codeScrum

Top 5 mistakes entrepreneurs and startups make with their outsourced IT teams.

March 09, 2015 (Revision 1)

Miguel Diaz
Author

Johan Tique
Editor

Summary

Many startups and entrepreneurs searching for an outsourced IT team to help them develop their applications end up frustrated and having many problems with the development process, as it becomes painful and inefficient to deliver results. In this paper, you will learn how to properly manage the relationship between you, as the client, and your outsourced IT team, as well as how to avoid and recover from the top 5 critical mistakes that entrepreneurs and startups make.

Know the symptoms, and what you can do about them

The first step in improving something is to detect the problems that have arisen between you and your IT team and which of those constantly drain the energy required to move forward in your project. If you are unsure where to begin, check which of the following may apply to you:

1. **Inefficient communication**

   The first problem relates to something simple, yet key for giving the project a steady start. This may sound obvious, but there are key differences between an efficient communication and an "always on" channel.

   Abusing the communications channel

   Teamwork tools built on the concepts of instant, always-connected and with interaction history are great for communication, and have changed how people collaborate on projects, making information transfer a lot easier. However, do note that these same benefits can work against accomplishing tasks if not used properly.

   Availability and instant communication channels are excellent when discussing options, open topics, and to get multiple people on the same page. They are also great when the situation demands discussing subtleties on a specific topic, but there are some cases where this should be avoided.

   Take the following into account when communicating with your IT team:

   - Try to keep communications focused on one topic: the current one.
   - Avoid constant updates if they are not related to the particular topic being discussed or worked on at the moment, because they will only be converted into distractions. It is better to write these particularities down (e.g. in a shared document or notes) to be discussed when the time is right.
   - Let the communication naturally emerge. Communication between the team and you goes both ways. You do not necessarily have to supervise them at all times. Whenever there is an update from the team, trust them, they will tell you. This also leaves time for you to work on anything you need to do yourself.

Asking for too much details

Precise information on ***what*** *is to be done* must not get confused with *detailed information on* ***how*** *it will be done*. Asking for too much detail on how things will either be fixed, developed or performed by the team just to be on top of things, will actually delay what you are trying to accomplish. Ask yourself whether you really need this kind of information (e.g. technical information or explanations) to get the work flowing. Its ok to be curious, but try not to get tangled with the specifics.

2. Inadequate estimation/work scoping

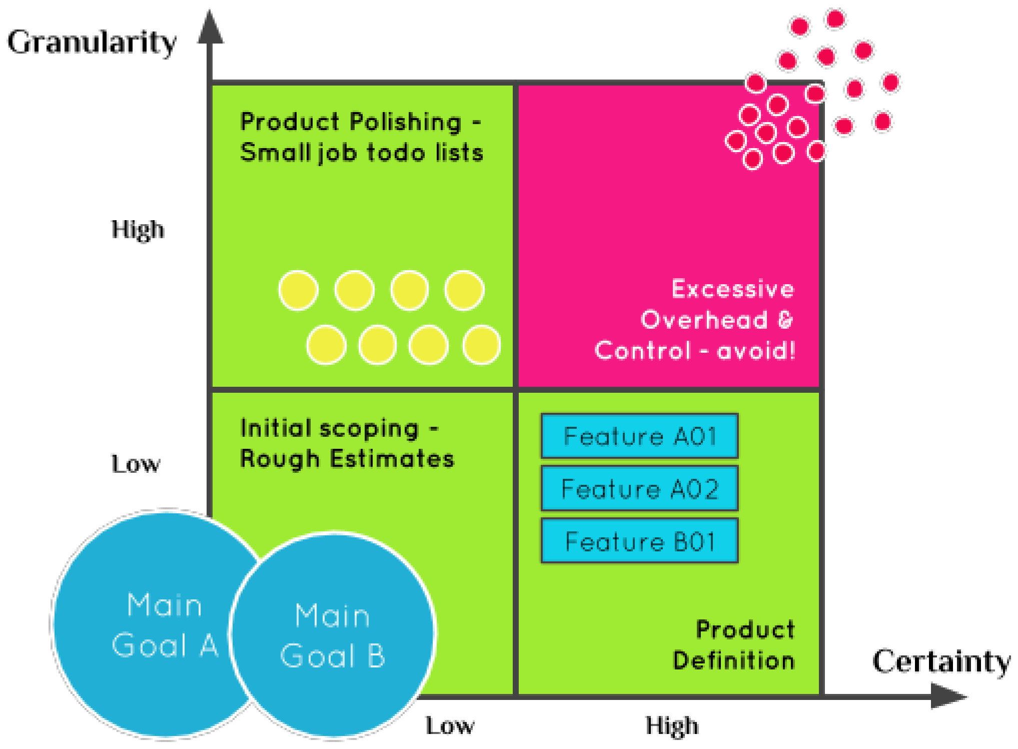
Handling expectations and work estimations is perhaps one of the most important parts of your relationship with your IT team. Estimation helps clarify what outcome you can expect to get, when and ultimately how much it costs you.

There are different "gears" for different types of work, and understanding this can help you and your IT team make the most out of any estimations you both require. These different "gears" are mainly classified by the level of granularity and certainty a particular estimation requires.

The overall idea is to determine the correct relationship between the size of what needs to be estimated and the certainty required for that given estimation. An estimation with low certainty can be done quickly; an estimation with high certainty requires a lot of work to identify each part of the solution and how it will be implemented in detail.

For example, an initial rough estimation for a piece of work may be given in days or weeks, with a low certainty (estimation error can be in days or weeks too). There is more certainty when it is possible to foresee specific features, without going too much into detail.

The graphic below describes a relationship in which both the granularity and the certainty are chosen beforehand, and shows how this maps into the different "gears" or estimation types.


Granularity
High
Low
Product Polishing - Small job todo lists
Excessive Overhead & Control - avoid!
Initial scoping - Rough Estimates
Feature A01
Feature A02
Feature B01
Product Definition
Main Goal A
Main Goal B
Certainty
Low
High

When a low granularity and a low certainty are chosen, you get rough estimates, good for an overall or initial scoping (for developing a product).

When a low granularity and high certainty are chosen, you get estimations suitable for describing the main characteristics or features of the product.

When a high granularity and a low certainty are taken, it means we get to-do lists, we want to keep track of what needs to be done. If small enough, you do not require estimates for each of these little changes, which is why this may be left for product polishing stages (i.e. before a launch to tidy up the product).

The critical case comes when both a high granularity and high certainty are chosen. What this means in practice is that you and the IT team estimate a large quantity of little changes and try to do so with high precision for each. This can quickly become a great overhead, and even worse if there is constant update on these details. The problem with this is that it tends to leak in the communication problems described before, with the added overhead of estimating and expecting every little detail to be on track.

3. Inefficient feedback loop

Another important element when working on an agile project is for the IT team to get feedback (and for you to provide the feedback) to see if everything is on the right track, making the necessary changes to correct any deviations. Here are some common problems that you have to be aware of when addressing this feedback stage:

Not having a designated time for feedback

Not having a designated time for feedback falls into a communication issue since giving constant feedback may get in the way and distract the work in progress. Agree on times with your IT team, to give feedback.

Multiple issues at the same time

Try to describe only one problem at a time, which will help focus efforts from you and the IT team. If the issue being discussed needs more work, then make it possible to resume feedback on other topics later.

Not really discussing feedback

Feedback gathers information from things that are currently in a visible/tangible state and does not have to do with other descriptions or work. The mistake here is making clarifications or updates about topics that the IT team has not yet even begun to work on.

Not summarizing the feedback

At the very end of the feedback gathering, take a moment to summarize the changes and make sure to log them into the product backlog as independent issues, so that nothing is missed.

Very short feedback loops

Quick feedback loops are essential to better align product vision, but it is not necessarily always true that shorter is better.

Feedback loops also have their sweet spots in terms of frequency for different types of work, but the majority of times, there is no need for constant supervision.

Give the IT team time to accomplish its goals. It is not necessary to initiate a feedback loop when first realizing something has to be fixed or changed. Also, these feedback loops should tend to be less frequent over time, on the basis that the product vision is understood and that more confidence is put into the IT team to deliver bigger chunks.

The IT team should ultimately come up with solutions to the overall problem you describe, without losing focus on the product vision and making full use of its expertise and becoming in practice, a part of your team.

4. Inadequate prioritization

Inadequate prioritization is a distraction that constantly changes what the IT team is working on. In the essence of a Lean Startup, pivoting is sure to be a way of seeking the right approach, but it has to be distinguished from not knowing what comes first.

If the priorities are changed frequently without allowing sufficient time to complete the tasks, then the backlog will be populated of unfinished items and it will be difficult to track progress and deliver value.

You need to be careful with prioritization issues that may signal a lack of clarity on your product's purpose. If this is the case, it is best to take some time and carry out research activities with your users and your market, to clarify what the priorities should be.

Prioritization issues can come a long way from the top of your business definition, however most of time they will end up being the result of inefficient feedback. By trying to fix the newly discovered bugs, you tend to naively putting them on top of the list without taking some time to figure out how they relate to all the other items of the backlog.

If you need a formula to help you decide on what to build, first you should build features that affect a lot of users greatly (an important feature) and that are quick and cheap to build. This will be the feature that gets the highest value with the formula **(u * i) / c,** where **u** is the number of impacted users, **i** is the relative importance of the feature (a number), and **c** is the cost of development in days.

5. Over-engineering and feature creep

You should avoid the mistakes of over-engineering and feature creep. First lets differentiate the two concepts.

Over-engineering is to design the system more complex than it should be. Usually, at the MVP stage, you want to gauge the user interest of a particular feature, so initially, you could implement a less robust version. For example, if you are providing a drop-down list, you could hard code the values of the list. Once you have a better indication that the user requires the drop-down list, you could build a database table in the backend and add some forms for configuration of the list values.

Feature creep is to continue adding more and more features to the system beyond what is necessary. At the MVP stage, having less features is better. Less features will facilitate your user understanding and interaction with your product. Additionally, less features will cost you less to develop and maintain.

Incredibly complete MVPs

In order for an MVP to be "successful" (even if it "fails"), startups need to strip as many features as they can and test the most important assumptions for the business. In order to avoid adding extra features, always challenge yourself how can the product be simplified and if what you ask the IT team is really needed. Remember the 80/20 rule (80% of all you need is supplied by 20% of the features) to avoid being taken over by this problem.

What to look for in an IT team

An ideal IT relationship will vary depending on the nature of the startup and what the specific needs may be, however here are some tips to keep in mind that may help you decide when comparing your options. Your IT team:

- Asks questions in order to understand your service and business
- Focuses on doing few things at a time, but doing them right.
- Helps you differentiate details from core features.
- Helps you keep organized by taking into account your general priorities and adjusting work for it.
- Helps you keep focused by avoiding discussing issues that can be left for later.
- Doesn't necessarily do what its told, proposes other ways to solve problems from its own expertise.

codeScrum

Claim a free quote to help you plan the delivery of your application.

Jairo Diaz
Director

jairo.diaz@codescrum.com

Click here to claim a free quote