

It is important for her to keep some of the above strategies and methods in focus to improve the client's experience, and to ensure that the development team completes the project both on time and under budget.

The company keeps a project plan for each client, to create a higher level of organization and efficiency, she said.

"It's a way for us to organize the work. That way we don't always have to worry about what's next," she said.

The key to success is in making sure the project is well laid out, she said.

"A really great project manager can modify and mitigate risk before it becomes a roadblock," she said.

Kvalheim said it is important for scenario planning, and envisioning what is possible under any scenario, including the worst case. That way developers,

and the company can remain ready for any possible situation, making the operation more flexible in general.

Kvalheim said it helps when the right resources are in place on the developer side, and for the client. Below are a few key questions she asks.

- Does the client have the executive sponsorship to keep the project going? This serves as a strong indication of the client's level of commitment.
- Is there a subject matter expert for the client? It helps when the client has a person, or team, invested in the success of a project.

There are also a few factors the company evaluates before beginning:

- Does the client have a current product in place? Are there technical artifacts already established, and a defined process?
- Is this a new idea? Is there already a product at market level?
- Can the project be established with a minimal viable product? The project may be something lightweight that can change over time.
- The type of project. Is it a migration, or will you be working with a new technology? Migrations can take a long time because a new development company is interpreting and working with somebody else's code, which can be complex.

Kvalheim said her goal is to make sure that communications between Praxent and the client are efficient, and the client feels in control of the scope and budget.

Agile is an excellent framework for delivering the product faster. In a waterfall system, tight deadlines can be extremely difficult. Some waterfall projects can take years before the client sees anything at all.

"With agile you can deliver something in rapid succession, so the client sees value a lot quicker."

Choosing The Right Developer

Any software project, no matter the size and scope, will be a significant investment for your business. Hiring the right development company can make or break the success of your project.

Businesses should be looking for a developer with a proven track record. The developer should be able to display that they have worked on similar projects before, and that they can meet the necessary demands.

Before you hire the developer who builds your ideal piece of software, you can ask them about their approach. Find out how they convey that shared vision for your product. Ask about the staging environments your project would pass through before a final version is delivered. Inquire about the company culture. These are all factors that will play into the success of your project.

Try to find a developer who will use real user testing to validate data so that no time is wasted operating on invalid assumptions.

Culture is important in choosing the right development company, but so are things like prototyping, QA testing, and practicing automatic deployment of code. It's important to find a developer who can convey trust, and show you that they understand your vision and have the capabilities to carry it out. It's important they have the necessary resources to dedicate to your project.

The process works best when the client also has dedicated resources, and a vested interest in seeing the success of the project. This often means a knowledgeable point person, who can quickly convey the company's needs.

A development company will have a person that the client can reach out to, so they can always stay in contact on a given project.

The development company you choose doesn't have to necessarily prescribe to an agile philosophy, but you should understand the differences between the two approaches. Software projects can often take years to complete. That is a long time to go without a deliverable, and little contact with a company you've hired to work on such a large project.

When you are satisfied with the developer's technical skill set, you can find out about their communication processes. Find out how they communicate progress, timeline and budget.

A piece of software is a significant investment, no matter how large or small. It is critical to understand enough about the process to ensure you receive the best product possible. It's also critical to establish quality communication, and a high level of trust.