"The Robert Plank Show"

Episode #013

How to Create and Sell Software on the Internet

Facebook: www.RobertPlankShow.com Blog: www.RobertPlank.com

Contents

The nerds have won! Two out of the top three richest people in the US have one thing in common: they built their massive wealth by creating software. Hundreds of software multimillionaires have built their software empires by following the same pattern, over and over. They develop a hot idea for software, transform these ideas into best-selling software and then, rake in truck-loads of cash by offering it to millions. Bet you're asking: "What does that have to do with me?" If you dream of building a business that can stand the test of time, that can deliver consistent massive profits, and doesn't take a genius to create and maintain, then your dream can now be a reality.

Others want you to think that developing software is too hard or too expensive, they want you to believe that software development is only for the elite, the super-smart, or the super-powerful. Today you're in luck. Because we have internet marketer and software developer Robert Plank on the line. And if there's one thing he knows, it's software.

The secret to creating massive software profits boils down to three simple steps.

Step one: get a killer software idea. Companies spend millions of dollars engineering processes to generate software ideas. What a joke. All you need is a system that consistently delivers hundreds of software ideas to your doorstep, day after day.

Step two: Build a software using a time-tested, proven blueprint. You've seen it, software that just plain sucks. It sucks because someone forgot to use a time-tested, proven system for designing their software. They get lazy. And guess what: you pay for it. All you need is a system that gives you the step-by-step blueprint that delivers great software every time.

Step three: make the software available. Every year, software companies like Microsoft, Google and Apple spend millions of dollars promoting and marketing their software. What? Are they crazy? Yes. All you need is a system that delivers proven strategies to generate massive low cost and free exposure to your software and techniques that get people to buy right now.

That's it. Three steps stand between you today and a software empire that could have your name on it tomorrow. The formula doesn't require you to be a software programmer, in fact, I hope you've never seen a line of code. In the next few minutes you could have, in your own two hands, a step-by step formula for turning software ideas into software gold. So close down all distractions and get ready to welcome our keynote speaker, Robert Plank.

Great Even If You're Not a Programmer!

I really want to stress the benefits of outsourcing and getting your own software created, especially if you're not a programmer. That is the number one resistance point I get when I have any kind of software training at all. Where people say, "You know what? I'm not even going to worry about software, I'm not a programmer, I'm not a coder, I'm not a geek." But you're really limiting your own income if that's your point of view.

It is possible for you to understand certain things about software, certain things about managing coders and then you go ahead and hire coders and let them do the groundwork and the legwork, but you plan it out. In the same way that you might get an information product created or ghost-written, where you create the table of contents, you decide what should be covered, and then you let them do the dirty work.

Where you might outsource articles, and if you just go and say, "Here you go, here's a few hundred dollars, go ahead and create a hundred articles for me," you don't know what you are going to get back. But if you did the keyword research for them, if you decided, "here are the titles of the hundred articles, here are the points I want you to make," then you're going to be a lot happier with what comes back. And the same is true of software. Even if you don't want to become a full-time software creator, even if you're just a local business, or an online business or an information marketer, you can benefit by having even a simple piece of software in your business.

Even a simple piece of software that maybe you bought rights to or bought rights to the source code, and now you're going to get it improved. Or maybe you just created what we call a "spec," a specification for a piece of software that maybe you've always needed in your own life. Or you might have a course already, and as part of this course you can include this piece of software, to basically outshine and protect yourself against your competition.

Now think about this: whatever information or course or membership site that you are selling, there are a few levels about it. So let's just say that the first level is to create an information product. You say, "here is my report, or here are my videos, on - let's just say- how to create a Facebook fan page. How to advertise on Facebook." That's one thing that you might decide to talk about, and that's a product right there. Now let's take it one step further and let's make it a "How To" product. A product where you take someone from point A, which is in this case not having a Facebook fan page, to point B, which now is you have a Facebook fan page, you have Facebook ads, you have this whole entire business where your leads come from this popular website.

And now, let's take it even another further step, let's say it's level three. Level one was creating an information product, level two was to make it a How-To, point A to point B step-by-step solution and now level three is training on how to use a specific particular piece of software.

Make It Easier...

There's software out there that helps you figure out, if you put in a keyword, which advertising they go after, which fan page is the target. You might say, "here's my course on Facebook, and as part of it, or as part of one of the modules, I'm going to give you my affiliate link to go and buy this piece of Facebook software." Or maybe if they don't have the affiliate program you can say "here's a resource you can buy," or if you can work out some kind of licensing deal, which I've done many times, you could say "I've included, I even bought your copy of this piece of Facebook keyword software, and as part of this one module, I'm going to show you how to use it." Let's say that's level three. So we got level one: info product. Level two: step-by-step course. Level three: instructions on how to use the program.

You might get to the point where you think "it might be a lot easier if I didn't have to send people off to someone else's software product. Or if I didn't have to kludge together all these solutions." What if you had an all-in-one tool that would somehow create your Facebook fan page for you, set up your ad campaign, maintain it, or did all these different things that you, as the product's trainer, you as the person who has developed the system and is now marketing it, this software matches indirectly with your system and nothing else exists like it, or at least nothing else exists with your personal twist. I'm not saying that you have to become a totally different person/marketer, but is it possible that there's something in your business, whatever it is, where a piece of software can benefit?

Even all these local retail stores, for example Walmart has a cool iPhone app where if you're going to buy a TV... what's the problem with buying a TV at the store? You don't know what size you need, you have to do all these measurements, but Walmart... I'm not sure if they still have it, I checked recently and I couldn't find the app - used to have an iPhone app where you could take a picture of your wall and it would somehow figure out the distance between you and your wall. I think maybe you had to walk backwards or something like that, but it was just a picture of your wall, and they drew a square, like a bounding box, to figure out what size TV would work on your wall. So you take this photograph, and then draw a box over it and then it would say "that wall is good for a 52" TV."

I don't have to tell you that different pizza chains have the ability now where you can order online or you can order on an iPhone app, you can place an order on your app and then go ahead and pick it up. All kinds of businesses can benefit from some kind of software app. And even if you're not a local chain, even if you sell training or have a book or something, software will really help you. Even something as weird as if you're a motivational coach or you're in self-help, time-management, productivity, feng shui... you can create a piece of software that helps people stay organized. Like your own unique to-do list or a piece of software that contains different kinds of hypnosis audio or things like that.

What's really cool about you having a software product of your own is that selling it is super easy. Let's think about this: Say you have this course that has fifty videos, and it has all these modules and all these bonuses. It becomes really difficult to explain it, right? And you come across this issue of "well, now it takes me this long sales letter, this long video, this long webinar, to explain everything that's in my course."

My other weakness is "what if I list fifty things and someone only wants three of them?" Now we're looking at this whole big package thing, "I just want those things, the rest of it will be wasted." And also, competitors. It's very easy for competitors to jump in and pretty much clone and knock off your entire course, and of course not do it as good, but they may look at your sales letter, or look at your table of contents, and try their best to teach it, and then it's hard for someone to distinguish them from you.

On the other hand if you had a two-minute video showing your piece of software, "click-click-click," and this big, complicated, tough task that people used to struggle with, is now done in just a few seconds. So now you can have the quick commercial, the quick infomercial, and just demonstrate it, and people buy based on that. And even if the piece of software has already been created, you can add your own unique twist.

I'll give you a few examples. For years and years I have sold what was at first a piece of script and is now a WordPress plugin called "Action PopUp," which is a webpage pop-up tool. You can pick any webpage that you have, and then have this thing where it dims down the screen, it puts a box in there, you can put in a video, link to website, or an opt-in form, and Action PopUp is the first - there are a few now as well, but it's one of the very few - that is designed specifically for email opt-in forms. What I mean by that is that if you have a pop-up, you pop this thing up and it says, "fill in your details to join my email newsletter."

With most pop-up software it was really annoying because it would take someone off of your webpage, and this was the first and only plugin where you could put in an email opt-in form and it was smart enough to submit this in the back-end, so this pop-up would come up, fill your details, hit the button and then it would fade out and people would stay on the exact webpage that they had been on without any fancy programming.

And there were other things too, like I could make it so that the pop-up wouldn't let people past, so people would have to submit their details. And all these cool little twists and hooks that other pop-up creators didn't think about because they were all egghead programmers who didn't actually use the thing they were talking about.

Now Backup Creator. It's a tool, a plugin that backs up your entire WordPress blog including the plugins, content, files, videos, all that stuff. What makes Backup Creator unique is that it's very fast. Which means that, we still are the only one that do this, where you can directly copy a backup from one server to another and so we can, in about two minutes, back up the plugin in one place, install the plugin in your brand new site and move it over in just a few seconds, and no one else does that. The big flaw of other backup programs is that it was very technical, you had to backup, download, go over here, upload, wait for it to finish... and ours was just as few steps as possible.

I could go on and on, we had plugins like WP Import, that imports all the content in your blog, Paper Template which is a sales letter plugin... and what happened was, we were not the first backup plugin, we were not the first sales letter plugin, I didn't even create the first pop-up software, not by a long shot, but I noticed a gap in the marketplace, I noticed a way where I could put in my own unique twist and I'm sure that you can think of different ways that you can this as well.

One of my mentors, Armand Morin, created a piece of software, a whole lifetime ago, which was an audio player, and you could record an audio for your webpage, put it online, have one of those little buttons, and the irony of it is that if you are a programmer, it takes about five minutes, if that, to make it, but it's a solution that people are willing to pay hundreds of dollars for. They pay 20, 30, 50 bucks every single month just to have the ability to have a button on the webpage. The same is true with video players, things like that...

Someone else who I like a lot, Russell Brunson, had different kinds of software, and one that comes to mind was called Zip Brander. Back in the day, everyone had unzipping software installed, and maybe if you sold a series of PDF documents, or even a piece of software or anything like that, you had to zip it up, put it in a little package. With Zip Brander you could bundle your own unzipping program and put on

a little ad there so that maybe you could have an upsell to some other product you sold. So there are a lot of cool little twists to things that already exist out.

Software comes down to three components: number one is to get an idea for your software. Number two is to create a spec, a specification, a blueprint for what the software is going to do, look like, and all those kinds of things. And then finally put it out in the marketplace and sell it. First things first: How to get different ideas for a software. I hope that you have an idea already but if you don't, I think that there are probably four different types of ideas for software, so let me give them to you all and then we will sort through each one.

Approach #1: Better Solution to an Existing Problem

First of all, a better solution to an existing problem. Number two, to simplify an existing process. Number three, to combine multiple existing solutions. And number four, an add-on to your existing product or course.

A better solution to an existing problem: let's say that there is some payment processor, or an affiliate program, or some online service where you can do a better job. For example, there are several different services that allow you to run webinars, streaming video over the internet.

The most powerful one is the GoToWebinar, and people always have complaints like "it doesn't do this, it doesn't do that"... What if you created your own low-cost webinar service with a low monthly fee or a one-time fee, that uses Amazon S3? Amazon S3 is a file storage service, but what a lot of people don't know is that you can actually use it to stream video through it, and you can create a Windows or a Mac program that could record the existing screen that is in front of you and then slowly put pieces of that video up on Amazon S3.

Then when people were viewing your video they would download and stream the video as you created it. You can say "I have some idea of how complicated that is to create but that is outside of my personal skill level," so in order to get a little bit into how to organize this and how to make your intentions clear about what you wanted to do, that way you don't have to do it, you don't have to learn all this stuff but you're also happy with what comes back.

Another example of that: There are all these online services that you might not even know about, but there's one called Twilio which basically lets you connect a telephone to anything you want, to the internet.

For example, you could create a call-forwarding service or a voicemail service, and it's all done by what is known as "system calls," API calls, basically talking to this infrastructure that's already there. Also, based on Twilio, I have a friend called Chris Brisson, who has a service where you can build a list of phone numbers, kind of like with Aweber, which is an auto-responder service, and you can give people the ability to give you their phone number and build this list of 1000 phone numbers, so when you have something to say to them you can send a text blast or a voice blast to 1000 phone numbers, you can even put them on a step by step sequence.

I have some courses where if you join my course then my system will call you every single day for a minute and give you a motivational message every single day about where you should be in that course, and it uses my friend Chris' service, but his service is based on this thing called Twilio. So there's all these better solutions to things that are already out there but you can add your own little twist. That was number one: better solutions to an existing problem.

Approach #2: Simplify an Existing Process

Number two: simplify an existing process. There's something out there that you see people doing over and over again that could be simplified. You might see different tools that maybe publish content online, and someone I want to mention in a few minutes is a guy named Joel Spolsky. He has this cool blog called Joel on Software, and he had a program called CityDesk where you could do just that: you could run a website (like we do WordPress now) and organize all the stuff and it would put the whole website online, and there were templates, you could manage everything from this program, so instead of having to upload files or learn HTML or graphics, his desktop software would do all that.

I'll give you a freebie right here: think about, maybe at this step if you know WordPress or things like that, the process of putting a video, recording a video and putting it online, putting it on WordPress... there are a lot of steps there. There's five or ten steps, you might have to use three or four different pieces of software.

One to record the screen, one that can invert the video, then you have to go and find a video player, etc. What if you created a plugin that, installed on a WordPress blog, you could click a button and then it would record the screen, it would install a Java program or something, and as soon as you're done it would save it, put it online, put it on the video player, just in one click? That would be awesome, and at the time I'm recording nothing like that exists, and that is simplifying an existing process. Something that might take several steps, you just do it in one step.

Approach #3: Combine Multiple Existing Solutions

Something similar to that, is number three: combining multiple existing solutions. The second type of software idea we've talked about was simplifying existing processes. Something that maybe took ten steps, you can combine it into one step. What if you could already accomplish those tasks but you had to join together all these different tools? A good example: one of my good friends Robert Vance, wanted to be in the niche of WordPress security.

He knows a lot about locking down a web server, things like that, and that's a subject where people need to know, but he just had training. It was only how to lock down a web server by making all these changes. And people saw that and they said "I don't want to have to do all this work. Can't I just push the button and do it?"

I asked, "can you create a WordPress plugin that will secure any WordPress blog?" And he said, "well, there are already things like that that exist." And I looked at it and thought that the flaw here is that in order for someone to actually fully protect a WordPress blog they have to install five different plugins. They are all free plugins, but you have to install all of them, and tweak all of them...

What if you could have an all-in-one solution where they could just install your one plugin and it would lock it down, and it would do all the stuff that maybe you normally do to trick out your blog, but now it's something that you use? You're eating your own dog food - to use another programming term - where this is a software product that you actually use on a daily basis. Something that was holding up your

business, that was causing you extra time, and in order to get a productivity boost you had to create this plugin and now people can also buy the plugin.

Another idea for you here, another freebie: the process of creating a podcast takes a lot of steps. Once again, I have to record the audio, I have to convert it, I have to add tags, I have to upload it to a file storage service, I have to then connect a WordPress blog post to that podcast... I would love it if there was just an all-in-one tool where maybe I could click on "create new podcast," I would type in the name of the episode and it would ask for whatever tags, I click a button, it would record it, save it, upload it, create a new blog post... maybe I would have to save a few passwords in there, but it would just be awesome if there was an all-in-one app or desktop program that would interface with all these different services that I use to host a podcast.

A final one for combining multiple solutions: another area I have difficulty with is creating graphics. I don't really trust a lot of people to create graphics for me, I'm not the best with graphics, sometimes I have to create stuff and come back and look at it... There have been banner generators in the past, logo generators in the past, but what I would really like is a graphics generator, like a cover generator, where maybe you would have several good-looking templates and I could just pick the one, put in the name of my product, put in the different details and this cover generator could create a book cover for me if I wanted that, a CD cover, a cover that's compatible with Kindle, with CreateSpace, with Kunaki... one where I could just click and apply your template and now I could use that artwork on all these different services.

These are all real things that personally I would love to see made. If a solution already exists go ahead and let me know by going to RobertPlankShow.com, which will take you to our Facebook fan page, and just post there and let me know if I can get any of these software even if you created it. That would be even better! A great way to promote your own software product that you had created.

And finally, an add-on to your existing product: I gave that example of Facebook software. Lance Tamashiro and I created a course a few years ago called "Set Up a Fan Page," which was basically a course on how to set up a fan page, get fans, create content, get advertising... We were not the first fan page course and we were not the last, and maybe people who came after us outsold us because they bundled Facebook software with their course.

A Facebook plugin, for example if you've seen what we call "fan gates," where you try to go to a fan page and it says "before you can even look at what's here you have to click the 'like' button, or maybe you have to sign up for something, and then you can get access to this fan page". Another thing is that with these fan pages you can create custom backgrounds or custom header graphics, so many people created generators and software where it would create the perfectly-sized Facebook background or Facebook header graphic for a fan page and it would simplify this process to get people to where they wanted to go, which was to have a fully functional, completed fan page and this software and these plugins allowed people to do that.

Another pain point is installing a website, getting a WordPress blog online. And many people actually create tons of WordPress blogs, and at some site builders that set up Amazon sites or ad set sites or

video sites every day, and sometimes they might need to set up fifty or a hundred new sites per day, so what if you created some desktop program that would set up WordPress blogs? In one click you could even put in a list, say "I want to create all these blogs, and all these topics," and in a few clicks you could set up the blog, put a theme on there, put the plugins you wanted and then grabbed different pieces of content or dripped content... that would be amazing.

There are a lot of possibilities there. The different types of software ideas that I have given you are: 1) A better solution to an existing problem, like a webinar service that uses Amazon S3. 2) Simplify an existing process, for example a WordPress screen-capture plugin. 3) Combine multiple solutions, like a podcasting creation app, or 4) An add-on to your existing publisher course, like if you teach something like Facebook for example, or you teach something like setting up a WordPress blog, what if you bundled this software that would make the process so much easier? And then people would buy just because of the software. The training is now secondary, so it's easier to pitch and you have less competitors.

Niche Software

There are a few things to worry about, which is why you need to continue listening to this podcast right now. I hope that we are getting a few ideas in your head. So if you are in real estate, if you have a real estate course, could you put in a mortgage calculator in your membership site? Or if there was some reason why it needed to be in a desktop program, then something like a desktop program, something that would grab these real estate listings, maybe help you calculate, identify for you, even pop up an alert on your desktop if there is a piece of property for sale that falls within certain parameters. Let's say that you had a real estate course about buying foreclosed homes, buying a bank-owned house, and then buying it and flipping it and making a quick twenty thousand, fifty thousand dollars.

Maybe there is some formula for identifying a piece of property you should bid on based on if it's going to be easier to sell, if it's underpriced, if it's in a good neighborhood where it will sell fast... and then you could have this piece of software that would put up an alert if there was a home within your area or within a certain mileage range where you could just go and grab that listing.

What about something like, even for a realtor, some kind of a WordPress plugin - and this would be actually pretty easy to do - where they use this thing called Geotargeting. What's Geotargeting? Let's say that you live in Los Angeles, California. I could make a plugin where if you came to my website from Los Angeles, California, I could show you a webpage just for Los Angelinos. Then if you live in New York City, I could show you a different webpage, and I can pretty much detect where you live based on your IP address. So if you know the right keywords, if you know what is called geotargeting, then you could create a WordPress plugin for realtors, or if you are a realtor you could use this, where you could say "I want to show this one landing page if someone lives within fifty miles of this location." That way you could throw a landing page showing local homes, and if they don't live within the area you could get them on a mailing list for some general real estate training. That could be pretty cool, right?

Maybe you could decide whether you want to sell them these five cities nearby, and based on exactly where exactly they're at then you would show a different kind of landing page. You could personalize or customize it, you could make a plugin that manages multiple real estate listings or manages the images and the videos and all the normal things that you would list in selling a home, you could make a real estate WordPress... You might have seen WordPress themes specifically made for restaurants, specifically made for doctors and dentist offices, where it was built in where you could make a reservation for that restaurant, you could book an appointment with that doctor and even have a follow-up, and this is all WordPress plugins.

What if you were in weight loss, and you somehow created either a desktop program, or an iPhone app or something in a membership site where you walked someone through a 30-day meal plan? And I'll give you another example: a few months ago I bought this thing called the Juicer. It's not a blender, but it's a juicer where you can throw in an orange and then out comes liquid orange juice. Throw in an apple, out comes liquid apple juice. Not blended together, but just the juice part of it. That's a cool thing, so what you can do is, if you want to lose like a pound a day you can switch to only drinking this kinds of juice, you can literally drink 10-20 apples in an hour just by downing these glasses, and get all kinds of nutrients and all that, and if you have a lot of vegetables you can get full from the vegetables, so

the idea is that you just taste the fruit so it doesn't taste like you're drinking a tomato or a cucumber. There are all these different juicing recipes, and you know what I would love? A 30-day software or membership site where it would walk me through, based on my taste, different kinds of things: spicy drinks, fruity drinks... just so I could figure out what my favorite juicing recipes are and you could show me ahead of time what fruits or things to buy at the grocery store and I would just blindly go through these 30 days and maybe even rate my favorite kinds of juices so at the end of the 30 days I could know what the ten juices that I want the most are. I would pay a couple of hundred dollars for that, to be honest.

Even things like if you had your own special weight-loss plan, something just for people that track their weight, track their meals, track their exercise or a calorie counter. There are a couple of iPhone apps for this, even ones where it's even built in that you take pictures of the food that you eat, and there's actually studies that have shown that if you take pictures of the food that you eat you're likely to lose weight because you won't eat disgusting things. Different things like that where you can have a lot of fun and still create simple software.

A good friend of mine created a time management software years ago, where basically you would list the tasks that you had to complete and estimate how much time each task would take and when you actually went to accomplish those tasks the software would count down and at the end of the day or at the end of the week it would give you a report of how well you did, which tasks you finished, which ones you didn't do, which ones took less time, more time... it was pretty cool.

What if you created a hypnosis software? I have seen a lot of people who sell hypnosis audios where you can listen to an audio for an hour and go to sleep instantly or improve your focus, your energy, your memory, overcome fear of driving, fear of elevators, fear of clowns, all kinds of crazy stuff. But what if that was all bundled into one software package? And you could just say "I want this, this and this." And then maybe blend them, or maybe even customize them. What if there was a software - I don't know if this is possible - where I could record specific things that I wanted to remember, or keep in mind, or should motivate me... Let's say that there was a hypnosis on confidence or increasing confidence. Why don't I record things like "I want to be more confident when I speak on a webinar or when I speak on stage or when I go out on a date," things like that, and I could maybe record those or could somehow mix my affirmations with the audio. I don't know if that is possible or if that is a thing, but it would be an awesome piece of software.

I mentioned dating: there are a lot of cool possibilities for dating software, and you might think of things like Match.com or eHarmony... that is a little too big of a scale. One thing that comes to mind is, if you are going to create a dating software, I don't know if you have heard of this guy Tim Ferriss. He has a few blogs, one is called The Four Hour Work Week, where he talks about time-management, outsourcing your life, having this cool lifestyle... And one thing he did was he outsourced his dating. What he did was, he joined this few different dating sites and he had an outsourcer, I think he even created his accounts for him, his profiles... He said "here's my calendar, fill it. I want to go on a different date every single day of the week and then I'll report back on who I want to see again or make future plans, and then you figure out sending flowers or a note afterwards, based on how the date went."

Pretty cool idea, right? So if you think about the pain that goes with things like online dating, like creating the profile or figuring out all the time management... What if you had some kind of a service where you said "I'm going to join all these dating services for you and I'll create the profile based on what you want to say, or sex it up a little - pun intended - and then I'll even figure out who you match to and simplify and arrange the dates for you," that would be pretty kick-ass. And if you think that sounds like a lot of work then charge more for it, charge whatever you think it's worth, and if you had a piece of software that could somehow go out and create the profiles, or interface with the outsourcers - that might be a little too grand scale - but there's always something that software can apply to.

Just look at the different iPhone and iPad apps that are out there. There are apps out there that can help you learn an instrument, but not in the way that you personally teach it, or not with your own unique twist. And even if you want to keep it really simple, I have a plugin called WP Notepad, and that will allow you to create fill-in-the-blank forms for your membership sites, for your WordPress sites.

Let's say that you had a real estate course, and you had your friends legal documents, like a lease or an eviction notice form or things like that, where you had to fill in the blanks. Our plugin allows you to type in or it will ask you these questions and then it will generate this whole form for you.

There are lots of cool ideas, and software can be as simple and as easy as just creating an audio button. A couple of things I want to warn you to stay away from: stay away from big games. Games sound like fun but games end up being a big pain in the butt to debug and get all the kinks out. Another thing to stay away from is the software that is mass market. You should be just conquering and dominating a tiny, tiny little niche.

We've been talking about self-help, dating, weight-loss, different WordPress plugins that accomplish simple things, but if you go out there and say "I'm going to create a word processor, I'm going to create the next Gmail, I'm going to create a video editor." That's a little bit too taken over. You don't want to have to go up against Apple, Google, Microsoft... they're in their own area, in their own space, they are never going to drill down into the tiny little mortgage or dating niche that you want to get in to.

Simpleware

You want to create what I have called and have heard other people call "simpleware." We have software, but if you think about this term "simpleware," we just want to create a simple software product that solves one thing: tracking your weight-loss, importing content, playing a video, it's sold as one simple thing. Why? Because it makes it easy on you, you don't have to make some big, grandiose thing that takes years and years to create and it's easy to demonstrate and easy to sell a bunch of copies because it's simple.

By the way, I know we are running a little long on time but we have a lot of stuff to cram in, and I want to send you over to a webpage called SoftwareSecretsExplained.com, and when you go there, there is a $50 product for sale that contains a report, a video course on how to create an outsource for software. The most important part of this, after you have your idea, is to create what's called a "software spec." What that means is that you want to explain in plain English exactly what you want your software to do. And you don't want to go overboard and make this hundred-page document, but you still should at least have a page or two about what it is you want your software to do. You might actually feel like you're doing some of the software programmers' job for them, and you'd be right, but there is a thing which is that you are the creative brains and you are the idea person and you know what you want.

You know what you will be happy with, your programmer is not a manager so it's better for you if you make those tough decisions early and just map out for them this simple piece of software that you want to create. And what you are paying for when you get them to turn that drawing and that English terminology and all of that, those sentences into programming code, is to duplicate what you have explained and getting all the technical stuff figured out and get all the bugs out.

I used to be a freelance programmer and I was always frustrated because people would tell me what they wanted and I would always say, "draw me a picture, draw me a picture. Write something out there, use Microsoft Paint, or draw on a piece of paper and scan it..." No one ever did, and the they would expect me to create the interface and then they would be unhappy with it, and we would always have to go back and forth and finally I would have to do what I felt was doing their job for them.

You need to figure out the interface and hopefully the software that you have in mind, you can picture in your head what you want it to look like, and there are already competitors so you can go out there and download those apps or install those WordPress plugins and figure out how they have arranged it. Of course you don't want to have an exact knock-off, you're going to have your own ideas about how things should be arranged, so maybe you say, "I want to have this button here, I want to have this text box there," and it's up to you to design the user interface any way you can, even if it's drawn on a piece of paper.

There are a few tough decisions that you need to make early on, and I'll list a few for you right now: 1) What platform will it be on? 2) What data structure? How do you store your data? This might mean a few technical things for you. You don't have to go as far as knowing coding but you do need to know enough to be dangerous. 3) User story: what will the user actually see as part of your software program? and 4) Your non-goals. What's a "non-goal"? It means what won't the software do.

The Iceberg Principle

Whenever you make software you're going to be tempted to add and add and add, and make it this crazy thing, but you are going to really have to be a little counter-intuitive and force yourself to make a simple piece of software that maybe doesn't do everything that you want to do eventually, maybe keep those in mind, but you want to create version 1.0. And the reason for that is this thing called "The Iceberg Principle." If you look at an iceberg - I've never seen an iceberg but I've seen them on TV, we've all seen the movie Titanic and things like that - the thing about an iceberg is that the term "berg" means city, it's a chunk of ice that's as big as a city.

The problem with an iceberg is that you only see the tip. So boats go along saying, "oh, we'll crash into this tiny little thing," but what they don't see is that 90% of this iceberg is underwater. And they're about to collide with an island, basically. So the iceberg principle means that you only really see 10% of software, you see the fun stuff. Like when you get the first version of it going. But then 90% is debugging, getting all the kinks out. I don't know how old you are or how long you have been using a computer but if you remember, earlier versions of Microsoft Windows crashed all the time. Like Windows 95, 98, 2000, you had to restart your computer every day, and blue-screen, and even modern computers still blue-screen. Even my iPhone, iPad, every now and then crashes, or an app crashes, or a web server crashes. So that's where most of the elbow grease goes into, just taking out these bugs. That's why you need to have the most simple version 1.0 possible.

Let's go over these things. What platform will you run? This is a very simple question but you'd be surprised how many people just don't know. So this idea you have, is it going to be web-based? Is it going to be a membership site someone logs into? And if you can even say "I want this to be compatible with WordPress and Wishlist Member," that's even better. Is this going to be web-based or is it going to be a WordPress plugin? Is this going to be something that people install on their websites themselves? Is this going to be a desktop program?

If so, should it be coded in .NET, which means that it can only run on Windows, or should it be coded in Adobe Air, which means it can run on Windows and Mac, although people have to install this thing from Adobe? Or should it be an iPhone app? Either way, you need to know very first thing, after your idea, where will it live? Where will it exist? And now you can figure out the interface.

As someone who used to be a programmer, the one thing I would have loved seeing is: when someone installs this plugin or this program, what's the first thing they see? What's that screen, what are the things there to fill it with? Then what would really help you - this might be a little beyond your abilities - but, as best as you can: What is the data structure? How will the information be stored?

I'll give you an example. If you were designing a membership plugin for WordPress, you'd have to be thinking you need a place to store the list of users somewhere. We need to store a list of levels somewhere, store the list of users and levels, so you need to have an idea of what would be stored in the database.

Since I'm talking about WordPress plugins, there is a tool in your cPanel called phpMyAdmin, where you can just goof around, browse around and look at how things like WordPress plugins store information. This is not required but it will really help you to figure out how an iPhone app stores information, what the structure is, how a WordPress plugin stores information.

Even if all you did for a software spec was to draw out the interface and figure out how things are going to be stored, that would be a huge help. Maybe a programmer would look at it and say, "I would change this and this," but if you knew a little bit about the data structure, I'm not saying coding, but if you just knew how things were stored, that would really go a long way as far as you planning out your software project.

I'm always surprised and amazed at how many people don't know that software architects exist. You have the managers, the marketers, the coders, but then when you're dealing in corporate America with these teams, you may have this person in the middle, a software architect, who doesn't even know how to code, who doesn't know marketing or any business stuff, but can draw up these maps, figure out the data structure and say, "this is a way that makes sense. Here's how to do this interface in a way that makes sense, that people will understand."

It's intuitive, it doesn't require an instruction manual, it will just work. So if you can figure out the data structure and how to organize different things that are stored, that will help so much, not enough time to go into it in this call.

The next thing that's really important though, is user story. It's one thing to draw out what this will look like and even how we will store it, but then, what activities can a user take once they use my plugin? What you're going to do is tell a different story about what someone might do with your program. So, these WordPress membership plugins, since we're on that topic: Story number one might be, "now that it's installed, I want to create a new membership level."

You would click on this tab, fill in this box, fill in this box, click Save Changes. Maybe another story is adding a new user as the site administrator. Then I would go click on this tab, fill on this, fill on that, chose that, save it, that's another story. Maybe then someone might join the site, and they would click on this button, click on that, fill this, fill that, click on the button, they'd be sent to this page, they would see that and you just get it as detailed as possible, where you explain what someone will go through, what steps they will go through in every possible scenario.

You might have five or ten user stories. And I'm just talking about a simple bullet-point list set out: they do this, they do this, they do this. Like "here's a user story about how I will delete a user. Here's a story about how I will create a payment button for that membership site. A user story about how I will give a user a certain access to content."

This is so important because just by explaining this someone can see, based on the interface that you've talked about, or drawn out, "here's how someone will use it." And once the software is done you have all these different tests to debug it, and you can say, "I have my 5 user stories, I have my interface, how to store it, you come back with a version 1.0 and we'll run it through these tests, and if it behaves as

we've expected it then that's great and now we know we have a workable product." Of course when you put it out on the real world people will inevitably find ways to break it, and you go back and make version 1.1, 1.2 and improve from there.

Then we have non-goals. We have the things that in version 1.0, you might even list "this program will do this, this and this." We got a membership plugin, you can manually add users, delete them, they can sign themselves up... but for now, for this first version, we will only process payments from the Paypal payment processor and we will only run this membership plugin to have one membership level, not multiple levels. But you make it a point to point out that in the future we'll have the ability to store multiple levels.

The reason why you point this out is because when you're planning your data structure or you're working with a programmer to figure out how things will be stored, you'll think ahead, in the same way that maybe if you're building a house you might put all the bathrooms on one side of the house because it probably makes sense that way, but maybe for now you can only afford to have one bathroom - to use a really bad analogy. But you're going to list your non-goals, things that you, on purpose, will not do. Like for example, all those membership things, or a back-up plugin.

There are a ton of things we could have added in there, but we said let's just sell this plugin on how simple it is and how it doesn't do all that stuff because you don't need that, you just want to push the button. You want to be able to use it and that's it, and then you want to be very careful in the future about adding those features later.

Let's talk about that for a couple of seconds. There's a book called "Rework," it's a really easy read, but it's about creating products, creating things for your business, solutions, and basically don't add too many things, don't make your plugin or your software a swiss army knife, but when you talk to your customers if you keep hearing the same thing over and over, to the point where you can't get it out of your head, then it's time, maybe, to add that feature.

If you create an iPhone app and you keep hearing over and over that maybe people running a calorie-counter app want to check their stats or pick their stats from the computer sometimes, log into the site or have a Windows program, then maybe it's time for that, to make a mobile and desktop version.

Maybe you want to have a feature where people can interact with other people, some kind of social interaction or social media, some Twitter or Facebook connections to these apps, and you can always add things like that, things like licensing where the plugin or the app calls home and that way you can charge for updates or charge for add-ons or have these anonymous statistics, and then you can say, "fifty thousand websites use our plugin," or things like that.

The point here is that you need to decide, make those decisions ahead of time. Like what platform you want to use, what data structure, the interface, user stories and your non-goals. There are people who explain this in a lot more detail than me and I'll give you a couple right now: Joel Spolsky has a book called "Joel on Software," and he'll sometimes venture into the programming stuff but don't let that stop you, just read it for what it is. You don't have to be a coding or a computer geek to understand a lot

of these strategic, management, planning types of things about software architecture. "The Mythical Man-Month," an oldie but goodie, how to manage a team of programmers, "Effective Programming" which is more about productivity hacks when figuring out software... And then, kind of a weird one, called "Headfirst Design Patterns" which will help you understand some of these different ways to structure software. Once again, even if you're not a coder or a programmer, it's still a good read just to understand how to organize these building blocks, so to speak.

You have your idea for a software, you have our specification, now how do you hire a programmer? How do you get a good programmer? There are a couple things. One would be oDesk. oDesk is cool, it's a job hiring site and you can type in a skill, so if you know that you're going to have this iPhone app created, you can type in "iPhone developer" and then get a list of people who are iPhone developers, and they're down by their location, their rating and their job history.

What's also good about oDesk is that you can post a job, hire them, explain your specifications, and even I would say to look at other people's jobs and model what they've put as far as what you have, and if you can find a similar-looking job, then you can kind of, not copy, but look at what they've said and you might catch things that they've missed or think of how to explain these different things like user stories or non-goals where you can actually have a better listing that them.

With oDesk you can make it a fixed-price job or an hourly job. You can say "I think you should charge $500 for this, and that's it. I pay you $500 and you make the software to my specifications, and then once you actually fulfill that you'll get paid." The other option is hourly, and you can say "I want to pay $10 an hour," and they might bid $8 or $15 and then once you've hired them it will actually take snapshots of their screen as they clock in time for you.

You're paying them by the hour but it's not like they're dragging their feet because you still see what's on their screen. What I would recommend - and this is where it gets a little tricky - is that you want to post a really simple job. You want to post something where it's a few hundred dollars, you make version 1.0, maybe plan for the future a little bit so you don't get painted into a corner, but just a really simple mini-project that maybe takes them a couple of days and that's it. Maybe a week and that's it.

Maybe it costs you $500 and that's it. So you have that really simple software product and you can hire them again if you like them, and you can look at things like if they have completed their code well.

You might be tempted to hire someone based on a low bid or a low price, and then you get what you pay for. But if you can, try to hire someone like an American college student.

You can tell by their picture if they're young, you can tell if they're in the US, and then they have a low bid, and if they went to school or to college or if they're attending they'll put that in their profile and their description. If you hire an American college student, they're actually in the process of being trained the proper way to program but they haven't proven themselves, they don't have a day job yet, they're just trying to make some money on the side... that's a good match for you.

As part of my hiring process for things like article writers and transcriptionists and editors, I have them take a really quick test as part of the hiring process. What I would suggest is, maybe if you know a coder (this might be a job to hire out), take a piece of code of something and put an obvious error in it, maybe so that you post your specifications, 30 people come back to you with their bid and then you email your response and say, "if you want the job, here's this broken line of computer code." Something that hopefully they can't google or look up.

Or how about this: open up a random WordPress plugin and take out a couple of lines or a couple of letters, or add a couple of punctuation marks and just say, "this is broken, how would you debug this?" Something that they could just, in a couple of minutes, figure out. I think that alone would help you to narrow down the hiring process.

That way you go on oDesk, you post your job, post your specifications, you say "I want it to look at this, here's some of the user stories, it won't do this, this platform, here's my budget." And you base your budget based on the other jobs, of course. People come back with their bids, and if they don't have enough history as you'd like, or if they're in some Third World country or maybe not, or maybe they have some good ratings that could help... if you can find a college student, great. Then give them this audition, this broken line of code, and if they can fix that easily... I guess you'd have to find a way of seeing if they're right, but if you have enough of these responses come back, and if they're all pretty much the same answer, then you know they're telling the truth, so hire the first person to respond and there you go.

That would be the hiring process. The key here is knowing what platform, go on oDesk, type that in, look at similar jobs, you'll know what to have as your asking price, if the bids are good... You kind of have to take a little bit of a risk but if it's good then you can hire them again to add additional features, and it would be great if you could re-hire the same person because they know what they've coded and they don't have to figure out what some other programmer did.

The Joel Test Explained

I mentioned Joel Spolsky earlier. Joel, i f you're listening, then email me, robert@robertplank.com. Anyone who's listening, if this session, this training helped at all, then any feedback or any comments email me at robert@robertplank.com, or even better, go to robertplankshow.com and like the page and leave your comments there. What's so awesome about this guy Joel, is he has this thing called the 12-point Joel test. And a lot of these things won't apply to you, just because you're starting out with a programming team of just one and it's outsourced, but if you're getting into software there's all things to keep in mind, it's called the 12-point Joel test, and the more of these things that you can say yes to, the better of a programming process and an environment that you have.

I'm just going to go through them and you'll take from them what you will:

1. Do you use source control?
2. Can you make a build-in-one-step?
3. Do you make daily builds?

4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have the spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during the interview?
12. Do you do hallway usability testing?

Let us breeze through these:

1) Source control:

Do you have the ability for programmers to write code and fix different things and then check that code? In that way there's one central location where the software will save, you don't have all these conflicting copies of software. This is really important if you're dealing with multiple programmers working on the same project. Maybe you're not at that point yet, but if you ever get to the point where you have like a monster of a program, or it turns into this big thing, because what you can do is have, out of this team, one person who checks out code, checks out the whole thing, makes the changes and then checks those changes back in. That way people aren't overwriting different things.

2) Can you make a build-it-in-one-step?

This is really important. And this might not concern you, especially if you outsource it to someone who you haven't met, but if they can make a change, a quick change, and just click a button and it's all online, it's all compiled, the plugin's ready to go, that's really helpful. That's what I do when I create my WordPress plugins. I can click on one button and that latest version of the plugin is there, the code is up and protected, it's packaged up, it's put in the members area instantly, from one button, I don't have to go through all these different things so I don't make mistakes.

3) Do you make daily builds?

This means that basically every day you have - let's just call it a beta version - a version of what you have so far. So you have the public version of your software, the one that's been tested as far as you can do and then you have these different alterations where you're slowly improving on it, but the software never at any point breaks. You're just adding on these fixes to it and that way you can always roll back, so maybe if you lost something or you had to go back to something, you could go back to two days ago and get to that earlier built.

4) Do you have a bug database?

You might know of something like this, and it's kind of similar, called a support desk. Instead of handling issues or bugs or problems with customers over email, have a support ticket system where someone can submit a ticket and say "here's the problem. When I click on this button the whole thing errors out.

There's this big nasty screen." What you can do is they can submit a ticket and then you can assign it to one of your programmers and then they can figure out what the issue is, they can fix it, they can mark it as fixed, and then that way if the bug ever comes up again you can find out whether the bug was fixed or not, or what's the solution, or if it really is a bug, so that way everything's all organized in one place.

5) Do you fix bugs before writing new code?

Here's the thing: adding new features is fun, it's sexy... fixing existing features isn't as fun, but you want to have software that works. So at any time, if I have a certain limited amount of time or a certain limited amount of things I can do, and the choice is between keeping the features at a small amount and fixing the bugs that are on there, I'm gonna fix those bugs. Because you don't want to make your software even worse, and make it buggy and do all these things but it breaks all the time. If you have bugs be very careful about introducing new features and adding new code because you're only going to amplify the problem. It's better to play it safe and fix the bugs that are there before creating new versions.

6) Do you have an up-to-date schedule?

Do you know exactly when your software product will launch? When it will be done? When you're going to go into the debugging phase? And even as far as that, what milestones can you create? Milestones are like the best thing a programmer can do. And it's good for you as the software architect or the hiring manager or whatever it is you want to do, because you can say "here's the date. In a month, the software will be done, but I want to have it ready to test a week before that. I want to have maybe some of these features on it." This way, you can hopefully have a product that stays shippable all the time and as it comes together with the programmer, you can try it out on your own, play with it, and maybe it doesn't have all the features just yet but it's functioning and you can avoid issues along the way if a problem gets caught by you early on.

7) Do you have a spec?

Have you created this document that exactly outlines in plain simple English what it is you want your software to look like, do, and not do? That way, you're happy with what comes back.

8) Do programmers have quiet working conditions?

This is kind of a productivity thing. You might have heard these different sayings, like if someone gets interrupted one time, that's fifteen minutes of time they need to get back in the zone. Fifteen minutes of productivity lost. And if you have a team of programmers or if you are always interrupting your programmers, they can't get in the zone, so having these quiet working conditions means that sometimes people have to solve their own problems, and they can't always be running back to you for this decision, that decision... Just get something to show me. Instead of talking to me all the time, just get these things done.

9) Do you have the best tools money can buy?

This pretty much means we don't want to go cheap, we don't want to have to use cheap tools that take a programmer's five hours to do something that could be done in one hour. Let's say for example that you're me, and you're creating a backup plugin for WordPress that backs up to Amazon S3. I could spend two weeks coding all this extra stuff to interface with Amazon, or I could just buy a library, plug it in, and even though I might have spent a couple hundred bucks, now I've saved myself weeks of time and even more time in debugging because I just did that shortcut and instead of trying to save a couple hundred bucks to waste a month, I did the smart thing and just used the best tools money can buy.

10) Do you have testers?

This person creates a software product, and they'll test it as much as they can, and they'll miss a few things. Bet you'll use the software, test as much as you can, and you'll miss a few things. So you're going to have to really beat this thing up. You're really going to have to try to break it, try to do the dumbest things. If something asks you for a numeric value, like asks you to type in 300 calories, what if you typed in "FFJQW"? What if you did it as much as possible, to click on stupid stuff, type stupid stuff, and see if it could break it? Your programmer needs to test this, you need to test it, and also have other friends to test the software so that they can break it and it's intuitive.

11) Do new candidates write code during their interview?

Break a piece of code and see what everyone's answers are. Some people's answers will be stupid but some will - the one that comes back the most common - probably be the correct one, so see who answers correctly. In this case he's talking about hiring someone in person, because if you have someone in a closed, controlled environment, where they can't spend a day on it or they can't Google-research it, and then have someone in a room and say "here's a problem." Even something like "how many gas stations are in the US?"

Well, I don't know, but if you're hiring a programmer they'll make a guess. They'll say, "let's see. There are this many states, about this many cities in the US, so if you want to have a gas station that has an average of two cars every ten minutes..." They might figure out logically, "based on this, this, this and this, my best guessed estimate is this number of gas stations in the US." You might ask different programmer questions.

There's a site of Joel's called TechInterview.org, where it asks things that you might not even know what they mean, but you might ask someone, "How do you reverse a link list? How do you reverse a string? How do you sort a string so it doesn't take forever?" And all these different things, where you might have a whiteboard, you might have a programmer, not code, but basically write out their logic, write out the step-by-step sequence, just so you know that they know how to solve these programming problems, they don't just copy and paste someone else's code. This is why you can have the broken line of code, give it to your oDesk worker and see how they fix it.

12) Do you do hallway usability testing?

Could you grab someone just walking by in the hallway, walking by outside and could they figure out how to use your iPhone app or how to use your WordPress plugin within reason? One of his services is called Copilot (I actually used it a couple of weeks ago), which is a remote desktop service, where, let's say your grandma needs help figuring out how to type a Word document, and you could say, "hey, grandma, go to the site Copilot.com and type in this number." And then she types in this number and now you have the ability to see her screen even if you're across the country. There are a lot of ways you can mess this up, a lot of ways you could make it not-user-friendly, so if someone can just come off the street and figure out how to share their desktop, if grandma can figure it out, anyone can figure it out, right?

That's the 12-point Joel test, I'm sure if you just Google "the Joel test" you can find the actual blog post, it's in his book as well, he has a book for Kindle, and it is:

- Do you use source control?
- Can you make a build-in-one-step?
- Do you make daily builds?
- Do you have a bug database?
- Do you fix bugs before writing new code?
- Do you have an up-to-date schedule?
- Do you have the spec?
- Do programmers have quiet working conditions?
- Do you use the best tools money can buy?
- Do you have testers?
- Do new candidates write code during the interview?
- Do you do hallway usability testing?

If you get nothing else from this, it's kind of coming together that if you want to create software you do have to know a couple of things, but they are all things that your competitors don't even want to get to.

If you can figure out the rules of the game and get a simple piece of software created, then you can now have the competitive edge and you can have that course, that part, that book, that comes with the piece of software that is easy to demonstrate, that everyone wants, that you will get extra sales from, that people will pass over your competitors because they have the Facebook course, they have the weight-loss course, but they don't have the one-click solution that you're providing.

This was an extra-extra-long edition of the Robert Plank Show, we talked about getting your software idea for a simple software product, creating your spec and finally it's just time to launch your product, so hopefully you had an idea, mapped out the spec, hired someone, tested it as they went through these milestones and you're happy with version 1.0. It's not the best, not the greatest, but it's simple and it's the first version, hopefully, of many to come from you. Now you launch it. What you need to do, once again, is role model your competitors.

Where do they sell the software? Do they sell it on Clickbank? I was watching a documentary the other day about a guy who sold different musical rap beats. Like if you wanted to make a rap song you could just plug in his beats and create your own unique song. He got sick of teaching other people's software and made his own software and then sells that on Clickbank.

Google your competitors, look for where they're linked to. Are they listed on Download.com? If it's an iPhone app, are they listed on the iPhone app store? Do they have an Android version of their app? Lots of possibilities there, and I don't know what your situation is, but if you can create a software product that maybe installs as a Windows, Mac, Android, iPhone, there's a web-based version... that's pretty cool. I'm just thinking about software like Evernote or Dropbox, where there are so many different ways people can use the software, so many different ways people can discover the software.

As you're Googling your competitors look at things like review sites. Here's an idea if you want to try and sell a lot of books for example: a good way to do that is look at people who review lots of books in that niche, in that genre, and then send those reviewers copies of your book. The same thing with software. Who reviews your competitors' software? Who reviews software in your category? Contact them and give them to review it. Look at how your competitors advertise. Do they just do banner advertising? Ad networks? Or do they do things more creatively? Like do they joint venture with different services or websites?

I'll give you an example. There is a membership plugin that I individually buy licenses to. So if someone buys my membership course and I go out and literally buy copies of the software from the vendor just to bundle it with my course. Can you do something like that? Some kind of licensing situation? With iPhone apps there's all kinds of crazy stuff, you can literally pay different companies to get, say, 2000 installs of your app.

You might pay a dollar per install and then they might go ahead and pay users 20 cents just for installing the app, so then they keep the money but now you can literally buy your way into these rankings on the app store.

There are ways that you can buy advertising in other similar apps. I play an iPhone game called Turf Wars, where basically in order to earn special points in the game you watch a video, and most of the videos are for other competing games.

There's all kinds of cool things but depending on what kind of software you're creating, the platform, the niche, the market, look at how they're all selling it. And there are whole kinds of ways that make your product make you a lot more money. If you can somehow build in an upsell or an ad for your own product... For example, for Backup Creator we added what's called an iFrame. We made it so that whenever someone goes to the settings for Backup Creator, they basically view one of our webpages. And I can make that an ad to upgrade to something else or for some other plugin... that's cool. There are different ways to make your software program go viral, depending on what it is.

To give an example, years ago a friend of mine called Steven Schwartzman had a plugin called WP Magnet, where you could check a box and then it would promote WP Magnet with your affiliate link.

And I put that in Action Popup where you could fill in your affiliate link for Action PopUp and when the pop-up would come up on the screen it would say "Powered by Action PopUp" on the bottom, and someone could click on that, earn an affiliate commission, but now the software builds virally.

Many of these iPhone games, especially, have a social viral component, where you can add someone as your friends, you can earn extra points for tweeting about the app, liking the app, all kinds of fancy stuff, but you can't do any of that until you, personally - yes, I'm talking to you right now - get a simple piece of software created so that you can get that competitive edge against your competitors, it's easier to make more sales, make more money and get noticed and stand out from the crowd. I'm Robert Plank, that's all for me today, make sure you tune in next time for the Robert Plank Show.