Additional Topics, Easter 2010

Developing Commercial Software

Red Gate Software Ltd

James Moore

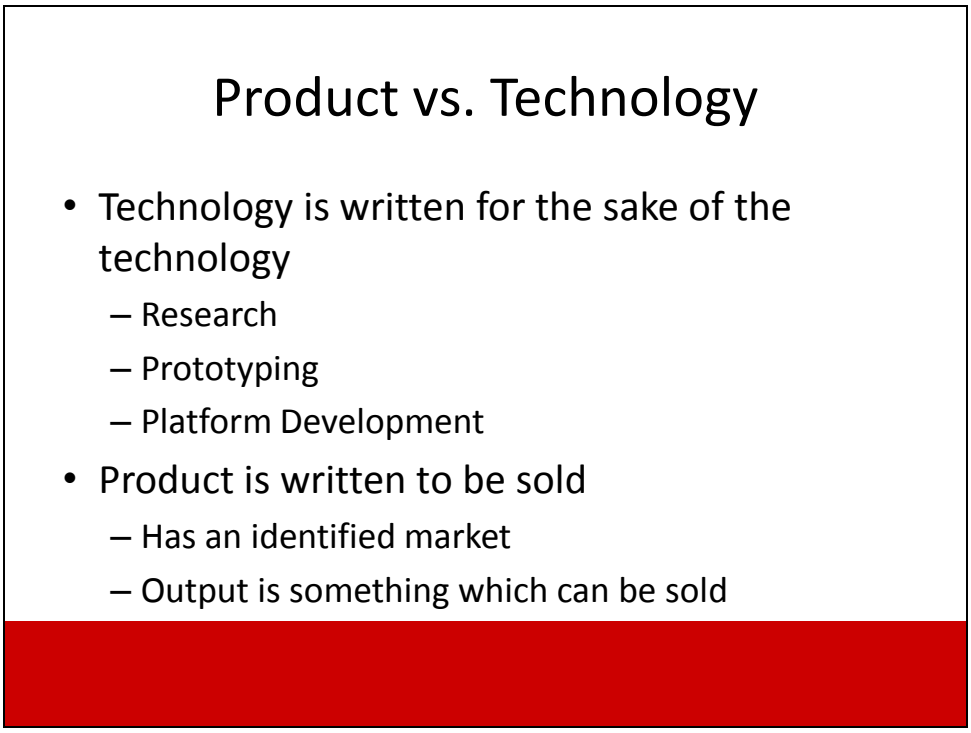
Introduction

Developing commercial software is expensive. It is not uncommon to spend several million dollars on a fairly small application. How can this be the case? There are often not huge technical challenges and unknowns. Where does the money go?

I am going to show you where some of this cost comes from and try and highlight some of the differences between creating technology and creating commercial software, and given the enormous costs, show you some of the tools and techniques we use to try and make sure we are creating the right product.

I have included a few examples from projects I have been closely involved in over the past few years

Product vs. Technology

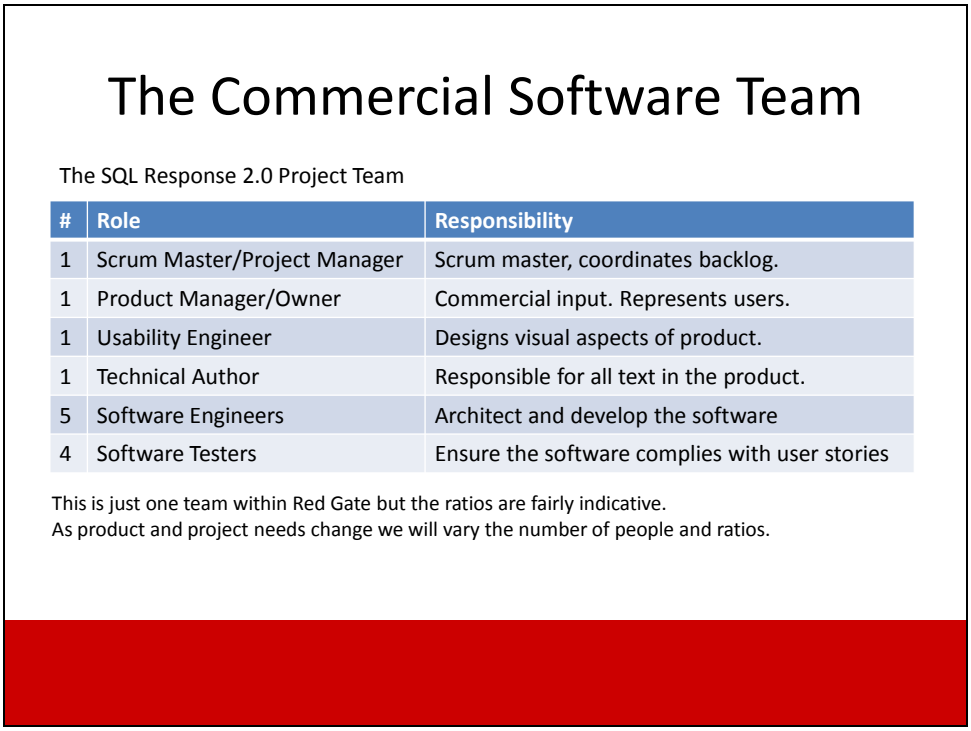


For me the difference between Technology and Product is the motivation for writing them:

Technology is written because it is interesting, cool, solves a problem in an innovative way and pushes our understanding of computer science further. The output of Industrial and Commercial R&D teams is technology. Some of this technology will turn into massively successful commercial software but this is often done as an afterthought or as a reaction to a highly successful piece of research.

Product is written to be sold. A potential set of buyers are identified and software is written with the sole purpose of selling to that target market. The output of commercial software teams is product which is ready to be purchased by a user. Some products will fail and some will be successful but as a commercial organization there is little or no intrinsic value in the software itself, the value is in the product.

The Commercial Software Development Team



The Commercial Software Team

The SQL Response 2.0 Project Team

#	Role	Responsibility
1	Scrum Master/Project Manager	Scrum master, coordinates backlog.
1	Product Manager/Owner	Commercial input. Represents users.
1	Usability Engineer	Designs visual aspects of product.
1	Technical Author	Responsible for all text in the product.
5	Software Engineers	Architect and develop the software
4	Software Testers	Ensure the software complies with user stories

This is just one team within Red Gate but the ratios are fairly indicative.
As product and project needs change we will vary the number of people and ratios.

At Red Gate only about 20% of our employees have Software Engineer in their job title. We only write software so what do the other 80% of people do? Well if we exclude Sales, Support, Marketing and Management then out of the remaining people only about 40% are software Engineers. The other 60% are made up of:

Product Managers/Product Owners: The product manager is the person who has responsibility of the commercial success of the product. A product manager will often be the constant over multiple releases of a product where as people in other roles (or even the whole team) might change between projects.

Project Managers: The person who is responsible for the overall health of a project. They might act as Scrum Master and will liaise with marketing and sales to ensure that they are aware of any delays or changes to the project.

Usability Engineers: The usability function are HCI & design experts. It is their job to ensure that people can understand and use the software we create. Our sales model is try and buy rather than us taking the CTO out to golf and dinner so when someone uses our software they must be successful the first time otherwise they will go back to Google and try one of our competitors instead.

Technical Authors: They are responsible for every piece of text in the application, the help file and support centre.

Test Engineers: It is hard to test your own code effectively. People make the same assumptions about a piece of code (which is why independently developed systems often exhibit the same failure cases and can be of limited value for redundant systems) and developers like to write new code rather than find problems with code they have already

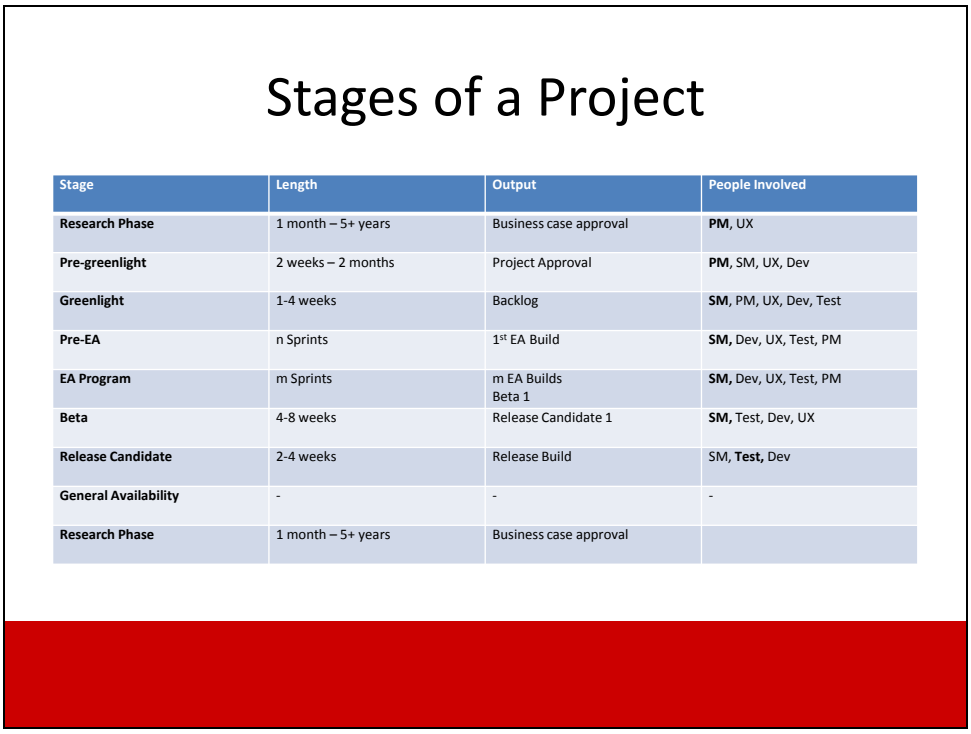
written so we hire test engineers who like breaking things to ensure our products stand up in as many different use cases and environments as we can (It is amazing how many text fields you can paste the entire text of war and peace into!).

One of the teams currently working for me is made up of the following people:

- 1 Product Manager/Product Owner
- 1 Project Manager
- 1 Usability Engineer
- 5 Software Engineers
- 4 Software Testers
- 1 Technical Author

Not all of our project teams are the same size but the ratio above is fairly typical.

Stages of a Project



Different functional groups become more or less involved in a project through its different stages. Different stages can last very different amounts of time. We try to have each stage have a defined output. By doing this the team can feel a sense of achievement. Also it allows a light weight mechanism of checking against the schedule.

Projects get late one day at a time

I have not come across a project where one day it was on time and the next day it was a year late. Having light weight mechanisms of tracking progress is vital unless you are happy to find out your project is 12 months late the day before you are due to release. To do this we have various gateways which a project must pass through before release. We try to

make these as lightweight and transparent as possible but you need to track the progress of your project carefully otherwise you will end up with a nasty shock.

When I first joined a commercial software company Product Management and Usability were the two roles mentioned above that I was least aware of and perhaps had the most impact on me as a software engineer. I am going to explore some of the problems they tackle and the tools they use to do this.

Product Management

The role of the Product Manager is to act as a conduit for all of the different sources of information about a product. The Product Manager is responsible for putting together business cases for projects. They often have P&L responsibility for a product (or product set). They do not sit in a room and think up the future roadmap for the tool. They go out to the market and listen to potential users. They talk to development teams, support teams, marketing teams and sales teams working on the project, they look at the economic trends and what the competitors do. In-fact they do everything they can to get a feel for the market and relay this back into the company.

The role of the Product Manager differs from that of a Product Marketing Manager (or Brand Manager) in that they Product Marketing Manager's focus is to tell the market about the features and benefits of your solutions. The Product Manager's role is to try and listen to the market and find the opportunities which the solution does not currently solve and which people would pay for.

There are many ways in which a Product Manager can interact with the Market, these will vary depending on your target market demographic and size.

The key ones we use at Red Gate are:

- Customer Visits

  It sounds very simple but go and visit your customers, or if you don't have any customers yet go and visit your potential customers! Watch them in their work place struggling with the problems they face every day. Try and understand how your product or solution can help them work more efficiently. Ask them open questions and listen. Try not to go in with preconceived ideas about what you will find.

- Attend Trade Shows

  Being based in Europe this can often be a little more time consuming and expensive than when based in the US but is well worth the effort. If you don't have the budget to exhibit then go as an attendee. If you go to the right show you will have a high concentration of potential customers who are relaxed and happy to talk. Most of your competitors will be there as well, go and introduce yourself, be friendly and civil and in my experience most of them are too! You never know when you might do business with them (partnership, acquisition or being acquired).

- Competitive Analysis

  What are your competitors doing? How are you perceived in the market in comparison to your competitors?

- Win/Loss Analysis

  If you have customers or potential customers who either didn't buy any solution or purchased competitors then find out why. If you had feature x would they have bought from you?

- Customer Feedback Forms

  Embed feedback mechanisms into your solution. Proactively request feedback and it will come.

- Support Requests

  What problems are your customers facing? If a customer is hitting this problem are potential customers hitting these problems and then deciding not to buy?

- Analyst Reports

  Gartner, Forrester and many others produce analyst reports for various different markets. Try and get your hands on these and read them! Be aware that people will often pay for analyst reports but they can give a good idea of trends within the market.

- Partner Customer research reports

  If you are working for a smaller software company they will often partner with a larger company. The larger company may well have an internal customer research team who produce reports. Try and get hold of them, they will help you understand your partner's strategy (and how you can fit into that) as well as highlighting potential opportunities you are better placed as a smaller and more agile company to exploit.

- Corporate Annual Statements

  Every public company must produce an annual report each year for the stock market (as well as making fillings throughout the year). These can be a real treasure-trove of information about your market.

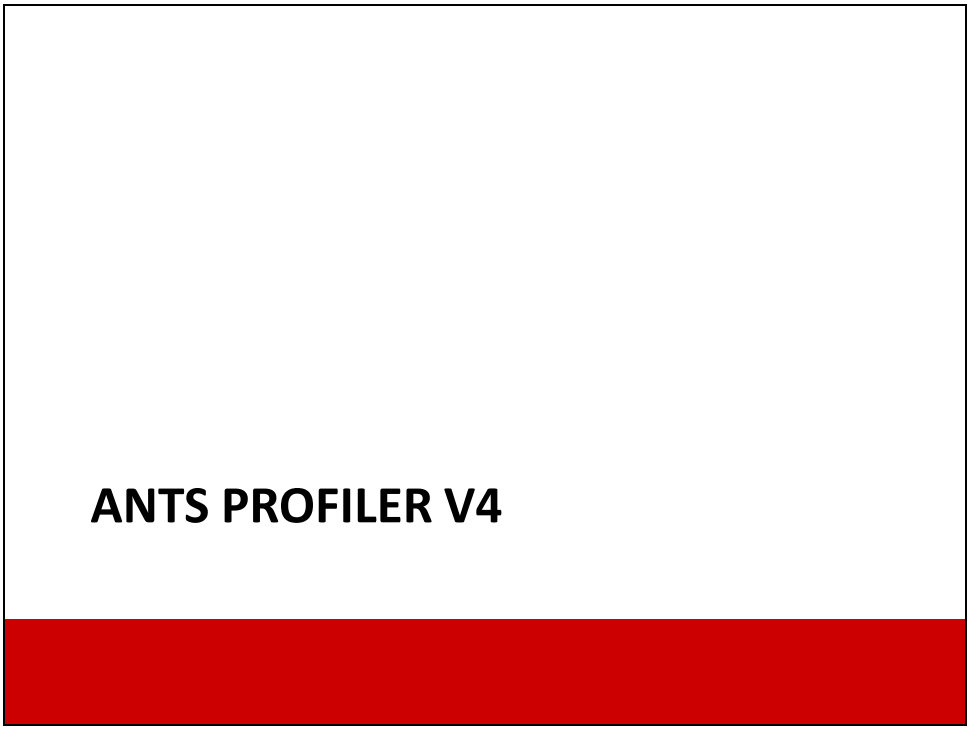
- Surveys

  We generally use these to confirm findings we have made via other mechanisms. By constructing a survey carefully and getting a reasonable number of your target market to reply to it you can ensure that the information you have found out via other means holds for the wider market.

The output of this research needs to be combined with Technical Innovation, other ideas from domain experts, the development team, usability team and support team to come up with a vision for the product.

Ultimately Market Research isn't a well defined path you walk along and come out with a single answer. There will be a lot of conflicting information which points in different directions. You will never get 100% knowledge and if you ever did you would probably have missed the market window.

Ants Profiler v4



In May 2007 we were struggling to grow our .NET Developer Tools business. Growth of our main profiling tool had stalled and the other players in the market were winning deals off us.

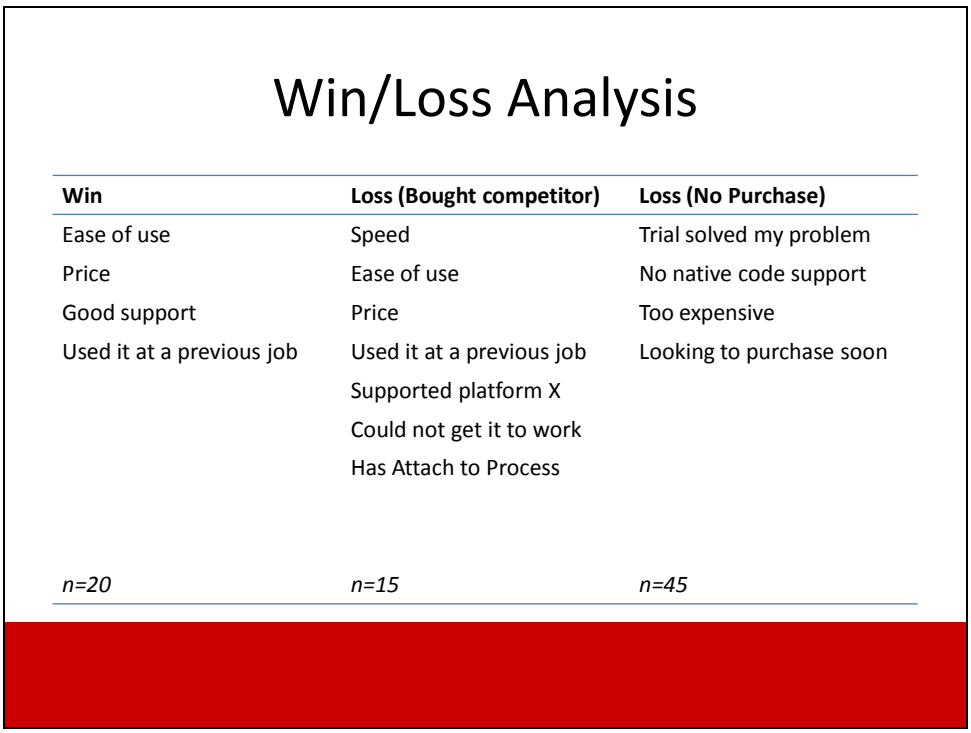
By putting a dedicated team on to the suite of tools and through Product Management, Technical Innovation and strong Usability we transformed the business from one in decline to one of the key growth areas for Red Gate. Over the past three years we have had a team working on the product set full time and each release of the product coincides with revenue growth.

Since May 2007 we have grown our Profiling tools business by 400% by applying some very simple principles and techniques to understanding the market.

Research Pipeline

The first step in researching a new tool, given a seed of an idea or an interest in the tool the first thing to do is to broaden your horizons as much as possible. Go out and search for adjacent or other ideas. Once you have a series of ideas you then need to begin to begin to collapse back down on a solution.

Win/Loss analysis



Win/Loss Analysis

Win	Loss (Bought competitor)	Loss (No Purchase)
Ease of use	Speed	Trial solved my problem
Price	Ease of use	No native code support
Good support	Price	Too expensive
Used it at a previous job	Used it at a previous job	Looking to purchase soon
	Supported platform X	
	Could not get it to work	
	Has Attach to Process	
n=20	*n=15*	*n=45*

One of the most important (and simplest) pieces of research you can do is to understand why your customers buy your product and why those who looked at your product decided to buy a competitors (if people are not considering your product in the first place then you need to solve that problem first).

We phoned them up and spoke to about 100 of them and asked them a very few simple (open) questions:

- What motivated you to evaluate our tool?
- Did our tool match your expectations?
- Did you evaluate any other tools?
- Which tool did you choose?
    - Why did you choose Tool X?

The output of this will be a (hopefully large) number of interview forms. From these we categorized customers into Win, Loss (Bought competitor) and Loss (No purchase). For each group we then looked for themes which were mentioned and prioritized each theme by how often it was mentioned.

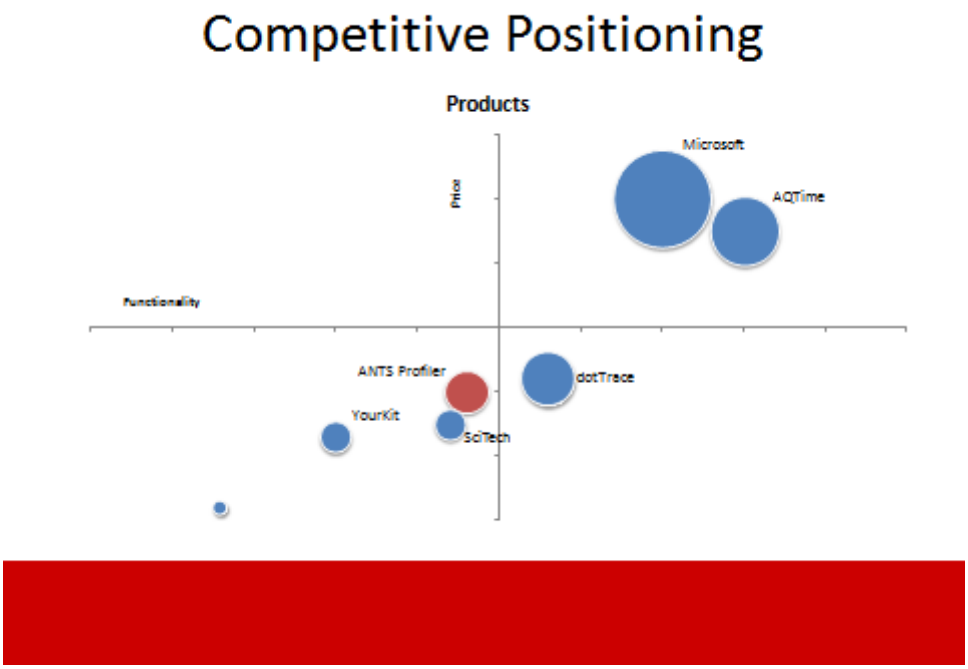
Competitive Positioning & Market Segmentation

The other key area to look at is at what your competitors are doing. You have to remember that what they have today is not likely to be the same as what they will be doing in 6-18 months time. We use many different mechanisms to find out information about our competitors and their plans. Look at any public support forums they have, talk to their customers, listen to their earnings calls and buy their people lots of beer at trade shows.

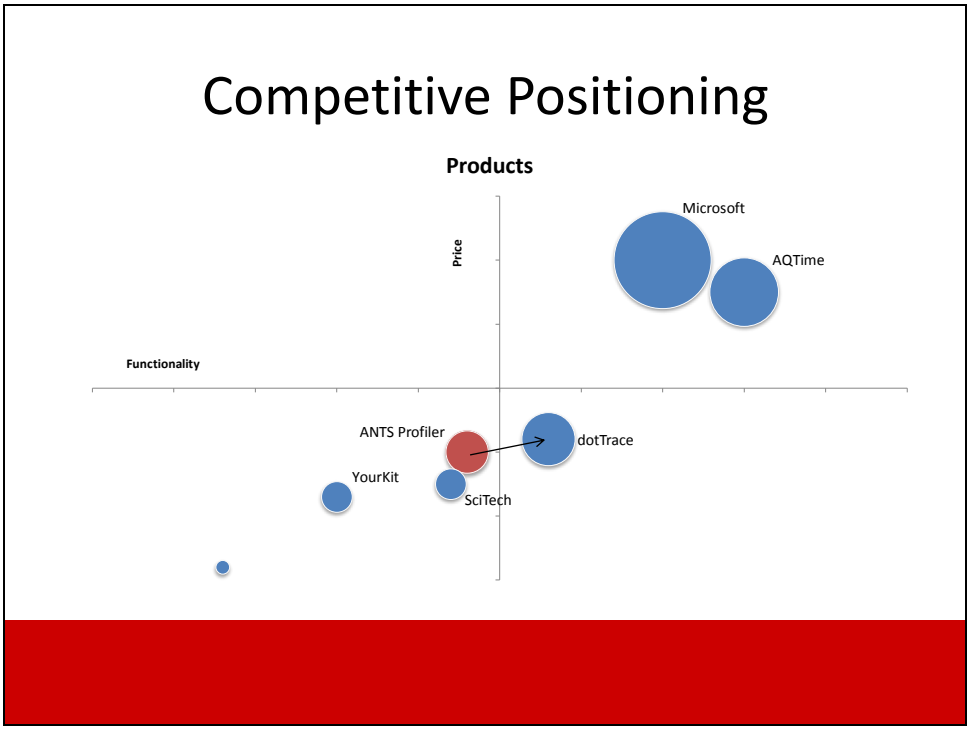
You need to try and figure out what their strategy is. Are they targeting particular verticals? What new features are they likely to add in new versions? Are they going to change their pricing strategy? How do they react to your new version? How can they compete with it, will they drop their price and if they do are they extra features you have comprehensive enough to justify a pricing premium? Is your market price sensitive?

We will often use a 2x2 matrix to represent our findings from this background research. Plotting two independent variables (often "functionality" and "price"). By doing this you can see the overall makeup of your market and decide where and how to compete.

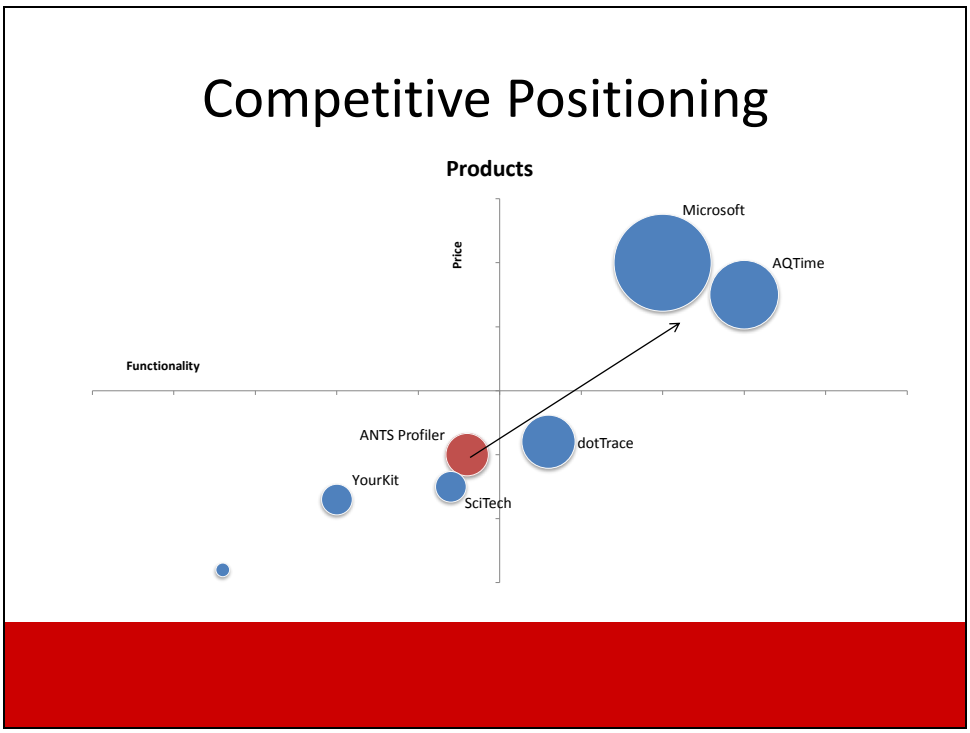
Below is the relative positioning for the .NET performance profiling market in May 2007:



Where should we aim to be after version 4? Our biggest competitor was dotTrace. They had more functionality than us at pretty much the same price point. But they were in the low price, high functionality bracket. We can't compete with Microsoft's total solution and AQTime had a very broad offering which covered Java, .NET and native code. The first temptation is to try and compete head on with dotTrace. Match their functionality and price and we will get a greater share of the market.



The problem here is that this is a snapshot of the market today. Where are we going to be in 12 months time? We couldn't be sure what dotTrace were going to be doing so our strategy was to make a much more aggressive leap:

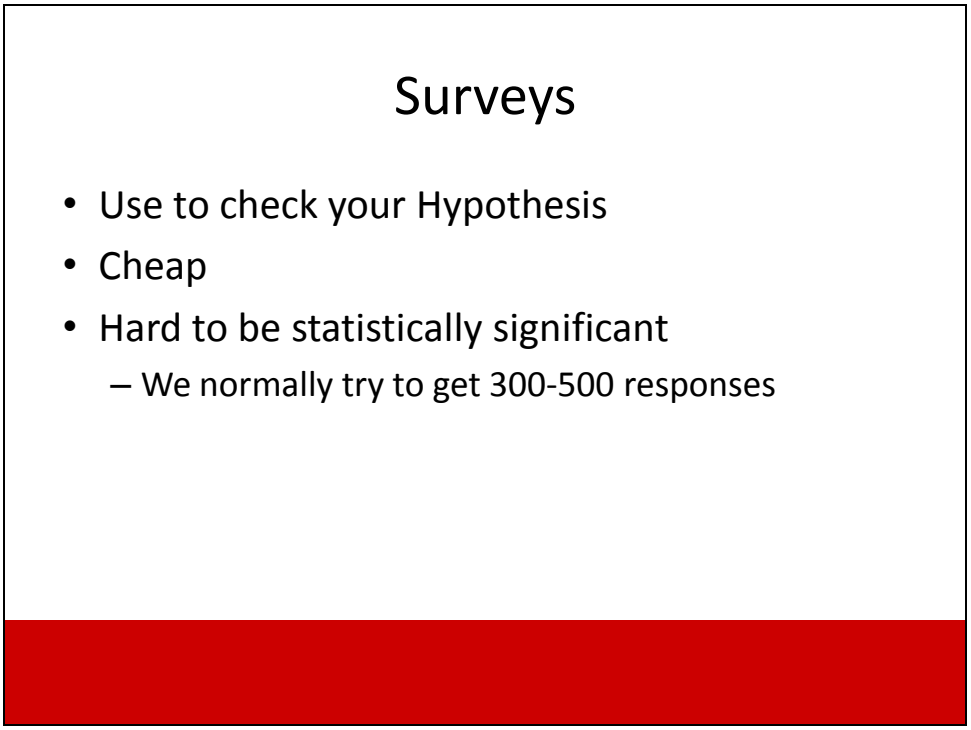


Now it would have cost far too much to compete with AQTime and Microsoft across the broad offering they have so we segmented the market and did a better job of solving the problems that a particular niche had and only compete in that niche (for us this niche was

.NET Developers). Also if dotTrace began to compete on functionality then we could drop our pricing if needed.

Alongside the development of the Profilers we wanted to embed our position in the market as the first name someone thinks of when they go to Google looking for a profiler. To do this we needed to increase the awareness of our brand significantly - this can be a very expensive thing to do (who really cares about .NET Profilers, almost no one unless they have a performance problem) so we decided to buy a widely used free tool (.NET Reflector) and use its brand to increase the presence of our own.

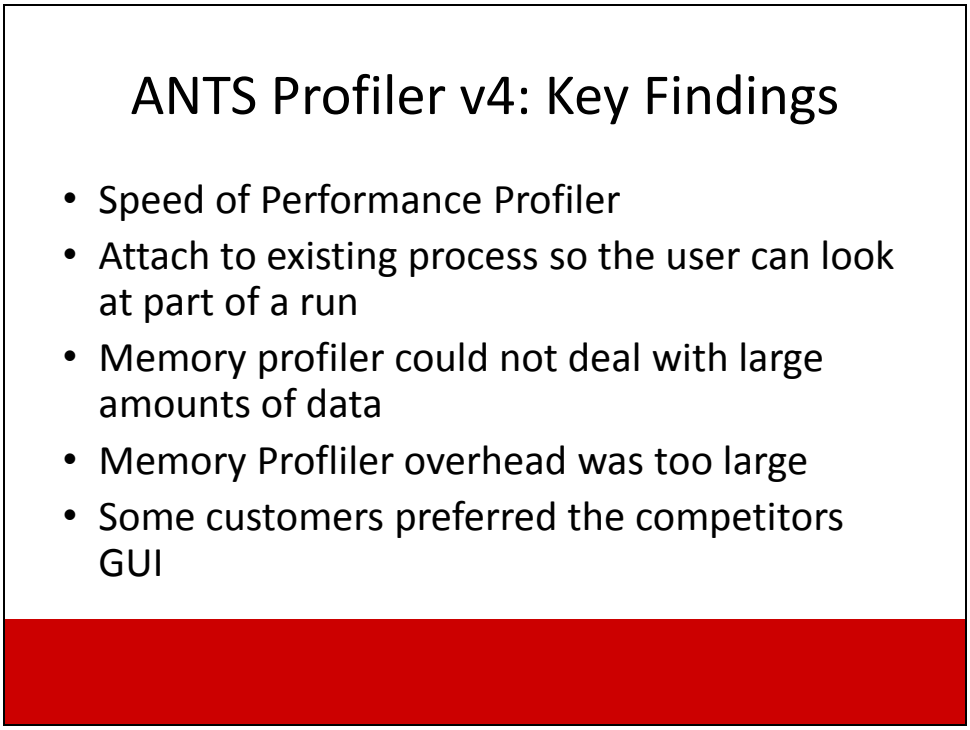
Surveys



Once you have a hypothesis about the product you want to build you need to double check your findings. It is normally too expensive to have direct contact with are large amount of your customer base and to double check your findings that way. The method we normally use is to run a small competition for people who are willing to fill in the survey.

Although it would be nice to be statistically through with your survey and ensure that you end up with a statistically significant proportion of your customer base it is often impossible to reach that many people. I will generally look to try and get 300-500 responses to a survey. Although not always perfect it is often enough to verify your earlier findings.

Key findings



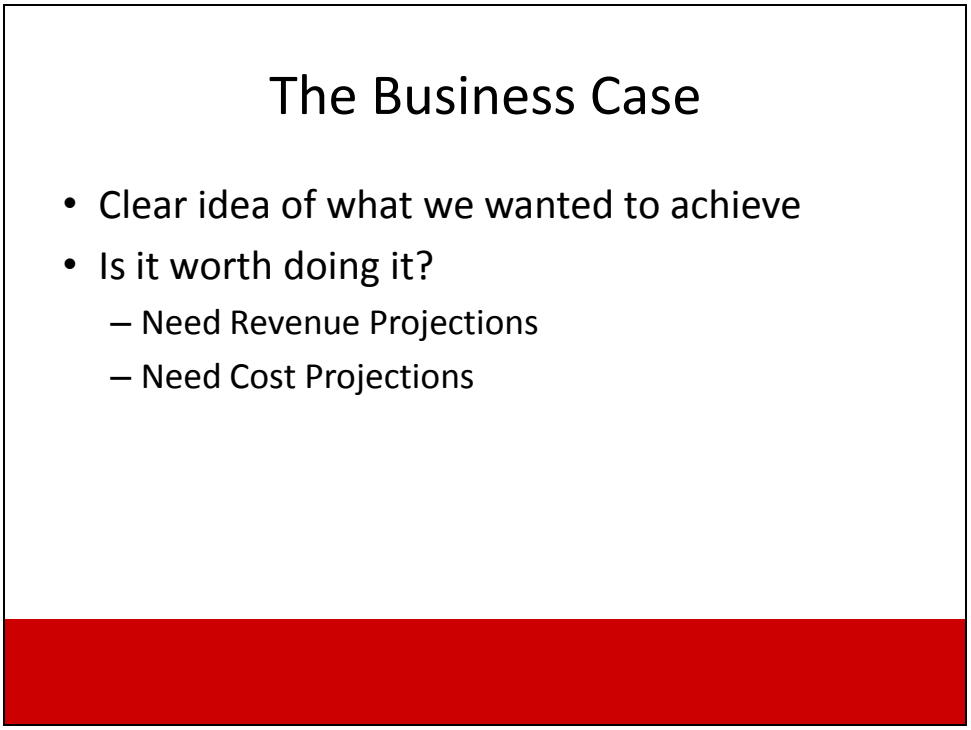
Our research found several key areas where we were lacking:

- Speed of Profiling was a big problem
- Attaching to an existing process was important as when profiling many people were interested in only part of their program's run.
- Memory Profiling could not deal with large amounts of data
- The Memory Profiler engine's overhead was too large and would often make a memory problem more severe limiting the usefulness of the profiler.
- Ease of use was OK but some people found one of our competitor's easier to use.

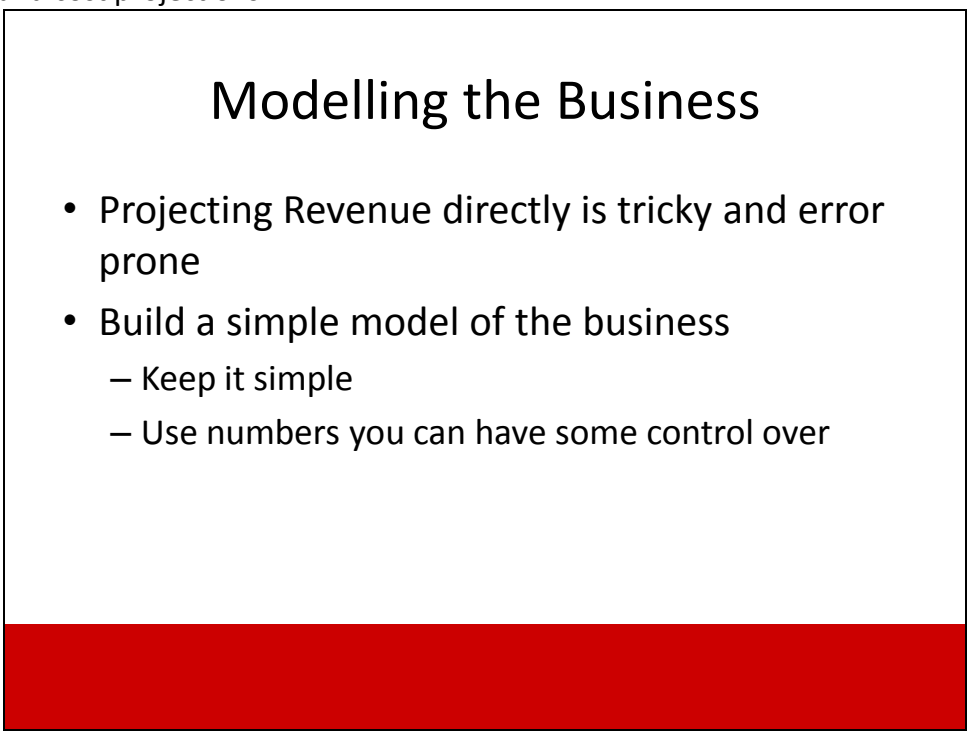
The strategy was to overhaul our performance profiler with a focus on improving the speed and ease of use. We decided not to do any work to the memory profiler in the short term other than to split the product into two (So we would have ANTS Performance Profiler and ANTS Memory Profiler). We did not have the resource to do both products at the same time.

We were then going to use our findings from the Performance Profiler to influence the Memory Profiler's design and try and solve the key problem of the size of data.

The Business Case



We had a clear idea of what we wanted to achieve in terms of end user experience (although we didn't really know what it would look like or our ability to deliver it without doing significant technical work). Before committing a project team to the project we need to ensure that the project is worth doing. To do this we will build a model of the project revenue and cost projections.



Projecting revenue directly is incredibly difficult and error prone. You first need to build a model of the business which relates to the leavers you have. In our case the model we used was fairly simple:

Modelling the Business

- Downloads: Number of people trying out the tool
- Conversion Ratio: The percentage of these people who purchase the tool
- Average Transaction Value: The average spend of a customer

Revenue = Conversion Ratio x Downloads x Average Transaction Value

Average Transaction Value $\propto$ *Price*

$$Revenue = Conversion\ Ratio \times Downloads \times Average\ Transaction\ Value$$

Past experience form our other tools had also shown that

$$Average\ Transaction\ Value \propto Price$$

From this I could make projections about the amount of interest we would have in the tool (Downloads), changes in the Average Transaction Value (based on changes in price) and the conversion ratio (The % of downloads which purchase the tool).

Modelling the Business

- *Use simple cost models unless something more sophisticated is needed*

$$Monthly\ Project\ Cost = Number\ of\ People \times 23 \times \$1{,}000$$

$$Cost\ of\ Project = Monthly\ Project\ Cost \times Number\ of\ Project\ Months$$

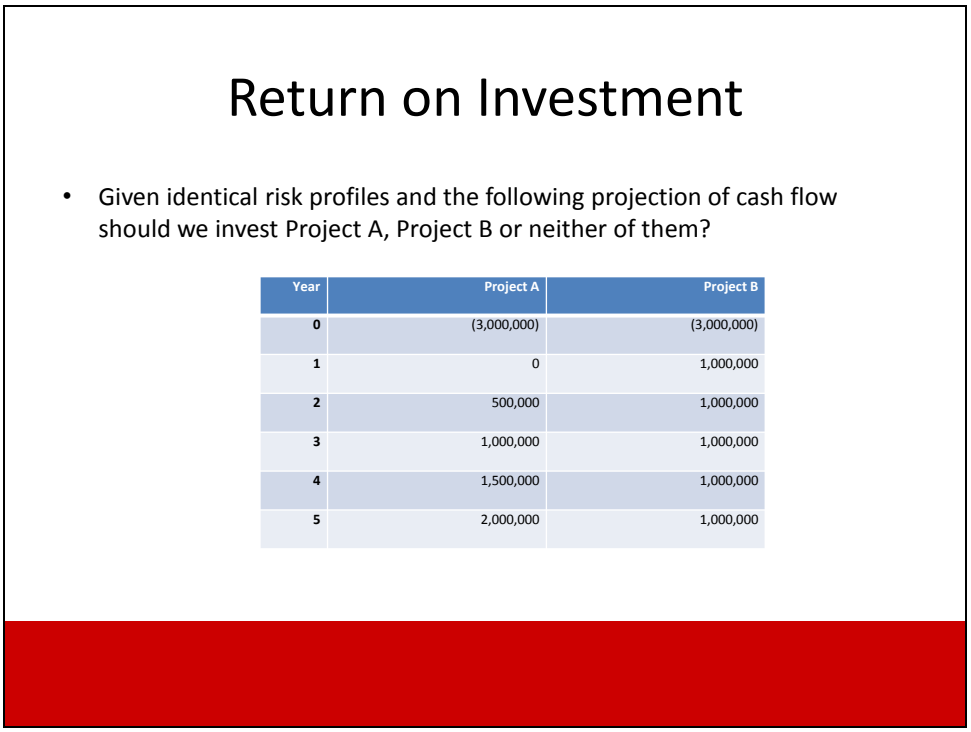
For the cost of development we also took a very simple model:

$$Monthly\ Project\ Cost = Number\ of\ People\ \times 23\ \times \$1{,}000$$

$$Cost\ of\ Project = Monthly\ Project\ Cost\ \times Number\ of\ Project\ Months$$

The aim was to ensure that we overestimated the cost of the project. If we did this and it still made sense to do the project we knew we didn't need to spend too much more time doing project modeling and we just needed to ensure we were always within our worst case scenario.

Return on Investment

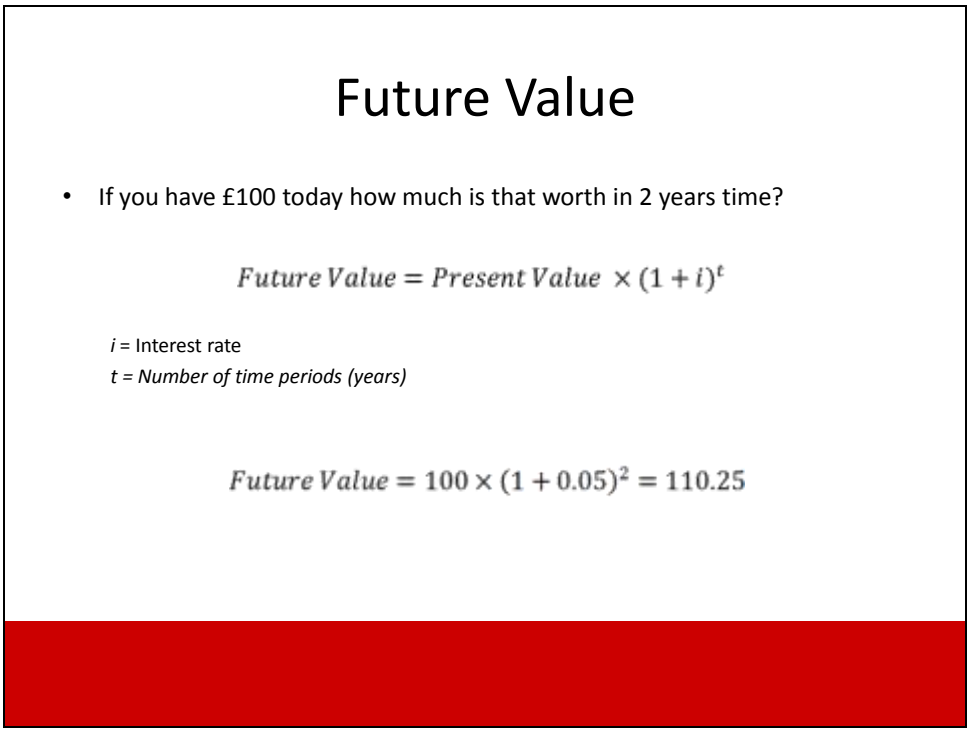


Given identical risk profiles and the following projection of cash flow should we invest Project A, Project B or neither of them?

Given the following cash flow projections for the two projects how can you chose between them? There are a few fairly simple mechanisms we use to choose.

Year	Project A	Project B
1	(3,000,000)	(3,000,000)
2	0	1,000,000
3	500,000	1,000,000
4	1,000,000	1,000,000
5	1,500,000	1,000,000
6	2,000,000	1,000,000

Future Value



The mechanism we use is fairly simple. We use various calculations based on the future value of money. If you have £100 today how much is that worth in 2 years time? Given a nominal compound interest rate of 5%pa we can calculate the value of £100 in 2 years using the following formula:

$$Future\ Value = Present\ Value\ \times (1+i)^t$$

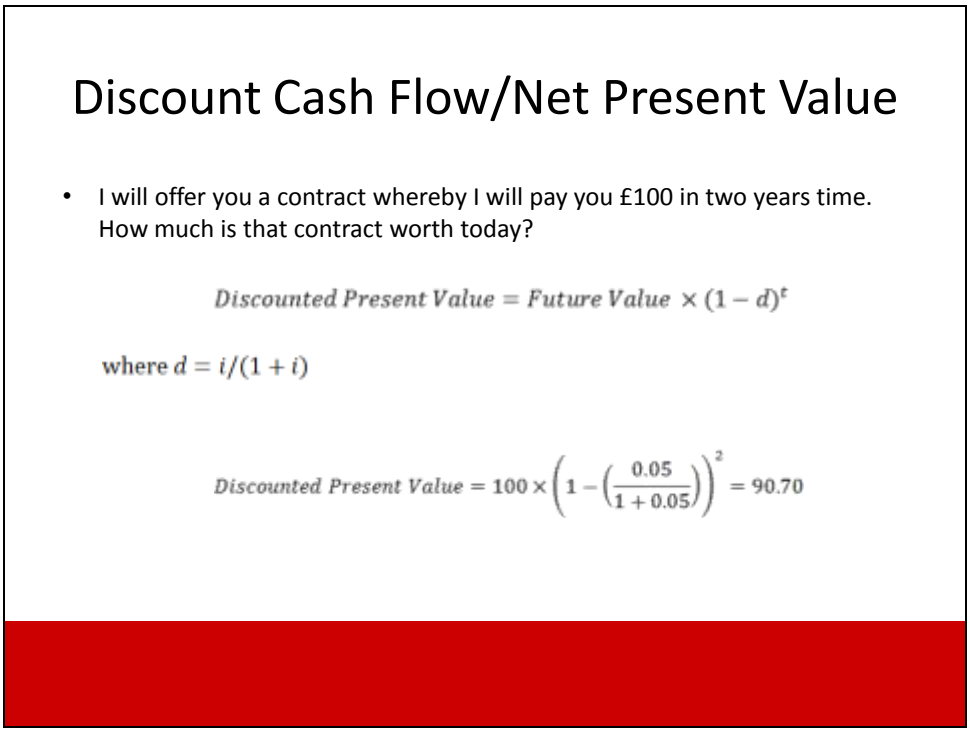
where *i* is the interest rate per period of time t.

So:

$$Future\ Value = 100 \times (1+0.05)^2 = 110.25$$

Note that by using a compounded interest payments then we end up with exponential growth.

Discounted Cash Flow/Net Present Value



I will offer you a contract whereby I will pay you £100 in two years time. How much is that contract worth today?

Well we know it's not going to be worth £100. I am going to have to compensate you for the delay in payment. But how much? We call this discount the Discounted Present Value (or the Future Value adjusted for the delay in receipt).

Using the Future Value formula above we derive that:

$$Discounted\ Present\ Value = Future\ Value\ \times (1-d)^t$$

Where $d = i/(1+i)$

So our contract for £100 with a 5% compounded interest is worth:

$$Discounted\ Present\ Value = 100 \times \left(1 - \left(\frac{0.05}{1+0.05}\right)\right)^2 = 90.70$$

Discount Cash Flow/Net Present Value

- For a cash flow:

$$Net\ Present\ Value = \sum_{t=1}^{N} \frac{C_t}{(1+i)^t}$$

i	Project A	Project B
5%	1,065,198	1,266,168
10%	391,728	718,897)

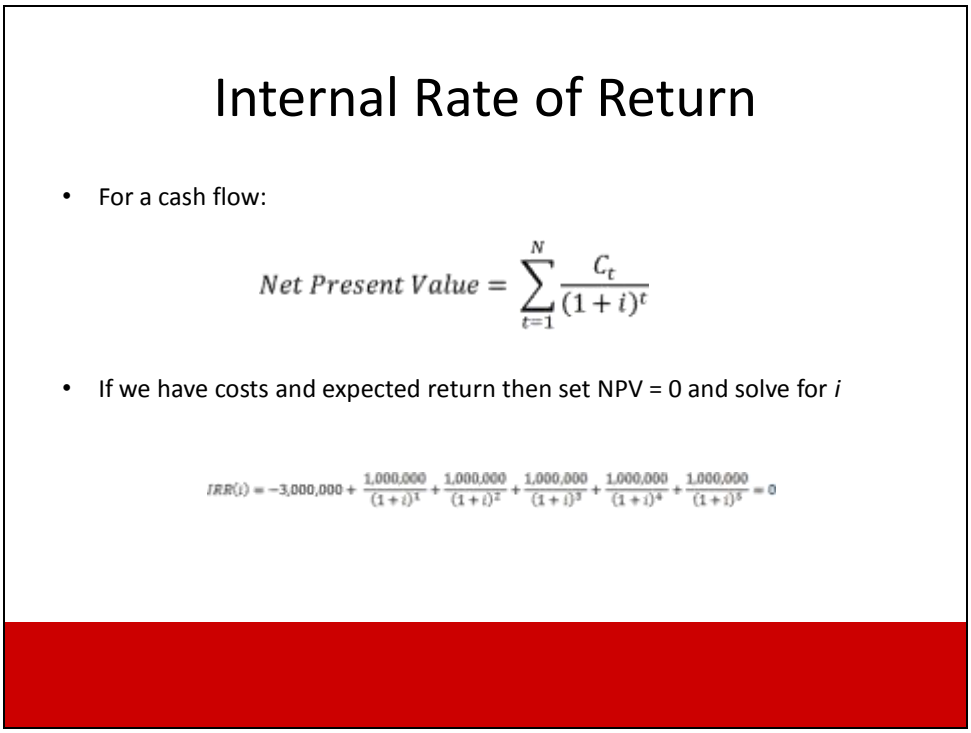
Hopefully cash flows won't be a single one of payment but a long term income. So how much is it worth you paying me today if I am going to pay you £100/year for the next 6 years?

$$Net\ Present\ Value = \sum_{t=1}^{N} \frac{C_t}{(1+i)^t}$$

We often refer to this as the Net Present Value. IE the value today of a future cash flow.

By computing the DPV for the two projects we can see that Project B is the better value project (DPV ~$1.2M). But is the project worth doing at all (given it's risk profile) or are we better leaving the money in the bank?

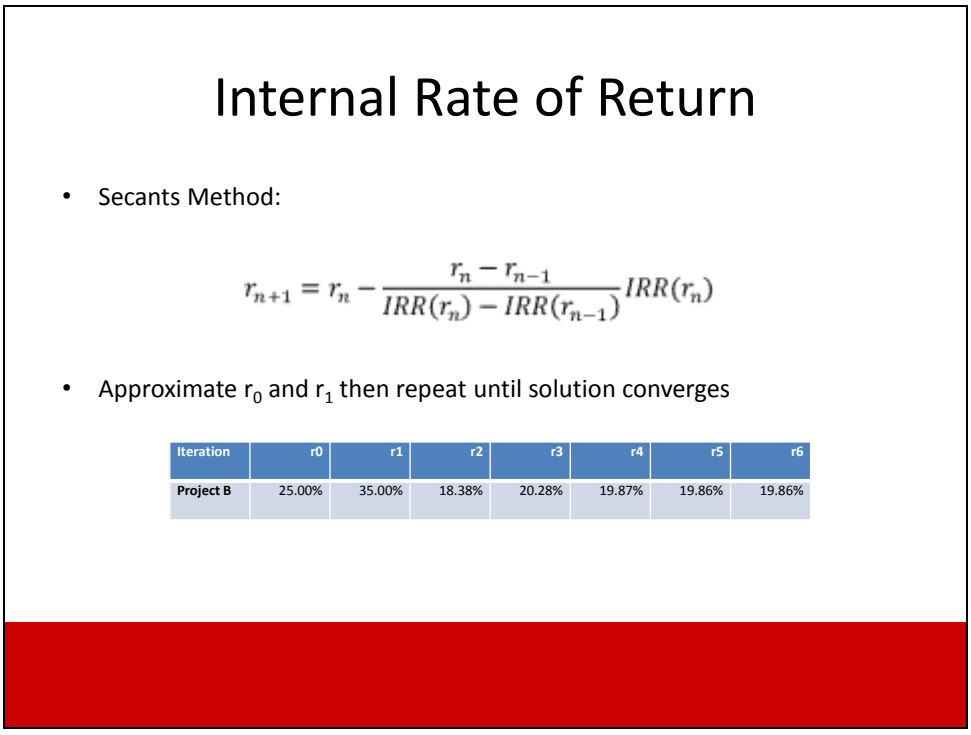
Internal Rate of Return



To answer this question we want to understand what the effective rate of return on a project is. We can do this (given we know the cost of investment up front) by finding the roots of the above equation for *i*.

Project B:

$$IRR(i) = -3{,}000{,}000 + \frac{1{,}000{,}000}{(1+i)^1} + \frac{1{,}000{,}000}{(1+i)^2} + \frac{1{,}000{,}000}{(1+i)^3} + \frac{1{,}000{,}000}{(1+i)^4} + \frac{1{,}000{,}000}{(1+i)^5} = 0$$



We can find the roots of the equating using the Secant's method:

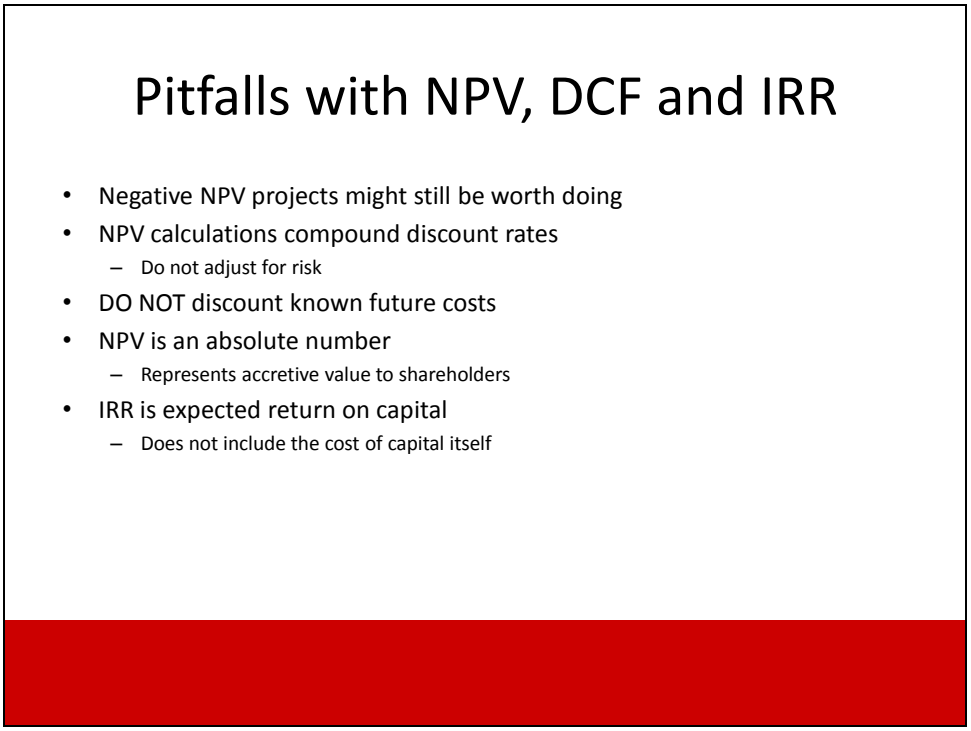
$$r_{n+1} = r_n - \frac{r_n - r_{n-1}}{IRR(r_n) - IRR(r_{n-1})} IRR(r_n)$$

We make an approximation of r_0 and r_1 then iterate until we converge on a root of the equation.

Iteration	r0	r1	r2	r3	r4	r5	r6
Project B	25.00%	35.00%	18.38%	20.28%	19.87%	19.86%	19.86%

So in this case an internal rate of return of 19.86%, taking into account the risk profile, might compensate us enough to make the investment.

Pitfalls with NPV, DCF and IRR



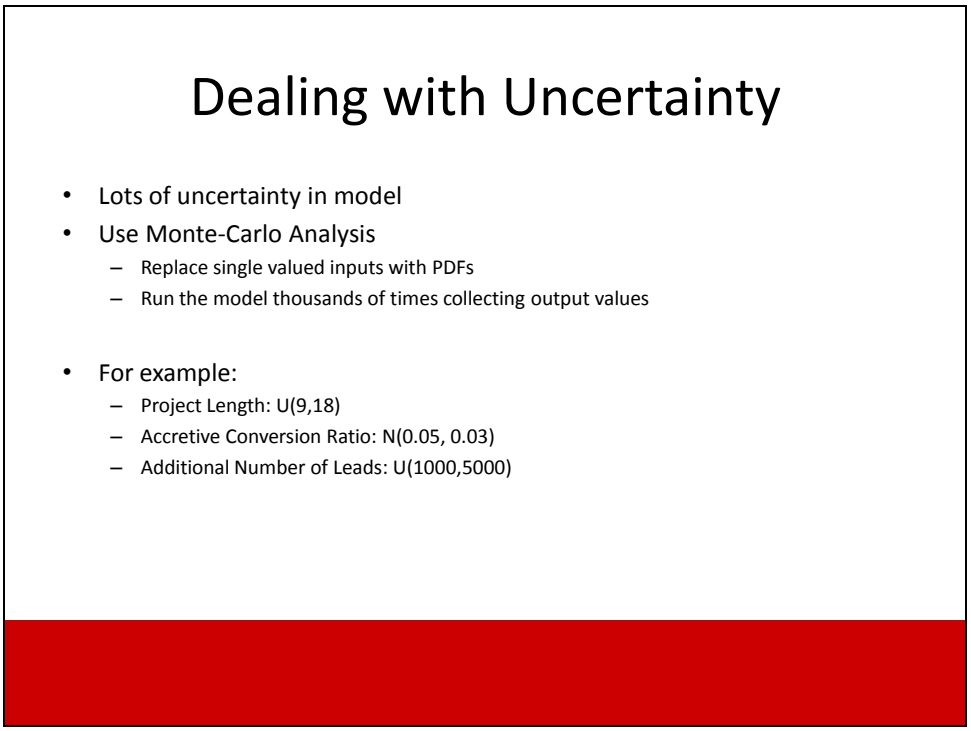
Although a project may have a negative NPV it may still be worth doing. This is because the world does not remain constant and if you do not invest in your tools and your competitors do or a better solution comes along you can lose more money by not doing anything than by kicking off a loss making profit. Equally if you do not spend today then you may well have to spend more in the future to continue to compete in the market.

Remember that NPV calculations compound discount rates, this means that you should not make the discount rate higher to take into account additional risk otherwise you are compounding the risk year on year. We would normally use a tool such as Monte Carlo modeling to try and take the different risks into account.

If there are some costs which occur later in the project which are committed to now then try to figure out the financing costs of these debts today and put them into your calculation earlier.

NPV is an absolute number and thus will often vary with the cost of the project (a larger NPV might solely be a function of a lower rate of return on a higher initial investment). IRR reflects the expected return on capital but does not include the cost of capital itself so comparison of two projects just using IRR is not a good idea either.

Dealing with Uncertainty

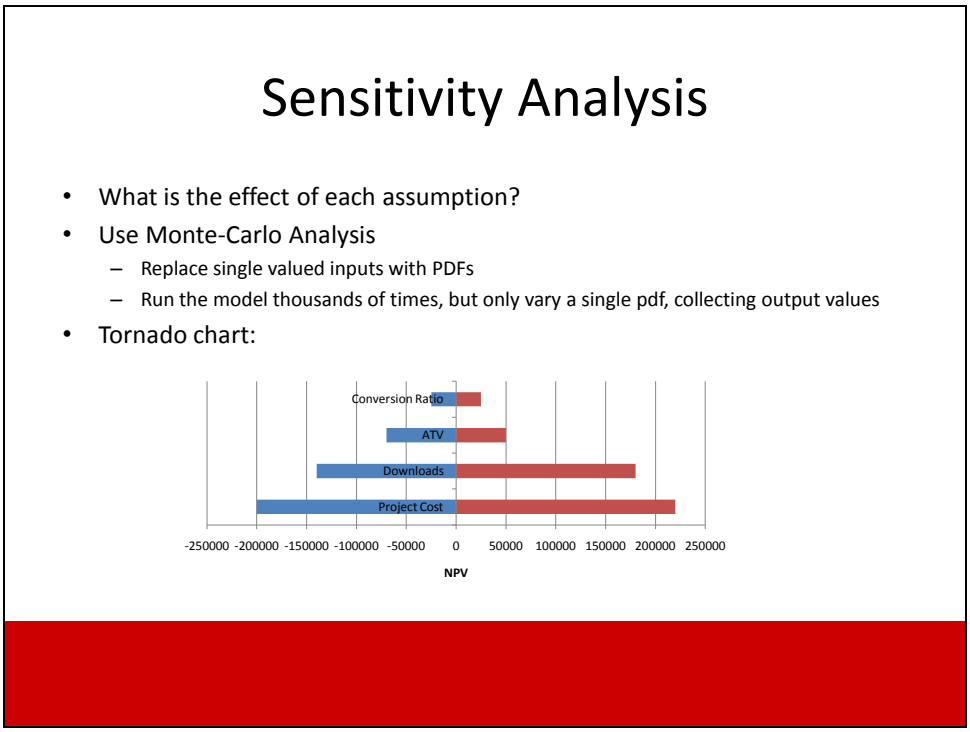


Unfortunately it is very rare that we have such clarity over our investment and return. Our project estimates may be out by 25% and we might have overestimated the future cash flow we will get in return for running the project. We still need to be able to compare projects in the pipeline and choose the best with some level of certainty.

To do this we use a method call Monte-Carlo analysis. Rather than having single values as an input to our model we use PDFs. For example the length of a project (in months) may be modeled by a uniform distribution $U(9,18)$. We then may model the accretive conversion rate by the normal distribution $N(0.05, 0.03)$ and the increased number of leads by another uniform distribution $U(1000,5000)$.

With these three inputs we then run our model thousands of times by sampling the inputs, creating the model and recording the outputs. This allows us to answer some interesting questions: if the project overruns and does not perform as well as we had hoped would we still make money? What is the likelihood of us not making money?

Sensitivity Analysis



Another form of analysis we perform is sensitivity analysis: *Is our project model unduly affected by the variance of a single input?* To do this we do the above but only vary a single PDF and record the output again. This is often useful as if a project's profitability is very sensitive to the variance of a single PDF then you need to ensure that the underlying assumptions in the model are not over or under pessimistic.

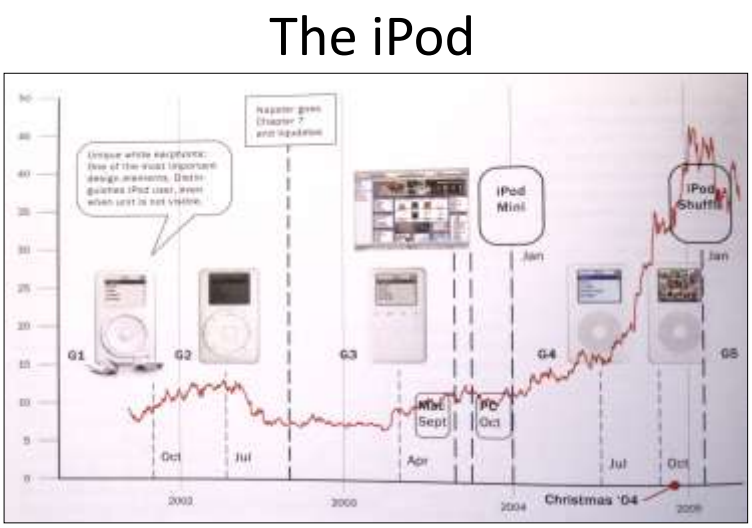
Usability

Our usability team is one of the most important teams within Red Gate. Ease of use is all the rage these days but it does not happen by magic or by having some genius on your team. It,

as with everything else related to software, takes a lot of hard work, research and perseverance to achieve a high quality user experience.

A brief aside to show that it is not as easy as putting some nice gradients on your application..

The iPod



Bill Buxton, Sketching User Experiences, p48, 2007

Bill Buxton, in his excellent book "Sketching User Experiences" discusses the iPod in some depth. It is perhaps the largest success of usability in the market place in recent years (along with the iPhone). For me it is a great illustration of how so many different things have to come together for usability to be a real competitive advantage.

The graph above illustrates that the iPod was anything but an overnight success. It took three years and continued, significant investment before the iPod really took off. The first signs of this are in October 2003 when the iTunes Store launches for the PC. But it wasn't until the 4th Generation iPod that sales really took off.

The iPod was not the first MP3 player. There had been many others launched previously, some had moderate success but none of them come close to the iPod in terms of sales. Apple, through the use of branding, styling, usability and the provision of a total solution turned the market from a niche market into a massive consumer market.

Apple got a jump on the market but they did not sit back as market leaders. They stayed ahead of the rest of the market by competing with themselves and continually innovating. Until Apple make a mistake it is going to be very hard for anyone else to catch them up and compete effectively.

Apple have the best designed product but it's still not perfect by any means. There are many short comings and possible improvements but it is better than the competitors and that is what matters. Design, after a certain point, is relative not absolute.

Apple continued to bring out new versions of the iPod on a very aggressive schedule. Why did they do this? The 1st generation iPod looks very similar to the 4th generation one and even with the new nanos and shuffles you can see a family resemblance. But Apple did not just improve functionality with each generation, they changed the feel of the product significantly. If you play with a 1st gen ipod today it feels clunky and unrefined, yet when it came out it was a revelation.

The iPod itself was not a success but required iTunes and an incredible marketing campaign to be a success. It is the total solution that people are buying, not just the iPod. If I show you a silhouette on a colored background it shouts iPod regardless of what that silhouette is..

Steve Jobs had no idea how successful the iPod would be. He hoped it would be a success but did not know. Apple persevered with the product, continued to innovate and solve problems until it became a success but that might never have happened. They got somewhat lucky with the growth of the internet and the acceptance by the consumer market of this sort of product but if you continue to innovate and push the boundaries backed up with strong research you will get some hits.

Improve Ease of Use

One of the main aims of the project was to solve the speed issues with our current profiling engine. We had come up with an initial architecture which we felt would solve this problem. I was concerned that we would not push ourselves far enough though. I wanted to get the team to come to the project with no preconceptions at all about how we were going to present the information.

To do this I took some fairly radical steps.. I came in very early one morning and removed all of the computers from the development team's desks and replace them with pencils and sketch books. So often we all get excited by the possibilities of what we could do that coding would start of the first day of the project. People would try things out and see how they worked.

I wanted to get the team to explore a wide range of possibilities very quickly and there is nothing like a pencil when doing this. The other great thing about sketching is that they are rough, unfinished and people feel happy criticizing them. Take someone a beautifully rendered screen shot and they will look at the style rather than the substance. Show someone a rough sketch and they will talk about the concepts and if something isn't clear you can both grab a pencil and start drawing. I was not sure what to expect when we took all of the computers away but I was blown away by the results we got.

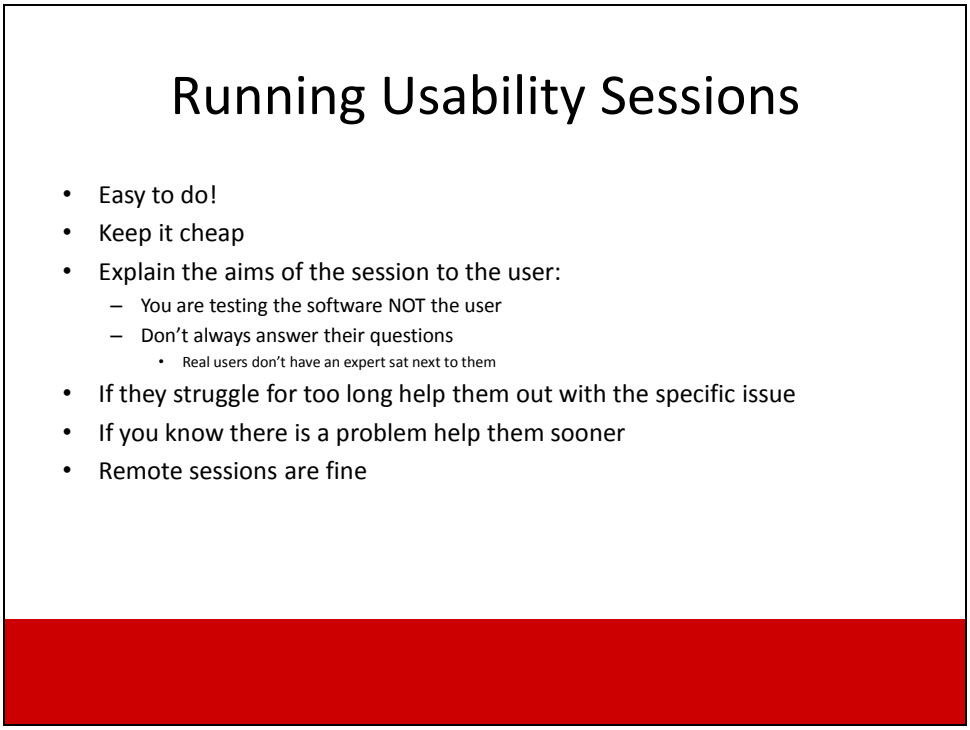
In about a week we had explored hundreds of concepts and had come up with some great ideas which ended up looking remarkably similar to the final release. Best of all where as previously we often ended up changing designs radically later on in the project (which costs a lot of time and effort) because we had a clear coherent, joint vision of the end result from early on the project was completed in record time for a project of this size.

After the initial concepts had been developed we created some nicer renderings of the tool and took these out to customers to test. We would have a stack of screen shots of the different parts of the application and ask the customer to pretend they were using a computer and swap in and out the right page depending on the action they took. This allowed us to test the designs early on and make changes to the concepts and layout of the different parts of the application before we had spent significant amounts of money getting it to a point where they could use it directly.

One of the other key factors in the success of the project was large amounts of feedback early on. The moment we had a build which could be described as semi-functional we released it as an early access build. By doing this regularly we ensured that we had stable builds at all times (great for agile methodologies) and we could have very fast feedback loops through the beta testers we had and through the usability sessions we ran.

Over the course of the project (8 months) we ran about 60 usability sessions. These were a mix of existing profiling users but new to ANTS Profiler, existing customers, people who had never profiled code before and there were several people we ran regular usability sessions with so they could react to the changes we made.

Running usability sessions



Running a usability session is not a hard thing to do and will almost always give you some great feedback about your tool. If it is too expensive or difficult to reach your actual users then substitute them for someone you can reach. Nearly anyone will do (as long as they don't have an emotional attachment to the project). Someone not quite right is still better than not running them!

There are plenty of companies out there that will charge you a lot of money to do eye tracking etc. Unless you have a real need to use them don't bother - keep the cost of the sessions down and do more of them! We use a laptop and microphone when running them. If it is a remote user session we have a conference phone and use webex or similar software for remote control and a screen recorder such as Camtasia.

At the start of the session explain to the user what the aim of the session is. Make it clear that you are not testing them but the software. They might get stuck and have problems, if this happens then they shouldn't worry, they are showing you an area of the software you can improve. Ask them to think out loud so you can understand their thought process.

Finally, make it clear to them that if they ask you a question you might not answer it as you want to see how they progress without external help. For me this is always the hardest part of a usability session, when do you help the user and when do you let them struggle. My advice here is when they hit a problem let them struggle for a minute or two, if they don't solve the problem themselves then make a note and tell them how to achieve what they are trying to do and be quiet again. If you are running a series of user sessions and people keep hitting the same problem then we often prompt them after 20-30 seconds explaining that we know this area is a problem already.

Some people are very anti remote usability sessions saying you need to watch the candidate and understand their frustrations. In my experience a face to face usability session is the best but can be costly to arrange and remote usability sessions deliver nearly as much value so you are much better spending your time doing a few remote usability session than spending time and money travelling miles.