A Software Development Guide for Non-Profits

Because a little bit of knowledge goes a long way.

Revision 00
20 May 2017
Look for the latest revision at www.oscarbaruffa.com/npsoftware

Who this guide is for.

If you are a Non-Profit, this guide is for you[1].You might be in an NGO, NPO, NPC, Profit not-for-distribution company, charity, or funding a good cause from your own pocket.

Why specifically these organisations? In many cases, nonprofits face pressures on all fronts. Investing in custom-developed software such as websites, web apps and mobile apps becomes necessary, but there isn't the budget or manpower for the organisation to afford any mistakes.

Sometimes, the nonprofit may receive the software development for free. Perhaps as a donation, or through some other channel. This introduces an additional challenge: the client (you) feels bad about complaining or being assertive.

Why you should use this guide.

The cost of using software far exceeds the cost of developing it. If you are stuck with poor software, you could be in a worse position than when you started.

Why do projects fail? Unmet expectations.

In business, as in life, the number one source of stress and frustration comes from unmet expectations. It doesn't matter as much whether we have high or low expectations. What matters is whether those expectations are met.

Developers and clients get very excited when looking at developing and implementing software that will improve processes, streamline operations, or even radically change how the organisation operates. This can lead to both parties forging ahead on the project without laying a proper foundation to ensure the software's successful development AND implementation.

However, anyone getting involved in software development for the first time might be surprised at how difficult it usually is. I think that this is because apps in our personal lives (for example) are seamless to include into our lives and just as seamless to remove. We assume apps and software in business[2] will also be as simple. However, business apps invariably interact with organisational workflows which make them much more difficult to adopt. If the software isn't rolled out properly, there is usually far more negative implications for the organisation than when a personal app isn't used.

The seeds of destruction are usually sown early on and can be found in one of two places:

[1] This guide will have value for any organisation, but this is tailored to non-profits.
[2] I'll use the term "business" loosely, as nonprofits operate in much the same way as for-profits when it comes to procuring services, managing projects or interacting with stakeholders.

1) The software isn't solving the right problem, or
2) Poor User-Adoption i.e. how readily staff or stakeholders will adopt the software into their normal working practices. Don't underestimate how much people resist change!

Both issues can be avoided by spending some time going through a series of questions to make sure each party understands what is expected of them. Clients and developers alike are entitled to certain expectations, and neither can succeed if the other fails.

You can use this checklist for each project as a way of documenting the expectations of each party. This serves two functions:

1) Clearly establishing the requirements of the software and;
2) Helping to identify when design requirements change, which may impact on the original expectations. This ~~almost~~ *always* happens!

I strongly encourage both clients and developers to insist that a document such as this is completed and agreed upon before embarking on any work. Even if you don't have all the answers, you will at least know what it is that you don't know.

I also recommend getting the first version out as quickly as possible - preferably mockups of screens that show how the software works. Software development, like any other creative process, is much better to collaborate on once there is something that can be looked at, interrogated, and commented on. Until that point, the software will be abstract idea.

How to use this guide.

Have a look through the checklist and the accompanying detailed explanations. Ask your developer as many questions as you want. Be nice, but be assertive.

Get references, see working examples of past projects. Speak to existing clients & get advice on how to communicate with the developer.

Seek advice from people. Googling can help you, but don't worry if the internet is more confusing than helpful. Software is often context-specific.

Be wary of developers who talk excessively about the technology they are using. This shouldn't matter much to you beyond that it is secure, is maintainable by any other decent developer, and doesn't cost an arm and a leg to use (e.g. licenses etc).

Developers are people too[3]. They make mistakes and can have bad days, but generally they want to produce good quality work. You can help them do so!

[3] Some of my favourite people are developers!

The Checklist

- ❏ Describe what the problem is. In detail.
- ❏ Describe what the result of a successful implementation looks like.
- ❏ Describe the current system.
- ❏ Number of people that will use the software.
- ❏ Number of User roles required with description of each.
- ❏ Complete a User Story for each role.
- ❏ What types of devices and operating systems will the Users be using?
- ❏ Will Users be using the software across multiple devices?
- ❏ Which authentication methods will be used?
- ❏ Client-side Project Manager details.
- ❏ Developer side point-of-contact.
- ❏ Test group details.
- ❏ Has the test group been briefed?
- ❏ Rollout plan for implementation.
- ❏ Has client agreed to that Users will not use old system in parallel with new system?
- ❏ Detail all the costs involved, who is paying for what. Don’t assume anything!
- ❏ Is a Transaction Agreement in place?

Keep adding to this list and share your experiences with others.

The Detail

Note: Software is a broad term. It covers apps (like on your phone), web apps (apps that run on internet browsers) and web sites. For this section I will just use the term “app” as it covers most of the questions quite well.

What problem are you trying to solve?

What result do you expect to see when the app is rolled out and being used? What does a successful implementation look like?

A detailed description of the problem and what you will consider a success is the most valuable but often-forgotten question to answer.

Describe the current system being used.

What works? What doesn’t work? No business app works in isolation. Context matters. The developer will need to use a lot of imagination to put themselves in the shoes of those using the app and those managing the business, based on the results of the app.

The more information that is at hand, the better the developer will be able to anticipate the needs of the business.

How many people will be using the software?

This has implications for subscription plans, hosting plans and how the app will be rolled out across the organization.

It’s assumed that ongoing support will likely be required. The developer can correctly price for this support when the number of users is known. As an example, the level of support needed for an app with 3 users is vastly different than for an app of 300 users, and even more so for 3000 users. Contract agreements will need to reflect this.

How many User roles will there be?

Each role can be defined as a functional role of the user. For example, Managers, Grant officers, Social workers, and Administrators are 4 separate roles, because each is likely to need to access different information.

Keep the user roles to a minimum. The less roles there are, the less complicated the app will be and the easier and faster it can be developed, debugged, amended, and maintained.

Have User Stories been completed for each role?

User stories are a useful way of clarifying what it is that the user will be able to do with the app.

A user story will describe in as much detail as possible what a typical user will be able to do with the app. This must be completed for each user role. It can be written in either the first or third person, but I suggest writing it in the first person, as people seem to have a better chance at visualizing the required workflow properly when picturing themselves doing it. For example, say “I am Social Worker, Bob Smith. I when I start my workday I open my app to see my list of scheduled visits. I can see the open, outstanding and completed visits in one view.” and so on. Then the developer can ask “Well, what if there are hundreds of completed visits?” Then it’s easier to identify that only X number of completed visits should show. Or only the completed visits from the last 7 days should be displayed.

What devices will they be using?

Android and iOS? Phones and tablets? Desktops? These are important considerations for the developer. There are many ways to design the user interfaces, and the type of devices used will have an impact on many design choices.

Will any users be using the app across devices?

When a User uses an app across multiple devices, this is called multi-tenancy. For some types of software, multi-tenancy does not have as large an impact on the design as for other technologies. However, multi-tenancy may affect some ways in which data is collected, sorted, or represented.

What authentication method will be used?

This refers to the method which will be used to restrict access to the app. It is very important for the developer to know this upfront to set up their own environment to match. Do you need to restrict access to just people in your organisation? Will people from multiple organisations need access? Are you ok with using Social Authorisation (twitter, facebook etc) ?

Who is project managing from your side?

Ideally someone with expertise regarding the problem at hand should also manage the project from the client side. It is also advised that you provide the developer with a single point of contact and a single designated authority who can sign off on changes to the app. Multiple points of contact often lead to chaos.

Has a dedicated test group been identified and briefed?

Select a dedicated test group who commit to providing timely feedback. This will catch a lot of bugs and workflow issues upfront, it may even prevent them from reaching the production (i.e. real-world roll-out) phase entirely. Your project manager should brief the test group. Nothing is worse[4] for the developer than having to create buy-in from your own staff to adopt the app.

Develop a roll-out plan and get commitment for system exclusivity.

For a large deployment, consider rolling out in phases. Each group that uses the solution should use the app exclusively. Running in parallel with the old system may result in Users not giving the new system the attention it requires, resulting in low user adoption and ultimately failure to implement the app. People often prefer using an old, familiar system whose flaws they understand than a better system whose flaws they don't understand.

Is an Transaction Agreement[5] in place?

This can apply to all projects, but is especially needed for anything that the nonprofit is not paying for.

This should include the expectations of the paying party e.g. what the donor gets in return (media exposure and so on) which will make it clear that just because money isn't involved doesn't mean it's not a transaction.

Your nonprofit can expect and must demand the same level of professionalism, ethics and respect as any other business would, regardless of whether you are paying for the service or not.

A Final Word

Digitising your nonprofit's processes hold immense value. Yes, it can be a bit intimidating - but once you've gone through this exercise you should feel a whole lot more confident. Knowledge is power.

[4] Well, not *nothing*.
[5] This isn't a legal term, I just wasn't sure what to else to call it.

About the author

Thanks for reading this guide, I really hope it helps you achieve your nonprofit's mission!

I've spent 6 years working in the private sector and 4 years in the nonprofit space. I've been a user of software forever, built a few apps (even had a few people using them in the real world) and given many people my opinion on how to solve their problems - sometimes they've even asked for the advice :).

If you found this guide useful and want to say thanks please do. If you have any comments on how to improve it please let me know as well.

You can find me on:

Twitter: https://twitter.com/oscar_b123 (@oscar_b123)

Linkedin: https://www.linkedin.com/in/oscarbaruffa/

Email: oscar@oscarbaruffa.com