ASCENDLE

Software Innovation ROI Calculator

An Executive Guide for Business Leaders

Software Innovation ROI Calculator:
An Executive Guide for Business Leaders

Table of Contents

Introduction

The time has long since passed when business leaders in mature market verticals could look at the innovations coming out of Silicon Valley and think "that high tech stuff doesn't apply to my world". Today, offering engaging and polished customer-facing software technology is no longer the domain of a few technology companies. Every company, no matter their history or traditional industry vertical, needs to excel at being a tech company.

Few business leaders from outside traditional tech—and even many within traditional tech—have the experience to evaluate software development projects and the best strategy for undertaking them. This executive guide was written to help those business leaders understand the expected value of the software innovations they know they need, how to address capital expenditures of costly software development, and the factors for managing the work in-house, via outsourcing, or through a software development partner. Ultimately, we know that the insights offered can help the business leader dramatically improve the balance sheet.

Let's dive in!

Expected Value Defined

How does your company define the success of a software development initiative? The definition is pliable and unique to you and your customers. You know the variables that matter most to your company, and you can drive your software innovation strategy by analyzing the outcomes of your current strategy against your desired state.

Companies use a variety of factors to determine their software development strategy, putting different weights on factors like the importance of capital expenditures, time-to-market, and customer satisfaction.

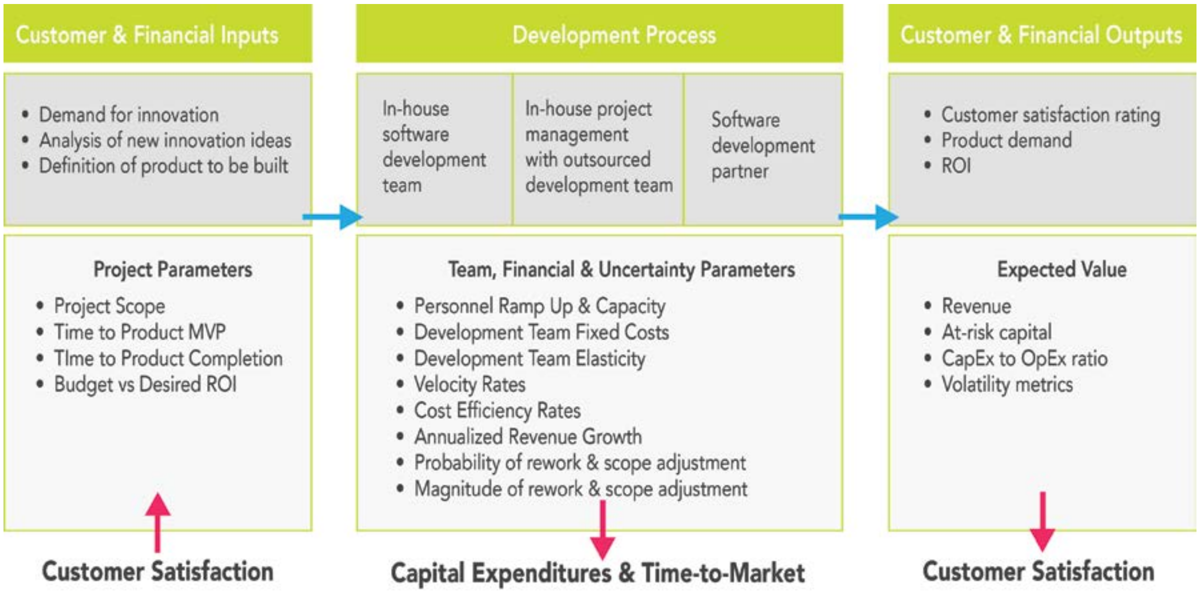

Figure 1. Inputs and outputs of software development. See calculator for more detail.

In the figure above we are assuming that customer satisfaction, capitalization of expenditures, and the time it takes to get to market are the values we weigh most heavily in determining the success of an innovation project. They are also the indicators for continued support of innovation projects. We need to take a look at what is influencing these three factors to help us better understand how we can achieve the expected value of a software innovation project. The **expected value** is a predictable value of a variable. It is the sum of all possible values, each multiplied by the probability of its occurrence.

Customer Satisfaction & Agility

Customer satisfaction plays the largest role in the success of product innovation, driving the success or failure over the life of your software products. In addition to the obvious attention you give to trending sales, margins, and customer retention levels, it is important to also track your company's ability to be nimble while innovating.

Customers are expecting rapid innovation and don't know or care about what software development process you use to get there. What they expect is that you listened to their requests, you understood them well, and you deliver or overdeliver on their expectations. Consider that constant, rapid improvement is the formula for success, and delaying release to strive for perfection is costly.

It is important to track your company's ability to be nimble while innovating.

If customers are requesting a specific innovation in Q1 and you cannot deliver until Q3 of the following year, or you deliver features that are misaligned with their actual needs, your balance sheet will force you to take a hard look at the process you are using to respond to customer needs. And you wouldn't be alone. CHAOS analyzed over 25,000 software development projects and found 60% of all projects were not completed on time, 19% were cancelled before completion, and 52% of projects cost 189% of their original estimate.

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

How do project completion rates and cost overruns impact your definition of project success or failure? You probably allow tolerances to your project costs and timelines. In some extreme cases even, a cancelled project may be deemed a success; perhaps you prioritized launching early to test for market validation, and then the market quickly rejected your innovation. You deem it a success because you spent $250,000 rather than $1.5 million narrowing in on your customer needs. In other cases, only a 25% cost overrun or 10% time overrun are palatable.

Project Tolerances and Archetypes

At the moment you are either sitting in front of financials that have you asking yourself, "How can I get to ROI faster?", or "How do I replicate this success in other departments?" Our best advice is to study the software innovation successes or near-successes your company has had, whether in your department or in another division. Copy them. But copy them with a caveat.

The most agile or nimble thing you can do to increase the efficiency and success of software innovation at your company is to quickly evolve to the software innovation process that tightens your focus on increasing customer satisfaction.

There are different project archetypes: greenfield development; enhancement of existing or legacy products; refactoring of a legacy product. Make sure to compare apples-to-apples when forecasting your next project. Successfully using internal teams to enhance your keystone legacy product has different considerations than using that same team to rearchitect for the cloud or launch greenfield development.

If your company works best spinning up internal teams to add a new product line or strengthen your legacy product, develop a strategy to do that as quickly as possible. If a neighboring division showed success bringing in a software development partner, find out who they worked with. In either case, find out what the structure for success looked like: who were the stakeholders providing the project vision; what were the critical project testing and review points; how quickly was the product in the hands of end users; what advantages and disadvantages were there to the software development team; why did they consider it a success, and what would they change?

On the other hand, if you do not have model projects to reference, examine the failed processes. How did you deploy your team(s)? How flexible were you in deploying your team(s)? What was your production time/dollar? What was your lead time and burn time? Think of ways you can make improvements or try a new process for getting the innovation launched faster, and with a better customer service score, than before.

Here are four tips to consider: [Using Business Goals to Keep Development on Track]

Capital Expenditures & Resource Efficiency

Annual enterprise software spend is over $400 billion (Gartner), nearly $250 billion of that is spent on proprietary software (Harvard Business Review), and 66% of software projects have cost overruns (McKinsey). These are large numbers but we doubt you are surprised since the statistics are against you having completed the majority of your projects on-time, on-budget, and on-point.

Companies want capital expenditures because they can defer the P&L impact over the life of an asset. And yet, innovative projects in software and R&D cannot be capitalized until they have a defined commercial value. In the case of software that equates to a validated build. That means you need to fast track your innovation to the point where you can amortize the product, so it does not hit the P&L all at once.

We often see this as a deciding factor when a company is selecting what process they will use to realize the innovation: in-house teams, outsourced teams, or a software development partner. If they use internal team(s) they have to weigh the added up-front fixed costs of assembling the team members while waiting on the lead time to begin producing the future asset. These alone could prevent an internal project from getting approval. This may be a non-issue if they have an existing team with the capacity and desired technical expertise, who also operate in short development cycles that stress the importance of achieving validated builds quickly.

Gartner: https://www.gartner.com/en/newsroom/press-releases/2019-10-23-gartner-says-global-it-spending-to-grow-3point7-percent-in-2020
Harvard Business Review: https://hbr.org/2018/11/how-software-is-helping-big-companies-dominate
McKinsey: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/

As an alternative they could look to one or more internal project managers to oversee an outsourced technical team with the appropriate technical expertise. This is often seen as a less expensive solution since hourly rates are typically multiples less than internal resources and software development partners. However, there is often little attention to **test-driven development** with this solution, leaving comprehensive testing to the end of a development cycle and extending development by months. Outsourcing the technical team also typically does not lead to the development of shippable product increments every 2-4 weeks, putting into question the ability to deliver a validated build, and reaching amortization requirements, quickly.

Read a piece we wrote on the business value of **test-driven development**.

A software development partner is worth considering if your internal teams do not have three things: a proven, sustainable process for quickly developing shippable increments; the technical expertise to innovate without requiring new tooling and training; and the capacity to commit to new development within the desired timeframe. Software development partners tend to come with a fully-gelled team whose makeup greatly reduces the oversight required of company stakeholders, lending more time for stakeholders to focus on understanding and managing their business goals.

When engaging with a software development partner make sure they guarantee a cadence for producing shippable increments.

When engaging with a software development partner, make sure they have a proven process for delivering business value quickly, including a commitment to test-driven development and a guaranteed cadence of producing shippable increments. The last advantage to engaging with a partner is resource efficiency, having the ability to spin up and down teams quickly if business values shift dramatically, as we know they statistically do. This option makes you more elastic with personnel, reducing the fixed overhead you have with internal teams. Team capacity can be quickly increased or decreased to meet the demand, absorbing the shock of a project that gets cancelled, or supporting the ramp up of a project that gets fast-tracked. Some partners will even guarantee their work, so if no business value is achieved during a product iteration or sprint, the client doesn't pay.

SECTION 4

Time-to-Market & Speed

Innovation does not necessarily mean groundbreaking. Innovation for your company may mean moving your services to the cloud, reviving a failed product using a new software development process, or developing a new service with an architecture your company has not yet used. Regardless of your form of innovation, time to market is a key factor when calculating the expected value of the innovation.

Regardless of your form of innovation, time to market is a key factor when calculating the expected value of the innovation.

There will always be factors of time-to-market uncertainty with any project. The three main factors we hear about are redefining the project scope, reworking completed code, and operational delays. All have the potential to accelerate, prolong, stall, or entirely upend a project.

Redefining Project Scope

Redefining project scope happens all the time, and it should! A healthy product backlog will change regularly, taking into consideration new customer feedback, updated industry trends and internal budgeting discussions. The backlog should adjust with the strategies for achieving business value and should support accelerating your time to market.

Healthy backlog grooming also means the stakeholders are not getting in the way of your software development team. Once the team begins developing a prioritized set of features, they should not be interrupted, with the understanding that those features will be demonstrated to the stakeholders within 2-4 weeks' time. In this scenario a marketable product often matures faster than expected because the stakeholder and team are in a partnership to develop the most valuable product as soon as possible.

Costly uncertainty inserts itself when the scope of a feature that is actively being worked on by a team is adjusted.

Costly uncertainty inserts itself when the scope of a feature that is actively being worked on by a team is adjusted. In this scenario the team has a difficult time getting footing, having to throw out work, and reset to the newly reprioritized workload. When the stakeholder does not respect the importance of prioritized features, the team cannot efficiently work towards a marketable product.

There are other realities of redefined scope that have little to do with stakeholder and team makeup. Market demands can shift quickly and development agendas materially change. Redefinition probability and the magnitude of said event may not change whether you use your internal team(s) or external resources. However, consider compounding challenges with operational expenses for internal teams—severance for a disbanded team or re-tooling for an architectural change—a volatility that could be greatly reduced with external resources.

Read our piece about **value stream mapping** to help frame the discussion around rework. If your chosen software development solution is working well, there should be little to no rework.

Rework

Rework largely depends on your software development process. Minor rework happens in all projects but if it becomes frequent, put a magnifying glass up to it. It is healthy to keep tabs on the frequency and intensity of the rework your software development teams are experiencing. Is the rework related to poor feature prioritization, a weak team member, or an incomplete testing process? And where does the need for rework usually get defined – during a stakeholder demo, deployment to production, or customer feedback? Often being aware of the frequency, magnitude and timing of rework requests can put you on the path for improving rework demands.

Delay

People are the lynchpin of all projects. They are the only thing that leads us to the successful completion of a project and they are what lead to the greatest delays in launching a product. Delays associated with miscommunication of feature sets is real, but difficult to calculate. The costs associated with on-boarding, training and tooling team members are more tangible. Your company likely has standard costs for these, and they should be taken into account when setting expectations for your time-to-market goal.

Combat these three factors of uncertainty by analyzing past projects so you can start committing to shorter development timelines. Ask the hard questions: Do you historically adjust scope mid-production, and on a frequent basis? If so, by how much? Are you often unhappy with the end product the team is developing, requiring regular rework? Do you have high employee turnover? Does on-boarding take longer than you expect, every time? Is there one external resource that consistently stalls a project?

Analyze past projects so you can start committing to shorter development timelines.

Expected Value Differs for Each Stakeholder

A wide array of stakeholders care about the expected value of software innovation, and no two stakeholders will calculate expected value the same way. A CEO will likely be focused on variables that drive to a competitive advantage. A CFO may concern themselves with minimizing fixed cost risks while maximizing expected financial returns. A product manager may think that solving a specific problem is the best use of their R&D budget, while a technologist may be preoccupied with how many breakthrough inventions result from a given level of investment.

No two stakeholders will calculate expected value the same way.

As we mentioned at the start, the ratio of risk-adjusted input to output determines the expected value. The inputs for a customer may be speed, ease of use, convenience. The inputs for the CFO of the software product may be time to assemble team, on-boarding costs, and time to market.

The outputs are value. For the customer that is reliability, relatability and relevancy. For the CFO output is a function of minimized fixed cost risks and maximized financial returns. While the CEO cares about all outputs, their eye may be focused on revenue, margin, profit, market share and growth.

10 Most Critical Software Innovation Practices

Companies evolve to become successful software innovation practitioners. We see it regularly. These are some of the most critical factors we observe as companies champion themselves into the world of fast software innovation.

1 Team Trust

Business stakeholders must trust the team to build the right product and the development team must know the features are prioritized. This relationship, based on trust and communication, takes time and commitment.

2 Team Communication

Business stakeholders and the team are speaking the same language. The feature descriptions or user stories are simple and clear. Everyone with a stake in the project can explain a feature.

3 Focus on Business Value

Focus on Business Value – The product backlog should always be customer focused. What does the customer want? And what do they most want first?

4 Produce Shippable Increments Quickly

Drive the product to a potentially shippable state every 2-4 weeks. Everything should have been designed, coded, tested, and all bugs fixed. It is ready to put into customers' hands.

5 Have a Strong Product Owner

Have a Strong Product Owner - The product owner is the absolute owner of the product's vision and requires critical thinking to be able to map out the route to delivering value.

6 Product Visibility

The development team is demonstrating the shippable increments to the business stakeholders every 2-4 weeks. It is most effective if the product is deployed so stakeholders can use it themselves, or send to others for feedback.

7 Use an Estimating Process

You need to be able to drive a schedule. If you want to ship something in six months, the team shouldn't have 12-months' worth of work to do. Use a process to figure out how fast the team is moving and extrapolate to predict the schedule.

8 Create a Shared Definition of Done

Create a bullet list of requirements that need to be met before a user story or bug can be considered done or shippable. Then share it with all members of the development team and all stakeholders.

9 Use the Right Tools

Rely on the right set of architectural and testing tools to make sure you are building a quality product. Match that with a committment to test-driven development.

10 Follow Agile Principles

This is a catchall. But generally, we find when people truly stay nimble they can better address market demands more quickly.

Calculating Your Expected Value

You can use the back of scrap paper or an elaborate **spreadsheet** to calculate the expected value for your next software innovation project. Keep it all to yourself or use it as a tool to drive discussion among your leadership team. Regardless of the role this exercise plays, it will only be valuable if you are realistic in your assessment of past work and the potential for continuous improvement moving forward.

Consider the various inputs your company values, the inputs you have control over and the ones you do not. Sketch the outputs you want to strategize towards and then reflect on the processes and systems that are in place, or need to be put in place, to meet your expected value.

All practices mentioned in this guide can be supported by either internal teams or external partners. Use the instructions in the next section to calculate the expected value of your next software innovation project.

Here are key steps to calculate the expected value of your next innovation project.

1 Know Your Customers

Your customers and their demands should be driving your business goals. Ensure you have a clear picture of all your customer sources. They may be longtime fans or strangers, waiting in the wings to try your next product.

2 Understand Your Customers

Which customers are you trying to keep and who are you trying to attract as new customers? Why? Are you sure you understand what they are asking for? After all, they may not really know what they want! Do you have a tried and true process for capturing and analyzing their feedback and requests?

3 Know Your Inputs

Inventory all the resources needed to execute your innovation projects. Drill down on the personnel and technical requirements, as well as the distinct components that come with the specific type of project you are planning: greenfield, legacy, refactor.

4 Know Your Tolerances

Does your company or department have standard tolerances for personnel overages, time-to-market overruns, and completion rates?

5 Relate Inputs to Business Value

Know how customer feedback flows into the formation of your company's business strategy and your business goals and objectives. Does your division have a masterly way of capturing customer feedback and market trends to help guide your strategic decisions? And how does this flow through to a prioritized product backlog?

6 Know your Team

Get a clear picture of what your team currently offers, where they excel, what they could improve on, their skillsets, and their deficiencies. Be realistic with their capacity.

7

Know your Process

Regularly analyze the work your software development process produces. Flag the improvements that could be made and the successes that should be repeated. Include yourself and other stakeholders in this analysis. Who usually provides the product vision to the team? How is the appropriate architecture and technology typically chosen? How quickly is the most innovative aspect of the new software application demonstrated in a shippable state? What role does Dev Ops play in your process and how can you engage them as early as possible?

8

Know your Process Options

Have a pulse on the software development options available to you. Your internal team may be the only solution you want to use because of their continued success. But what are your options to quickly produce a high-demand product if your current team is at capacity? Know the options for spinning up another internal team or looking externally for solutions.

9

Make Friends with your CFO

Getting project approval for innovating software projects requires messaging a succinct value proposition. Make sure you understand your CFO's concerns and interests around innovation projects.

10

Know your Time to Market Constraints

Be realistic with the constraints you have in getting a product to market. If you need a demonstrable product by a specific date to sell at a tradeshow, know the features that are absolutely required to get you there. And let go of the nice-to-have features for now.

11

Relate Inputs to Business Value

Nail down they key outcomes that are important for you and your team/department to succeed. ROI, time to market for MVP, time to market for version 2, ability to predict completion time, and rework, scope change and delay tolerances.

12

Have a Plan for Getting Customer Feedback, Fast

Getting feedback on the most innovative aspect of your product first is critical. Do not create a plan that will leave the innovative development until the end. Your team and all stakeholders should be testing the newest innovation within weeks of production in case trouble arises and you need to adjust the innovation. As soon as your customers get their hands on the innovation, make sure you are there to capture their activity and feedback so you can continue to prioritize the work with the greatest business value.

Dive In to the Software Innovation ROI Calculator

We took a stab at defining some standard inputs and outputs when looking at expected value of software innovation. Experiment with our **software innovation calculator** to find the right mix of internal, outsourced and software development partners to compliment and diversify your software innovation projects.

The calculator shows you expected value in terms of dollars, but also in terms of time and scope. It also provides some variance metrics to help you understand how volatile your situation might be. Your software innovation strategy is built from understanding the weight each of these values plays in your overall innovation portfolio.

Download our Cost of Software Innovation ROI Calculator to evaluate the risk and efficiency of your software development strategy.

Use Calculator

As you review the outputs from the tool, keep in mind you can analyze more than just cost per hour comparisons of your various development team options. Afterall, cost per hour comparisons may not lead to the right decision if time-to-market or product quality are your greatest drivers.

It all goes back to knowing your customer. Can they best absorb a delay, a cost increase, or a feature reduction? Every customer base and possibly each project will have different tolerances. Listen to the needs, reference your own project experience, and don't be afraid to try something new.

How Ascendle Can Help

Let us help you get the most out of the innovation ROI calculator, or brainstorm your next innovation project with you!

Ascendle

PORTSMOUTH OFFICE:
One New Hampshire Avenue
Suite 125
Portsmouth, NH 03801

BOSTON OFFICE:
399 Boylston Street
Floor 6
Boston, MA 02116

833-392-4453
engage@ascendle.com

www.ascendle.com

Our evidence-based software development services help firms realize their innovation goals on-time and on-budget. See some of our **case studies**.

Contact us for a **free consultation** or to walk through the **innovation calculator** together.

Ascendle guarantees our software development partnership. We build software applications that don't fail. **Guaranteed**

We are your software development partner, building cloud enabled desktop or mobile applications, in situations where failure is simply not an option. We buck the trend of software projects that go over-time, over-budget, and don't deliver on business value. In fact, Ascendle has a 100% guarantee that will not happen on any software development project we take on. We are the software development partner you hire when you've gotten fed up with every other option. Your success is our success.

Our Services

Software Strategy

For a 3-6 week period we devote a team of software professionals to your product vision, client issue or innovation fatigue. They listen intently to your ideas, ask stimulating questions to get at your true pain points, and drive at the most pressing business priorities.

Your Ascendle team works closely with you to think deeply about your product solution, calibrating the deliverables until they are confident you will find the software strategy deliverable valuable.

At the end of the strategy engagement your team will hand you an executable document explaining your product's:

- Key requirements for each of your product features
- Technical strategy for developing the produc
- Build schedule
- Cost estimate

We deliver business value or the work is **free of charge**.

Software Development

We build shippable product increments every two weeks, using our proven process and the technologies best suited for our clients.

When you leverage an Ascendle Accelerated Delivery Team you'll get:

- Face-to-face communication with your team on a consistent basis
- Access to source code, which you own
- Fully-tested code with comprehensive coverage by automated unit tests and continuous integration
- Reliable delivery on a predictable schedule
- Utilization of the latest cloud and app development technologies

During the software development phase, the application is built in a series of two-week iterations. A private demo environment enables you to conduct a hands-on evaluation of the product at the end of each iteration and provide feedback for revisions and further development. Each piece of functionality is coded and tested to deliver a final, usable product.