Software cost estimation: what factors are key in IT project estimates?

Whitepaper

Siberis

Table of Contents

Software cost estimation

What factors are key in IT project estimates?

Say you're ready to start developing your application. You have an idea and the budget to fund it – so what's next?

The most pressing issue is finding a knowledgeable, reasonably-priced IT outsourcing provider to help you with development. The best way *to determine if the price is reasonable* is to understand the provider's internal process: what steps do they take to understand your idea? What line-items are included in their project estimate? Are there any project elements whose cost is noted as "TBD"?

The estimation process for web, desktop, and mobile app development is fairly similar, so we'll summarize the basics and then focus on some peculiarities for mobile application development.

Customer-provider dialogue and its impact on estimation: from the provider's point of view

When you're looking for talent to bring your idea to life and none of your usual partners can help, you typically have two options:

1) Describe your idea in writing, post it on freelancer platforms, and wait for proposals

2) Search for an IT outsourcing provider and contact them.

In both cases, an IT Consultant is usually the first person to contact you. In the case of option #2, we'll just assume for simplicity's sake that you contacted Sibers (an excellent choice, by the way). An IT Consultant is a person savvy in technical matters and communication. His or her role is to understand your idea and the implications of implementation.

Safety first

Some customers get uneasy when asked to describe their idea. They're afraid it might be stolen and that someone else could benefit from it, especially if it's posted in a public space. But rest easy — we're 100% certain that your idea will be safe, and here are several reasons why:

1) Like most providers, Sibers has worked with dozens of start-ups. Some have

achieved success, others haven't. The truth is, no one knows how a start-up will perform. Moreover, *it's not always clear why one app finds success and another doesn't*. Popularity with users isn't so much a product of technical excellence as it is a conflux of many factors: the client's personality/marketing savvy, the passion they have for the app, the timing of the app's launch, etc.

2) *A non-disclosure agreement (NDA)* is commonplace in our business and any industry where intellectual property (IP) is involved, especially with projects that require specific scientific know-how or utilize unique components. An NDA prevents a provider from talking about clients and their projects For example, if you post your idea on a freelance platform, the platform's members must sign an NDA before joining.

3) *Professional reputation* is everything to a provider. Getting caught stealing an idea would sabotage years and years of hard work, and/or put them out of business completely.

4) Last but not least, consider the old adage "There's nothing new under the sun". *Rare is the idea that is completely unique* and unheard of. That said, without proper execution even the most innovative idea is only worth the paper it's printed on.

Describing your idea

The reason we advise our clients to provide a comprehensive description of their idea is because it will ultimately save time for all parties involved. Two big factors here:

First of all, if you provide a detailed description you can expect to receive equally *detailed proposals* or thoughtful, relevant questions from providers with the skills and experience to tackle your challenge. And by extension, you can filter out the providers who don't provide a detailed response, because they would likely just waste your time.

Secondly, a detailed description creates a foundation for an informative, useful first discussion. If we read your description and can immediately confirm that we're a good fit, during the first contact we can 1) provide you with *examples* of similar projects we've worked on, 2) suggest *the best way to execute your project*, and 3) if the project is similar to something we've already done, we'll give you a *ballpark cost estimate*.

Additional questions to answer

Where does your project currently stand? Is it at the "idea stage", or do you already have specifications? Or maybe part of it is already developed and you just need some help getting it across the finish line? Depending on your project's status, we may ask some additional questions. For example:

— What is your application about?

— Who is the typical user?

— What problem can they solve by using it?

— What are your monetization plans?

Your answers determine where our focus will be: design, ad placement, functionalities, user-friendliness, and/or reaching a certain goal (i.e. if the app is enterprise-oriented and not publicly available).

Other questions:

— Are there similar apps? If so, tell us what you like, what you dislike, and why.

— Who are your competitors and what is your plan to challenge them?

— If you don't have specifications yet, can you explain how the app should work? (In this case we would either create a document together and make wireframe mockups for the core functionalities, or define the core functionalities and start agile-based development from the "Minimal Viable Product" until we reach the final version).

What does the ideal "specification document" look like?

1) The ideal specification document is one that starts with a short (one page at most), *high-level description* of the application. This description should answer all of the questions we mentioned earlier: what the main idea is, what it accomplishes, who the users are, and how it differs from the competition.

2) The second section should consist of *wireframe screenshots* of the app. This a drawn version of how each screen will look in the app, how the transitions will work, and where elements will be placed. Wireframes are key for evaluating the work scope and dividing it into tasks.

3) The third section also *features wireframes accompanied by notes*, i.e. important things the provider should be aware of.

While a specification document is not an absolute must, it's a great way to get the client and the provider on the same page in a very short time.

Estimation process insights

Not all clients we work with need a precise estimate from the get-go. Some clients –

usually those already experienced with IT projects – are comfortable with a ballpark figure as a reference, in order to start development right away. Others prefer detailed breakdowns, with man-hour estimates and concrete deadlines for each task. This type of request is common amongst clients new to software development outsourcing.

So how do we determine the details that go into an estimate? Let's take a look.

Estimation method

"Estimating" is the process of predicting how much effort will be needed to develop an application. This effort, which can be measured in terms of man-hours or total budget, is the sum total of a given quantity of developers working together on the application.

We start the estimating process with something we call "judgment-based effort estimation", or *"expert estimation"*. This entails looking at past projects (and there are a lot of them: since 1998, we've delivered over 1,500 software projects) to get an idea of the possible risks and obstacles involved.

At Sibers, the "expert" is a team leader or a senior developer with 6+ years of experience. These are highly skilled people with dozens of shipped projects under their belt and a deep understanding of each team member's strengths and weaknesses, the complexities of various platforms, innumerable technologies, and much more.

The expert meets with the IT Consultant (your original point of contact) and together they envision the complete picture: has all of the key information been provided, what risks are involved, etc.

Speaking of risks…

Risks to consider

No matter a project's size or context, obstacles can appear during development that increase the necessary man-hours. These obstacles include:

1) If we're working with *pre-existing code*, there might be bugs or problems with the initial architecture we aren't fully aware of until we start the project. Sometime we can tell immediately if problems will arise or not, and other times – i.e. if we see signs of bad code – it's actually faster to redevelop the app from scratch. When working with pre-existing code, Sibers operates on an iterative contract basis, because even after a code investigation things can quickly become unpredictable.

2) *3rd-party influence*, i.e. if Facebook launches a new SDK forbidding use of a previous version, or if some 3rd-party service closes while the app is still in development.

3) *Requirements that weren't discussed* in the initial phase might become relevant later on. For example: the client forgot to mention that videos uploaded to the app need to do so within 10 seconds, after we already implemented a more standard solution, like 30 seconds. When something like this happens, features must be redeveloped.

4) *Developers' personal issues*: sickness, family issues, etc. For short-term issues, adding a new developer rarely makes sense because the newcomer will need time to get acquainted with the project and its architecture/code, so the development time would increase either way. However, for long-term absences additional devs are added to the team.

5) After providing the client with a first version of the application, the client has *more ideas and/or wants* to add or change features. Often this requires fixes that can cause problems down the line. If the client has enough budget and an understanding of the changes needed, we can rework most of the app to make sure everything works as intended, but a completely new deadline is needed.

"Expert estimation" is the most commonly-used estimation strategy for software projects because of its generally accurate results. For obvious reasons though, the more unique the app idea is the trickier expert estimation becomes. A unique idea means we can't draw on previous experience for the estimate, nor can we foresee complex issues. In these cases the most sensible way to proceed is to make a prototype of the most complicated features and see hands-on if the concept works.

In rare cases, the development of a specific app/functionality/feature can't be estimated at all. In these cases the provider would let the client know this right away, and provide a very rough estimate range.

Sibers tracks all of its estimates. When an iteration is completed, we compare the estimate to the actual hours/cost, analyze the comparison, and discuss it as a team. This process greatly benefits the estimating and hours we put in on future projects.

Let's proceed to mobile app development peculiarities that are unique to the estimation process.

Mobile-specific estimation examples

Key units

There are two "key units" to consider when providing an estimate for a mobile app:

a screen and *a feature*. Apps with difficult logic are usually estimated *by feature*, while others, which generally consist mostly of interface, are estimated *by screen*.

Simple logic apps are things like pulling weather forecast data or news items from a server, or sending a food request order. In these cases the app consists of the UI, representing a simple dataset taken from the server.

On the other hand, an app like Instagram employs complicated logic, allowing users to apply filters to pictures (among other things). Each filter is a separate feature that is estimated separately, and collectively these features constitute the app. Aside from the bottom menu, name, and icon, everything else in Instagram is the result of logic.

Some projects have a complicated structure — for example, when an app features a "friends list" that's used on several different screens. If the look of the friends list remains the same, it makes sense to break down the *app into reusable UI components*. Even some logic elements can be reused.

Returning to the Instagram example: the photo feed is a reusable component. Your photo is shown in several tabs: your own photo feed, your friends' feed, and the search feed. We develop it once and then add it to different screens. If the component has no variations, then within 10 minutes it can be embedded to a new screen. But if colors, font, edit options, etc. change, then more time is required.

If the app has large blocks of features working in unison towards the same goal (i.e. geolocation or chat), each block can be reused as a separate *module* (a module consists of significant logic and UI).

To quickly recap: a mobile app can be estimated by feature, screens, or modules (blocks of features) according to how many times each one appears within the app itself.

What's the average cost of the most widespread features, UI components, and modules?

There isn't a standard answer to this question, since every project is unique. So let's consider a sample feature: integration with server-side. If we have a weather app, it requires "fetching weather forecast data" – or in other words, making *dozens of almost identical calls* to a server. And we could have an app with a social functionality, that requires updating statuses, loading comments and messages, checking for mentions on photos and more — in other words, integration with server-side in this case means *sending dozens of different repetitive requests*.

So it looks like we can use a *quantitative approach* to estimation. Can we?
In the following examples we'll see several concrete units of mobile app development

(server-side measurement units), starting with API calls:

1) API calls

A request can be very simple – for example, counting "new messages". The server sends an answer to the app (i.e. "5"), and that's pretty much it. We have our answer and we can display it.

A more complex case is when we need to get the messages themselves. There could be lots of messages, so we'll need to compare the received data with the data stored within the app, excluding duplicates and deleted messages, adding new ones, etc. After this, we need to consider the user's filters.

Though *both of these examples are single requests* (one logic unit), the first implementation takes about *15 minutes* of work whereas the second is a *4 - 8-hour job*.

Another scenario is when the app becomes more complicated over time (i.e. it's a video-on-demand app and we have to request a program schedule from the server-side). When development started, there were only three channels and the schedule was simple. But then the business grows, the app's popularity increases, and there are now 80 channels, each with many programs. Also, instead of giving users the schedule for a single day, we want to give them a schedule for the next two weeks, and translate it into different languages for different countries.

When approaching something like this, it's imperative to divide large requests into smaller pieces, since not all data is relevant to all users. The number of requests raises tenfold, while the data volume of each request gets smaller.

Now imagine if we added "review" and "comments" options to each program. The number of requests would increase significantly!

2) Photo effects and filters

The *filter type* determines development time. If we're talking about a filter that turns the photo to black & white, 10 minutes of development are sufficient. But if we're talking about filters like those featured in the popular Prisma app (which uses a neuron network able to turn a picture into a work of art), development could take up to 200 hours.

A group of filters also determines development time. For example, filters in Photoshop are grouped by similarity. The first filter in each group can take up to 30 hours of development, but each subsequent one can be done in around two hours.

3) Internal messaging

When chat features, which all require custom design, notifications, and special details, are central to your app's concept, you face a dilemma: take a ready-made solution and customize it, or develop from scratch.

This decision is usually taken according to a given time/manpower threshold for "from scratch" features and changes: when this threshold is exceeded, customizing a ready-made solution is more convenient.

In general, a standard chat with a standard interface takes about 6–8 hours of development to be functional within an app. However, we once worked on an extremely feature-rich custom iOS messenger that kept our iOS and .NET developers busy for over 4,000 man-hours!

4) Social network integration

Social network integration (ex. Facebook) is a broad term. Implementation time can range from 5–100 hours, depending on the exact features to be integrated and *how customized the interface must be.*

5) Search

Search (normally divided into "search" and "filtering") is a very common feature, and in many cases it can be implemented in a standard way.

Filtering is easier to implement because it works with results you already have, i.e. a list of events for the evening, filtered by name. You start typing a name and the app takes the locally stored list and hides the items that don't match the search phrase. In the simplest case, this type of implementation only takes around two hours. Filtering that involves dates, names, and other parameters simultaneously can require a much longer development cycle.

And then there's the process of filtration itself. A text filter searching a moderate amount of data can be developed in a couple of hours, but if the search involves a database with millions of records, complexity can quickly escalate.

6) User Interface

When speaking about mobile development, the interface is always the most complicated part. There are two reasons why:

1) Only the developer knows what goes on in the backend. When programming the app logic, our main criteria is that the result should be smooth; at this stage, appearance doesn't matter. Then there's the design phase, where size, colors,

and placement of elements are discussed, and *tastes are compared*.

2) The interface must *look nice* and be *fast*. The various components' positioning is defined in relation to other components and the display's borders. Roughly speaking, one screen is displayed in about 1/10 of a second: this is also true if, for example, we're trying to display a table with dozens of cells, each with its own coordinates. As the user begins scrolling, ten or more new cells must appear on the screen per second. How to protect against slow-downs? With a couple of development tricks for better performance, of course.

Development tricks = better performance

A mobile app's performance/speed is only as good as the computational power of the smartphone it's running on. This is non-negotiable, which means that making an app run faster is a matter of implementing "tricks", whereby the developer creates *a way to make the app seem like it's working faster*.

For example, let's consider accelerating a photo upload.

The trick is that the app starts the uploading process before the "upload" button is clicked: the process is launched in the background after a certain requirement is met – a "point of no return" of sorts, after which 95% of users usually upload the photo from their gallery.

So, while a user is writing a comment to the photo, or choosing a filter to apply, the app has already started the upload process. When "upload" is clicked, the work is already 80% done. That's how you can manipulate perception and make it seem like the photo uploaded in 1/10 of a second.

Another example: a table containing a photo feed with a rather complicated interface (sharing, liking, commenting features, etc.). Since a user can't interact with content while scrolling through the image feed, we can show him a thumbnail view of the various "boxes". Then, when the scrolling slows down, these thumbnails are replaced by higher quality pictures, and the possibility to like, comment, etc. is enabled again. It's a trick, not actual acceleration, but the user doesn't know this. This angle is particularly useful for older devices.

A deep bag of tricks is a prerequisite for any good developer. Clients frequently receive widely-varying estimates from different providers for the same functionality. The reason for this is that some developers don't have the know-how/bag of tricks to find a suitable compromise, and instead pitch the most expensive implementation.

Conclusion: ok, so how much?

App development starts long before the coding process begins and ends long after the coding process concludes. *Pre-coding client/provider cooperation* is vital in order for both parties to clearly understand the project. Only when this understanding (scope, app functionalities, value for users, etc.) is found can the estimation process start.

Unfortunately, there's no clear answer for to the question of "how much?" Even though some modules, screens, and features can be approximately quantified, the individual characteristics are unique for each project.

Estimation is a highly-structured, time-consuming process undertaken by experienced developers. A *developer's previous experience* (and that of their team's) matters a lot. At Sibers, everyone involved in estimating is an engineer who can ensure that the project's coding is doable, what the project's conditions are, and what kind of timeframe is practical. The engineers' involvement is the only way to confirm that we're always on technology's cutting edge.

Generally speaking, a pro bono estimate for a small project (i.e. less than 500 development hours) can take 1–2 days. Bigger projects (ex. enterprise-level applications) can require up to two weeks for proper estimation. Despite the length, this is an investment of time and resources that Sibers is happy to make.

Our *estimates are quite precise*: on an iterative development basis, if we give you a 200 man-hours estimate for your application, you can bet that unless unforeseen circumstances arise we'll do the job in 200 hours or less.

The last thing we'll say is that it's a good idea to leave yourself some financial wiggle room. Ideally, you would spend half of your budget on coding and the rest on *testing and polishing*. While not always necessary, these latter two steps are invaluable in terms of building an app of the highest quality. Be prepared to spend a little bit more in order to get an app that exceeds your expectations.