How to buy software

A short step–by–step guide

Mark Berthelemy

How to buy software

A short step by step guide

Mark Berthelemy

This book is for sale at http://leanpub.com/howtobuysoftware

This version was published on 2024-06-06

Contents

Introduction to this guide

You've decided your organisation needs a new technology solution. Whether it's to help you manage events, content, people or whatever… it really doesn't matter.

What does matter is that you need to make sure the solution you buy is right for your organisation:

- Will it work?
- Will it last?
- Will it fit?

Traditionally, this will be done by creating a Request for Proposal (RFP), containing a description of what you need, often with a long shopping list of key features. The solution supplier will then write a response, which answers all your questions, in a way that the supplier hopes will sell their solution to you.

Then you'd gather some people to review the responses, score the answers, and identify a preferred supplier. After that it's just a matter of agreeing the terms of the contract…

It sounds simple and straightforward. The trouble is, this process really doesn't work. At least, not as well as you might hope.

Both the supplier and the buyer organisations are working in the dark. The suppliers are making assumptions galore as they answer the questions - often without really knowing much about how your organisation ticks, and what you really need. And you, the buyer, are buying something that you're hoping, without much evidence, will work in your particular situation.

All those assumptions and guesswork are very likely to come back to bite you during the implementation phase of your procurement project.

Many organisations are starting to adopt a more realistic procurement process, known as "competitive dialogue". It's a way to iterate towards a solution that will work for both the buyer and the supplier.

You wouldn't use it for every situation, but for the more important, organisation-wide purchases, it's one of the best ways to buy software (or any service in fact, but that's outside the scope of this guide).

Who is this book for?

This guide is designed for those who buy software, particularly multi-user SaaS* products, for organisations.

> SaaS = Software as a Service, usually accessed through a browser over the internet.

It's not designed for procurement experts, but instead its for the managers and users who are tasked with making the decision.

If you **are** a procurement expert, then welcome! You might find that I've simplified some things a fair bit, or even left things out that you consider important. Please let me know. This book is in a state of continuous improvement.

Stakeholder team

For smaller projects, if you're the only person involved in the buying process, you might be able to merge some of these steps together. For example, steps 1-5 can work well if you consider them as a whole, and iterate around them a few times. You'll still need to cover all the bases described here, even if it's just in your head.

Even so, one of your first jobs will be to identify the stakeholders.

- Who will be signing off on the budget, on the decision to purchase, and on go-live?
- Who will be using the software?
- Who will be using the outputs from the software?
- Who will be providing inputs to the software?
- Who will be connecting to the software?

You'll need to involve all these stakeholders at some point in the process.

For larger projects, you will need to gather a team around you to support the process. That team should comprise, at a minimum:

- The executive sponsor - the figurehead who will sell this to "the board"
- A representative from IT - who will help you to smooth the path to integration into the organisation
- An internal procurement specialist
- One or two prospective "champions" who will help to identify the requirements, sell the solution to users, and help to lead the implementation

Ideally, you'll also have someone, possibly external, who knows the market for what you need, and can help avoid some potential pitfalls.

I'd advise that you should keep the wider stakeholder group involved - or at least informed. So don't forget to incorporate communication in your planning. How this works will depend very much on your organisation's structure and culture.

What are we going to cover?

The guide contains a sequence of steps designed to take you through the decision-making process:

Step 1: Identify the problem

What is driving this purchase? How will you know whether the new software has solved the problem?

Step 2: Identify possible solutions

What are your options? Could the problem be solved without new software? What have you tried so far?

Step 3: Make a long list

What does the market look like? What types of software are available? What are the common features? What are the things that would stand out for you?

Step 4: Create user scenarios

Who are your users? What will they do with the software? How does it need to help them?

Step 5: List the requirements

What are your high priority functional and non-functional requirements?

Step 6: Make the short list

Which suppliers look like they'll meet your requirements?

Step 7: Competitive dialogue

Have a conversation with the suppliers - digging deeper on each iteration, as you learn more about what you really need.

Step 8: Negotiation

What does a good deal look like? What can you tweak to make the deal work for both parties?

Series introduction

This guide is one of a series of small books designed to help organisations through key decision points.

They are concise and easily digestible.

You should be able to skim through each guide in less than an hour. To implement the ideas contained within them might take a bit longer!

If you have any feedback or ideas for additions to the series, please get in touch.

Step 1: Identify the problem

> The answers to steps 1 and 2 will form the bulk of the **business case** that you'll probably need to write before beginning to look for suppliers.

Before you can even start to think about what you need, you must have a reasonable idea of:

1. The problem(s) you're trying to solve
2. What to measure to show whether you are solving the problem(s)
3. The current size of the problem(s)

All of these are incredibly important, not just to you, but also to potential suppliers. They'll help in the purchasing process and afterwards, as you continuously evaluate whether things are getting better.

Let's dig into those questions a little.

The problem

Why are you considering the purchase?

What, or who, is driving it?

Is this an urgent problem, requiring a quick and dirty solution? Or is it more strategic? Are you thinking short or long term?

Strategic problems might include:

- Low efficiency
- Too many errors
- Poor consistency over time, or across the workforce
- Non-compliance with regulatory and legal requirements
- Customer expectations not being met

Measuring the problem

Measurements are important, so that you can gauge whether the software has made a difference.

Let's consider the strategic problems above. What could you measure for each of these?

Problem	Possible measurements
Efficiency	Throughput and costs
Errors	Numbers and types of errors
Consistency	Cost of fixing issues
Compliance	Risk and cost of being fined
Customer expectations	Sales revenue and cost of handling complaints

The measurements don't have to be complicated, but they should be meaningful. Don't just measure things because they are easy to measure. You'll probably end with people trying to "game the system".

The size of the problem

Setting a benchmark from the outset will help you to a) talk sensibly to suppliers about the issues you're facing, and b) measure whether the solution has resolved the problems.

If you can't currently measure what you should be measuring, then try to approximate. And make sure that any new software is built to help you capture the measurements you need.

Problem statement

Now you should have enough to create a problem statement that outlines the issue, its size and implications to the organisation.

Step 2: Identify possible solutions

You've probably already started looking at what might work to solve the problems you have. You'll have seen marketing materials, or stands at conferences, or maybe you've been cold-called by a product representative.

But let's take a small step back.

Consider these questions to help in your thinking and your future conversations with potential suppliers:

- What have you tried so far? Do you have a cobbled-together solution built on Excel or something similar?
- What has worked? What doesn't work? What characteristics would you like to keep from the existing solutions?
- What have you learnt in the process? What needs to improve from your current solution? Perhaps consider things like robustness, reliability, multiple users with multiple roles, version control etc… These will depend very much on your context.
- What are your competitors or partners doing in this space? How are they solving similar problems?
- What does your current "landscape" look like? In terms of other software, your IT infrastructure, current regulations and standards, your culture, your finances, and your organisation's existing capabilities. What will your new solution need to fit into?
- What **types** of solutions seem to fit your problem? Is there a specific genre of off-the-shelf software that you're looking for? Can you filter that genre into sub-groups. For example: You might want a system to help manage learning experiences, but you're working for a training company. That immediately narrows the field.
- What is your vision for the future? Consider changes that may be happening in society, technology, your organisation, your customers etc.
- How agile can you be? Are you able to manage continuous change well? Do you want something that works out-of-the-box? Or do you need something that is highly configurable or even bespoke? Or do you want software you can customise?
- Do you want an all-in-one solution, or a combination of different best-of-breed providers? Do you have the skills to join them together?
- What is your proposed budget? How much will this new software save financially, or how much will it increase your revenue?

With these answers, and the problem statement from Step 1, you now have enough to create a formal business case to request funding.

Without that formal approval, many suppliers are unlikely to take you seriously. With it, they're more likely to invest in the sales process. This will help the future iterations proceed more smoothly.

It might be helpful to work with a solutions architect during this step. A solutions architect is someone who can outline possible types of solutions that might fit into your organisation. They should also be able help formulate the business case. They are often generalists, able to look at the problems you're facing from a number of angles.

The open-source advantage

Open-source is a specific philosophy of software creation. Unlike proprietary software, the "source code" is freely available to use, inspect and change.

Instead of paying for a license to use the software, you generally pay for services, such as consultancy, customisation, support and hosting. You could, in theory, use it without paying anything, but that would come with too much risk for most organisations.

Open-source projects tend to be more aligned to open-standards for storing and transferring data. If this is important to you, then open-source may be the answer.

Buying open-source software

With closed-source (or proprietary) software, access to information is usually tightly controlled by the vendor.

With open-source software, there is nothing to stop you taking it and trying it out. You'll have full access to the code and documentation. That will give you a really good idea of the quality of the project. Is the documentation well-written and up-to-date? Is the software mature and easy to setup?

The larger open-source projects will often work through implementation partners, giving you a choice of who to support you. In this case, you'll need to consider both the fit of the software for your requirements and the choice of implementation partner.

Step 3: Make the long list

By now, you know the problem you're trying to solve, and you have an idea of how a new solution might help you.

You also should have formal approval to go ahead and start looking for solutions to buy.

So it's time to start looking for, and talking to, potential suppliers.

If you've ever bought a house, a car, a computer, or made any other significant purchase, it's likely you'll follow a similar iterative process: start with the long list. Consider what you like and don't like. And identify what's important to you.

Where to look

Ideally you'll have someone around who knows about the market for the type of solution you're looking for. They'll be aware of the key players, and will have an ear to the ground for what's available.

If you don't have someone, then here are a few starting points to consider:

Search engine generic search

There are a few approaches to searching:

- **By genre**: Type in the genre of software you're looking for. For example: "Project management software"
- **Alternative to**: Start with one of the major suppliers, and ask for alternatives to it. For example: "Alternatives to [name of product]"
- **By question**: Other organisations will have the same problems you do. Type in a question that will find their solutions. For example: "What software will help me comply with GDPR?"

If you're looking for free software, don't forget to also include *"open source"*. Open source vendors often don't have the marketing budget to compete with the commercial vendors, so it's unlikely you'll find them via adverts.

Whilst it might be tempting to get all your answers from an AI chatbot, I'd recommended using a normal search engine for this process. It's much more environmentally-friendly, and the results won't be much different from a standard search.

Search engine adverts

Type in the name of one, major supplier and you'll be given adverts for a whole host of competitors. Try this a few times to see what comes back.

Be aware of articles written by suppliers that are written like independent comparison sites. They're using a carefully chosen set of criteria, and often not comparing like with like. Of course, their product always comes out on top. These articles can be useful in your overall searching, but treat the opinions with a pinch of salt…

Comparison sites

Sites like G2[1], Capterra[2] and Alternative To[3] can be useful starting points.

They can also be a little frustrating. The feature lists are often provided by the suppliers, with very little indication of how and how well those features work.

Again, use these sites as a starting point for your own research.

Analysts and consultants

Different industries have their own specialist analysts. For example, for learning technology, you could look at Fosway's 9 grids[4]} or Craig Weiss' Find An LMS[5].

Check out some of the independent consultants in your industry too. They will often write reviews of systems, or create lists. Make sure to consider the factors that they include in their reviews. These can be a useful guide to generating your own list of requirements.

Be aware that product information becomes dated very quickly. Anything more than a couple of years old is probably not worth using.

And try to find out about the business model of the analyst/consultant. If they receive commission, referral payments, or payments for reviews, then they're not independent.

As before, they can be a useful entry into creating your own list of potential solutions.

Conferences and exhibitions

These are a great place to see what's around, talk to the vendors, and, even more importantly, talk to their customers.

Look out for those vendors that know how to ask you questions. If they ask you the same questions that you've been asking yourself, that's a good sign!

[1] https://www.g2.com/
[2] https://www.capterra.com/
[3] https://alternativeto.net/
[4] https://www.fosway.com/9-grid-2/
[5] https://www.findanlms.com/

One thing to be aware of though, is that events like these tend to promote what is currently fashionable - the new, hot technology. You might need something that is more down-to-earth. So try to get beyond the glitz and find the suppliers that understand and may solve your problems.

Peers

Talk to your peers. Find other people in similar organisations and see what they're doing. Hopefully you'll have a decent network already outside of your current employer. Make use of it.

But don't just post a message on a social network asking for recommendations for a particular genre of software. For one thing, you'll find it'll be mostly suppliers that come back to you.

More importantly, unless you can be specific about your context and the problems you're trying to solve, the answers that you get will be less than useful.

Things to consider

As you create your long list, consider the following:

- **Features** - does it do what you think you're looking for?
- **Pricing models** - are they even available? If not, then are you big enough for that supplier to deal with?
- **Support & training** - can you see any documentation? Is there free training available online to give you a feel for what's on offer?
- **Usability** - can you get a feel for how easy the software is to use?
- **Accessibility** - do the suppliers even mention this?

By the end of this step, you should have a considerable number of options to choose from (unless you're in a very niche market of course!).

It's now time to start testing that list, and whittling it down.

Step 4: Create user scenarios

Scenarios are one of the best ways to test software in near-real-life conditions. First, they'll help you to think about what's really important. And, secondly, they'll help you and the suppliers to quickly assess your needs against what they can provide.

A scenario is a short story, with characters, that either describes an end-to-end process, or picks out, in more detail, a specific part of a process. They should outline the experience you expect for your users.

Generally, I advise clients to pick three or four key processes that they want to test out. Ideally you would pick areas that are unique to your particular context and needs.

For each scenario, you might want to consider the following sets of questions:

Processes

- What are the main processes that the software will support?
- How do those processes currently work?
- Where are the current pain points?
- How would you like the processes to work?

Users and organisations

- Who will be the key user groups? Eg. End user, Client administrator, Client managers, Site Administrator.
- Will you need to break down the users into further sub-groups? For what reasons?
- How is the organisation's structure reflected in those user groups?
- Where are the touch points between the processes and the user groups?

User onboarding

- How will your users be brought into the system?
- How will they receive their login details?
- Will you be connecting into one or more Identity Providers (like Microsoft, Google, Facebook etc)?
- Will users self-register?
- Will users pay for their own access? How?
- Will users pay on behalf of other users? How?

Access control

- Will some users have access to some parts of the system, whilst other users have access to other parts?
- Who will need to see data about users from the organisation(s)?

User tasks

- What will prompt users to login? How will they receive that prompt? How often?
- Once a user is logged in, what do you expect them to be able to see and do?
- How they will know what they need to do?
- How will they see what they've done?
- Will their journeys be linear (following a standard path)?
- Will users be able to choose their route? How will they be guided?
- Will all users follow the same route?

Reports

- What reports do you need to be able to see?
- What reports do your users need to be able to see?

Now turn all that into a set of short narratives, like the example below.

Sample scenario

We sell online learning products to other businesses. The products are created in an elearning authoring tool. When a company buys into our service, we bring them onboard quickly - setting up access routes and licenses purchased. They will buy, say, 100 licenses. Any employee in the customer's company should be able to access the content easily, through single-sign on. When all the licenses are used up, additional employees will see a message informing them to request further licenses. Our platform administrator runs a report each month detailing the number of licenses purchased and used by each customer company.

Step 5: List the requirements - first draft

Now that you've examined what software is available, and have developed your scenarios, you can come up with your list of requirements. Try to be as specific and descriptive as possible.

A list of features on its own can be pretty meaningless. They gain more meaning when you say how you expect the feature to work for you, and why they are important.

This is the first pass. It will change when you've tested the requirements against what is available on the market.

I recommend marking each requirement with a priority, perhaps using the MoSCoW model:

- M = **Must have** when you first launch
- S = **Should have** when you first launch, but can defer to the next iteration if necessary
- C = **Could have** when you first launch. They will add value, but are not essential
- W = **Won't have** when you first launch, but it would be good to see it on the roadmap

> Don't forget to think about the future at this point. What is happening in the technology world in your context? How important is that to you?

Each of the categories below contains a series of questions to help your thinking. Some may not be relevant to your needs, so use at your discretion.

Non-functional requirements

Performance

> *concurrent* = the number of people actively using the system at the same time

- What is the maximum expected response time? (eg. pages should load within 7 seconds on a 4g connection)
- How many concurrent* users will the application need to support? (eg. support at least 2,000 concurrent users).

For many types of system under voluntary use, concurrency would average out at less than 10% of the total number of users. When usage is compulsory and there are deadlines in place, the concurrency level will increase sharply.

Scalability

- How do you need the application to respond to unforeseen peak demands?
- Are you happy for it to gradually stop working?
- Should it put users in a queue?
- Should it scale automatically to allow everyone to access with no degradation of service?

Give examples of what might happen to cause peaks in usage - such as deadlines or marketing campaigns.

Availability

- What is your expected minimum uptime percentage? (eg. 99.99% availability). Does that measure include scheduled maintenance times?
- How quickly do you need to recover the system should something fail? (*Recovery Time Objective*)
- How much data can you afford to lose should the system fail? (*Recovery Point Objective*)

Security

- How frequently should the application be tested for vulnerabilities?
- Do you require compliance with standards like PCI-DSS, ISO 27001 or Cyber Essentials

Usability

- Do you require a responsive interface for both end users and administrators?
- What level of accessibility standards do you expect? Consider the W3C Web Content Accessibility Guidelines and their levels A, AA and AAA.
- Do you need the application to have been tested for accessibility by a third party such as Shaw Trust[1]{:target="_blank"}
- Do you require the application to work in multiple languages? Which?
- Do you need the application to be able to handle content or information that you enter in multiple languages?

Maintainability

- Will you want to see evidence that the software is continuously maintained and kept secure?
- What sort of evidence would you accept?

You might want to consider things like issue reporting mechanisms, security announcements, git commit history, software roadmaps, and live system status reports.

Environmental impact

- Will you need suppliers to provide evidence that they are minimising their environment impact?
- What types of evidence would you accept?

Consider evidence like travel policies, and datacentre Power Usage Effectiveness (PUE) and Data Center infrastructure Efficiency (DCiE) measurements.

Regulatory compliance

- Does your industry have specific regulations around things like data immutability (often seen in finance applications), accessibility, security, data protection or data transfer standards. You'll need to note them here.

Internal compliance

- Does your organisation have specific policies around software purchases and technology integrations?

[1] https://www.accessibility-services.co.uk/

Integration requirements

Integration means different things in different contexts:

Integration: Look and feel

- Does the new software need to fit your organisation's design patterns?

This is especially important in public-facing applications, like government agencies, ecommerce, training or content management.

Integration: Single sign on

- Does the supplier need to support single-sign on against your preferred identity provider (IdP), for example Microsoft, Google, Facebook, Octa, Auth0?
- What standards should the supplier support?

Integration: User provisioning

- Should the software allow automatic provisioning through systems such as Active Directory?
- Or must the software connect to your HR or CRM system?
- Should this be automatic and continuous, or in batches overnight, weekly or monthly?

Integration: Data transfer

- Will the software need to communicate transactions to your finance system?
- Will it need to send records back to your HR, CRM or other operational systems?
- How often should the data be passed across?
- Do you require the data transfers to be confirmed by the receiving system?
- What do you need to happen if the system detects an error in the transfer?
- What will you need to import to or export from the system? In what formats?

Integration: Analytics

- Will the supplier need to make data available to your data warehouse / data lake / analytics toolkit?
- Will the users of the analytics accept data that may be 24 hours old, or does it need to be "live"?
- Are there any standards you expect to comply with (eg. xAPI in the learning field)?

Reporting requirements

- What reports will users need to see? For themselves, for their teams, about their customers, about their organisation?
- Consider the key metrics you identified earlier. Do you expect the new software to help you generate those?

Durability

When you're investing a lot of money, you'll want to know that you'll get good value from it before the solution dies a natural death.

If it's software, you'll probably want to be assured that the language or framework they've chosen to build on has solid foundations, a ready pool of developers, an easy-to-implement method of upgrading, extensibility through standard APIs, and can be used by as many hardware devices as possible.

For example, it's unlikely you'll want to buy anything that relies on browser plugins, or other non-standard technologies. It wasn't long ago that organisations bought wholesales into Flash and Silverlight, and then ended up having to rapidly change their approach as these technologies lost support.

Stability

You'll want to have a good idea of how strong the vendor company is. Consider whether the market is currently consolidating.

It's obviously hard to predict which way things will go, but look for companies with a solid business plan and a strategy that makes sense. After all, you're investing in their long term future.

Pricing model

- Are you looking for a supplier with a transparent pricing model? Or do you have enough potential users that you're OK to go with an "enterprise" supplier, with whom you'll negotiate a price?
- Does your budget look like it will be enough to cover the costs?

Functional requirements

This is where you need to list out the most important things for you, along with why you need them, and how you expect them to work.

Many companies simply create a huge shopping list, which is of little use to the supplier or the customer.

Instead, go back to your scenarios, and consider the functional areas that you're interested in. They'll vary depending on the type of software you're looking for. Let's take the example of project management software. In this case you might have functional areas like:

- Planning
- Collaboration
- Financial management
- Project visualisation
- Project reports

For each one of those describe your expectations.

A typical functional requirements sheet might look like this:

Requirement ID	Description	Rationale	MoSCoW	Compliance level	Notes from the supplier
R001	Able to login with Google Workplace account	We use Google as our internal identity provider	Must have	Meets in all respects	
R002	Provides a simple mode for new users, and advanced mode for experienced users	We want to reduce the adoption time	Could have	Meets partially	Users are provided with automatic orientation when they meet new functionality

Requirement ID	Description	Rationale	MoSCoW	Compliance level	Notes from the supplier
R003	Provides an AI chatbot to provide help on demand	We want to minimise support queries	Should have	Not currently met	This is in our product roadmap

Step 6: Make the short list

Now you're going to need to go through your long list and assess each supplier against your requirements - based on what you can glean about each supplier so far.

This does assume that you can find out enough from your desktop research. That will often be true where there is strong market with many suppliers.

For complex, enterprise level applications, where there are only a handful of suppliers on the market, you may not be able to do more without talking directly to the suppliers.

Go through the **Must haves** first and remove the suppliers that fail on **any** one of these.

With those that are left, then go through the **Should haves** and remove the suppliers that fail on the higher priority items.

Then do the same with the **Could haves**, until you end up with about 5-7 potential suppliers.

It's not always quite as scientific as the previous statements might lead you to believe… If you really understand the requirements then you'll be able to do a lot of this on gut feel - based on what you can see from the suppliers' websites.

If you're involving more people in the process, then you'll need the more rigorous and formal approach.

Step 7: Competitive dialogue

Now comes the point at which you start to engage with the suppliers (if you haven't already). Take your shortlist and arrange an initial meeting with each sales representative. During this call they *should* be asking you questions. If they simply do a presentation or demonstration then you might want to consider dropping them, unless they're already near the top of the list.

Ideally, during the initial call, you'll have a chance to explain what you're looking for, and what's got you to this point. That's steps 1 and 2 of the process you've followed so far.

You'll also need to set out to them what will happen next. This is the process of *competitive dialogue*.

> Some customers will do a procurement exercise based on a single *Request for Proposal (RFP)* and the suppliers' responses.
>
> I've always found that to be less than satisfactory.
>
> Both parties are making guesses and huge assumptions that will only be uncovered once implementation begins.

Competitive dialogue allows you to tease out assumptions beforehand. Yes, it takes longer than a single Request For Proposal, and may, therefore, be more expensive. But the end results are more reliable.

The competitive dialogue takes the form of at least two iterations. The higher the stakes, the more iterations you'll need. At the end of each iteration, you'll disqualify at least one of the suppliers. As the process goes on, the remaining suppliers become more invested, and you can ask them to contribute more effort.

Iteration A

Provide the suppliers with a document containing:

- The problem defined in **step 1**
- The types of solutions you've considered in **step 2**
- The scenarios developed in **step 4**
- A description of the competitive dialogue process, including stages, key dates and what will be required from the suppliers at each stage

Provide a separate document containing the draft non-functional and functional requirements (from **step 5**), listed with their rationale and prioritizations. Let the suppliers know that this may change before they receive the final list.

Ask the suppliers to prepare a demonstration of their software addressing each of the scenarios you've described. Maybe give them a week or two to get this ready. A good sales team will often ask for a pre-meet to go through any questions they have about the scenarios.

Use the demonstrations as an opportunity to dig into assumptions you're making. Hopefully the suppliers will be doing the same!

You'll probably find you can easily remove a couple of suppliers from the list pretty quickly at this point. Especially those that just give a standard demonstration without addressing your scenarios…

Iteration B, C, D etc

You might want to ask for more demonstrations on specific functionality or conversations about the non-functional requirements where you or the suppliers have questions.

Don't be afraid to edit the requirements as you work through the process and understand more about what you need.

Aim to whittle the suppliers down to a list of three at most.

Final iteration

This is when you do two things:

1. Send out the Request for Proposal (RFP). This is a formal, legal document, that will need to comply with your IT procurement policies. The suppliers will be expected to provide a formal response, including their prices.
2. Begin a 4 week Deep Dive into the software, to try to give it a real-life test. You're unlikely to cover everything, but try to make it do the end-to-end process (eg. planning, collaborating and reporting on a project portfolio, or creating a training programme that leads to a certificate).

Don't skimp on the the Deep Dive. This is why you really only want to do it with two or three suppliers at the most. It takes a lot of effort. So you'll probably not be able to do much other work during this period.

The RFP should contain, at a minimum:

- A formal invitation for a proposal
- Instructions to the bidders - about the process, the timeline, and any conditions and how you need the proposal submitted
- A description of the problem you're trying to solve

- The final functional and non-functional requirements
- Company requirements (eg. references, accreditations, financial status, project management methodology)
- A pricing template, so you can compare like with like
- A security and data protection questionnaire (speak to your IT security specialist)
- How the proposals will be evaluated

Be realistic in your timescales - for yourselves and for the suppliers.

Once the proposals are in, you should be able to evaluate them against your criteria and then narrow the field to one, preferred supplier.

Step 8: Negotiation

Then it's a matter of coming to a deal. There are whole books written on this by far more qualified people than me. So I won't go into detail.

The aim is to get to a point where both parties feel like they're getting value from each other, and neither feels hard done by.

You might find points of negotiation, such as:

- Numbers of users
- Whether licenses can be recycled, or whether they are tied to individuals
- When the users will start to use the licenses
- Sweeteners like additional functionality, professional services or access to content (if applicable)
- The length of the contract
- Break clauses, allowing both parties to walk away without penalties mid-contract

When you're at a point where you can both sign the contract then it's time to have a party, and get ready for the implementation. Hopefully with no surprises waiting for you around the corner!