The Newbies Guide
To Making Software

You Can Make Money in the Software Business

We all know about Bill Gates – the richest man in the world. He made almost all of his money with software.

As you look at all the fortunes made in software in the last 20 years, you may have wondered how *you* could take advantage of that opportunity.

You can.

Maybe you hesitated because you don't know much about programming.

Or perhaps you haven't gone into the software business because you didn't know where to start, how to market your product, how to sell it.

This is all much easier today than ever before, and all you need is a good software concept, and to know how to get your program built and marketed. We can help.

The broadband revolution of the last few years has made designing, developing, delivering and supporting software easier than ever before – and cheaper, too, since you can skip a whole bunch of steps that used to be expensive (such as packaging, printing manuals, traditional advertising, etc).

We'll walk you through the basics, from concept to completion, helping you develop your product idea, have it made and tested, how and where to sell it, and how to work with the people you'll probably be working with, like programmers.

By the end of this report, you'll be ready to go start making money with software right away!

To start making money with software you need to do four things: Develop requirements (what you want to make and what it will do), code the requirements or have them coded for you, test and improve the code, and sell the software!

Let's think about each step now.

Step 1: How to Get Started: The Idea

The most important thing you need to have is a goal – in this case, what kind of software you want to develop.

An Unlimited Marketplace – It's Not Just Microsoft

There are tens of thousands of software products introduced each year by developers large and small. These run the gamut from every kind of personal entertainment product to utilities, to specialized business software of almost any kind, to applications designed to super-charge or unlock special features of other software or make them work better.

Actually not only is the majority of new software not developed by big companies like Microsoft, almost all real software innovation is from smaller firms, since they have the impetus and flexibility to be adventurous in their thinking and to move quickly without layers of management, funding, and outside stakeholders causing delays and offering opinions.

You may well not make billions with software, but software can be a fine supplement to many kinds of online businesses.

Filling a Need – Or Defining One

Chances are if you are thinking about developing some software products, you already have some idea of who you want to sell to.

While the possibilities are really only limited by your imagination, let's take a moment and think about *major* software buying audiences and *major* categories of software by type that it can make the most sense for a small independent developer to work on.

Major Software Categories

There are a lot of ways to categorize all the kinds of software out there but here are a few convenient, generally agreed-upon MAJOR categories. There are of course dozens, if not hundreds, of specialized categories within them:

- Audio-Visual Software – such as music and video management tools
- Business Software – such as CRM and ERP tools
- Design and Photo Software
- Desktop Enhancements
- Developer Tools – software for making software
- Drivers – software to make software or hardware work
- Home and Education Software
- Internet Software – for getting more out of the Web
- IS/IT Software – for managing enterprise systems
- Utilities – specialized software that typically does one thing such as spyware removal or registry management

Software for Individuals

Individuals buy a lot of software for fun, to keep their computers working well, to organize information, for fun and entertainment and for other reasons. The popular categories include:

- A/V Software – the popularity of digital music and video has created a revolution in this area in the last couple of years
- Design and Photo Software – while businesses use many of these products on the higher end, individuals buy the majority of simpler tools to manage graphics
- Desktop Enhancements – mostly geared towards individuals, they add fun and function to the computer experience
- Drivers – system-resident programs that make other programs and hardware work together, though probably too technical an area to experiment with for a novice development company
- Home and Education Software – by definition for the personal market
- Internet software
- Utilities – specialized small programs that perform a custom function. Among the more popular in recent years would be programs that deal with eliminating spyware and viruses; firewalls and other Internet security functions; organization of large groups of files like photos, music, video clips; file conversion programs to allow users to use different file types with different programs; and much more.

Software for Businesses

Businesses buy a lot of software, to keep their computers working well, to organize information, to manage aspects of their businesses, for security, and many other reasons. With businesses, bear in mind that the transaction methods may be different and that you may need to consider more complex licensing rules, which we'll consider in the sections on selling and supporting your programs.

The popular categories include:

- Business Software – by definition, for businesses of course! We'll discuss some of the major sub-categories below.
- Design and Photo Software – higher-end tools
- Internet software
- IS/IT Software
- Utilities – specialized small programs that perform a custom function. For businesses, the most important utilities would have to do with keeping computers safe (such as firewall, encryption, anti-virus, anti-spyware kinds of programs) and specialized business functions, which will vary by the industry.

While all businesses are different, most businesses have many common business functions they need to perform, and today, there are software programs – and software opportunities – to address these areas, especially for the small and medium business marketplace.

Major business functions that benefit from software include applications, auction tools, finance (including spreadsheets, tax programs, and more), presentations, sales management, program and project management, legal and accounting, add-ins and custom extras for major business programs like MS Office, communication tools and much more.

Before You Start Developing

Even with the hints we'll share, making software can be fairly expensive. Before you get started, have in mind the following:

- What you want the software to do
- What you want the interface to look like
- What kinds of other products are on the market that are similar

- What you plan to spend to have the software developed or "made"
- When you plan to make the software available

Great…now that you know what kind of software you want to make and sell, let's start thinking about how to do that!

Step 2: How to Make Your Program – Or Get It Made For You

The most "vexing" thing for many would-be software sellers is: comprehending how to actually "make" the software. The good news is that in today's world there are a number of good choices for development, and even better, there are thousands (maybe hundreds of thousands) of talented programmers who you can hire for very little money.

Quality programming is now within reach of everyone because of these "developments," and you don't even need to know how to program even slightly. Did you know that Leo Fender, who developed the first practical electric guitar, didn't know how to play a note?

Major Development Languages and Tools

You don't need to know what language is the most appropriate for your particular software and you certainly don't need to know how to do any programming yourself, but you should be aware of the basic popular languages and software tools – if only so you can make intelligent choices when working with programmers and can "talk the talk" of programming even if you can't, and don't need to, walk the walk so to speak.

There are basically 20 major active programming languages in the world today. Not all of them are equally practical for every application but if you're familiar with this list you'll be familiar with almost every general concept of language selection.

While about a third of the following are *very unlikely* to be used in your development, we thought taking a couple of pages to touch on them would be time well spent. The "top 20" list below comes from a November 2005 market study of demand for programmers – meaning all of the following languages have active projects, whether new developments or maintenance.

In order of approximate popularity *by hiring of programmers*, and with a quick comment on typical uses for each, and the relative expense of development in each based on what a typical contract programmer might cost:

Java

Java is an object-oriented language developed by Sun Microsystems in the early 90s and intended as a replacement to C++.

Java is currently more popular than C/C++ mainly because of the percentage of new application development that is Web oriented, Java having been developed especially for online use. Anytime you see a page with "JSP" in its name that is a page being served via Java.

Java is one of four major contenders if you are planning on building online applications as opposed to desktop software or system utilities – the others being PHP and Cold Fusion, as well as for some things, Perl.

(Note that JavaScript has nothing to do with Java, just to confuse everyone a little more!)

Java programmers are readily available and there are more of them every day.

C

Developed in the early 70s by Bell Labs and consistently the most popular microcomputer language for the last 25 years or more, C is powerful, simple, and works well for both system software (for which is was originally designed) and custom application development.

There are some technical limits to C – such as lacking object support – that make other languages including C++ more flexible for certain kinds of applications.

C also "allows" programming that other more modern programming languages will stop when compiling, which means C will let bad routines get through to "final" versions of software if the programmer is not careful.

C programmers are extremely easy to find and accordingly not that expensive.

C++

Addressing many of the limits of C while remaining almost (but not quite) fully compatible with it, C++ adds classes, templates, namespaces and a host of other features.

Also it is easy to find people with these skills and relatively inexpensive.

PHP

PHP is an open source object-oriented language for developing certain applications, Web pages, and in a science-fiction-is-now-reality world, for developing other software itself.

"PHP" technically stands for Personal Home Page which gives an indication of what the developers had in mind, but PHP is now used much more broadly than that, especially given its easy compatibility with every major database language and many other programming languages as well as working on every major operating system including Windows, Unix and the Mac.

Developers are finding more and more cool ways to use PHP every day.

PHP is a hot technology so it is in demand but it is also fairly easy to learn and the licensing is less complex and expensive than many other development tools.

Visual Basic

VB is a major Microsoft update to BASIC, which was the original primary microcomputer language in the 70s. VB adds rapid development, graphical user interfaces, database compatibility and a host of other improvements to BASIC.

Being a Microsoft product ensures its ubiquity and popularity; indeed there are about a dozen specialized versions of VB already in circulation.

The greatest strength and weakness of VB is its deliberate simplicity. This makes developing in VB fairly fast and easy but some of the programming community looks at it with disdain – not that you should care if it meets your needs!

If you see Web pages with "ASP" in the name, those pages are being generated from a VBScript based Web site.

VB programmers are easy to find, and a computer literate person can probably do some development himself in VB without too much trouble.

Perl

Perl was developed by Larry Wall and is often called the "glue of the Web" since almost any CGI script and many other features of Websites are the result of Perl.

Perl is not typically a language that would be used for an entire application but for elements of that application and how it interacts, for example, with Web pages, although (file under it's a small world after all) PHP was originally written in Perl.

Perl has very powerful data handling capabilities and is popular in financial applications for that reason.
Perl programmers are relatively more expensive than C/C++/VB programmers.

C#

"C Sharp" (not "C Pound Sign") is an object-oriented language developed by Microsoft as part of the .Net initiative, which makes it appropriate for online applications and those intended to use standard .Net protocols.

Based on elements of Java and C++, we have not seen that many C# programmers around, so in the short term expect relatively higher rates from them if you decide to use this language for your software.

Python

Python is an open-source language developed in 1990 and very similar in features and functions to Perl.

The last major update to Python was in September 2005.

JavaScript

JavaScipt is an object-based scripting language developed by Nestcape, almost exclusively for Web site development but it has other applications. JavaScript remains extremely popular even though it has not had a major update in a couple of years.

JavaScipt programmers are plentiful!

Delphi

Developed by Borland and originally a Windows-only version of Pascal with object orientation, Delphi has been further extended to work with both UNIX and Microsoft .net.

SAS

Now we have to get a little technical – SAS is a fourth generation database-oriented programming language. In this regard it is very similar to SQL.

Ironically this format of language makes user programming easier than many others (you could start building a database in SAS or SQL with a Dummies book and a couple of hours) but it has limited use. SAS is primarily geared towards mainframes. A Windows version is still supported, by the Mac is not.

Unless you're planning on complex database products you probably won't need SAS.

PL/SQL

The other – and far more popular – fourth gen database-oriented language is SQL, or "Structured Query Language."

Having a long history going back to the mid 70s, SQL has many database applications and is licensed by ANSI/ISO which makes it excellent for use in any situation with those certifications are helpful (like in many industrial situations).

If you are thinking of a relational data component to your software, you'll want to do some more research on SQL.

VB.net

The version of Visual Basic for the .Net platform standard.

Lisp

Lisp – short for "List processing" is the second oldest standard language in real use today, having been developed in 1958.

COBOL

COBOL stands for "common business oriented language" and is a third-generation programming language that remains in use where it was intended – large administrative systems for big companies and government agencies.

You're unlikely to care about COBOL unless you go after that market.

Ada

Developed in the 70s and most recently updated in 1995. Used almost exclusively in military applications and military-like applications elsewhere, such as in flight-control systems.

Pascal

Developed in 1970, Pascal was, among other things, the basis for the original Macintosh operating system.

ABAP

Developed by SAP for use with its own software platform, ABAP has gone through a couple of evolutions and is currently the key to developing SAP applications for the Web via NetWeaver.

If your software is intended to work with SAP, you will need ABAP programmers, who are plentiful but very expensive ($1000/day is not uncommon). If you are not working with SAP, you'll never need to worry about ABAP.

Cold Fusion

Now an Adobe product (coming to them from Macromedia who acquired it from original developers the Allaire brothers), Cold Fusion is a very popular tag-based middleware development language used exclusively for websites and web applications.

Cold Fusion can be made to do a wide variety of things from a software standpoint, and anyone who is comfortable with HTML development will find Cold Fusion pretty easy to learn.

Cold Fusion programmers are plentiful and as the learning curve is very low, pricing should be competitive.

Fortran

Last, and oldest, of the programming languages Fortran was developed in the mid 1950s by IBM for use in scientific computing, where it remains popular even now. Originally a procedural language, an object-oriented version of Fortran was recently released.

Finding Programming Resources

Once you have an idea of what you want your software to do and what it might be "made of" you can start looking for programming resources.

The Internet makes this wonderfully easy.
We recommend starting with one of the major "freelance" Websites, which offer several advantages over the open contract market or local body shops:

- Registration eliminates some low-end resources
- Feedback systems create positive work incentives and make picking resources easier

- Managed conversations and relationships through a third party add security if desired
- Global resources will respond to your posting(s) allowing you to review them at your convenience
- Much more competitive pricing than in most local markets
- Security of payments

There are dozens of "freelance" websites out there, but there are two that pretty much own the space, and we have used both and can recommend them. These are Elance and Guru.com.

Those sites feature many more than just programming resources. For coders exclusively, we recommend two more sites, Scriptlance and Rentacoder.

Elance

Elance (http://www.elance.com) is a well-established eBay style site that attracts a wide variety of freelancers in technical and other areas. All fees are paid by the vendors, so as a client, the whole experience is free.

A large percentage of the programming resources you will find on Elance are "offshore," i.e. in India or elsewhere. While we strongly discourage using offshore resources for work like ghostwriting, we encourage you to work with foreign programming resources if their skills are a true match, there is no language barrier, and you are especially focused on keeping costs down.

Guru.com

A newer and in many ways better version of Elance, Guru.com (http://www.guru.com) asks more of its users both on the vendor and buyer sides, has several systems in place to make fraud less likely, and seems to attract a higher caliber of vendor as well as higher budget and higher quality projects.

If you are looking for developers who can really help spec as well as design and build your software, don't overlook Guru.com, which is also a great place to look for marketing and other resources you might need once you have the software finished.

Rentacoder

Rentacoder, as the name suggests, is a well-established resource for finding highly skilled programmers fast. More geared towards solving specific problems with specific skill sets, we recommend Rentacoder but the novice developer may find it less "user friendly" than some of the other sites. With favorable coverage in *The Wall Street Journal* and elsewhere, we think Rentacoder is a good reliable resource that also, like Guru.com, offers and encourages escrow payments. Rentacoder also offers some good general buying information, much of which can be found here:

http://www.rentacoder.com/RentACoder/SoftwareBuyers/FAQ.asp

Scriptlance

Scriptlance features a confusing bare-bones interface (http://www.scriptlance.com) but offers a programming-only job board that seems to have something of an "insider" feel to it. If the sophistication of the Elance or Guru systems doesn't appeal to you, this might be a great place to programmer shop.

Custom Software Development Companies

There are plenty of companies you can work with to develop your software from beginning to end – this may be a better than sourcing freelancers, in some situations.

If you do a Google search on "software development" several dozen companies in this field will come up, and with them you can go with a concept and they'll make it happen – though much less cost-effectively than solo coders, of course.

Regardless of where you find your resources, you'll want to be careful and specific in how you work with them.

Step 3: Coding and Testing

Working with Programmers

Programmers come in all varieties – not just of experience and skills but of personality types and styles.

Think about the kind of people you want to work with:

- Do you know exactly what you want and plan to give explicit directions?
- Do you want to "copy" something that exists in the marketplace and enhance it? This is common and many programmers are happy to do it, just be careful you don't use code (and they don't use code) illegally
- Do you want idea and concept input?
- Et cetera

Be as explicit as you can be in what you write in your ads, but bear in mind that the ads are publicly viewable so if you have "secret" or proprietary ideas consider how much you want to post; save some for a private discussion.

In terms of qualification, some programming skills require certifications to suggest real competence and others less so. Microsoft certification is usually a good indicator of skill in their platforms, Java and Perl coders often have no such "documentation" for example.

When actually contracting with programmers, be clear about the following:

- Exactly what they will do and when
- Exactly what you will pay and when
- Who provides what – for example, some coders have their own development environments and some will expect you to provide something
- Who owns what – make certain that all work is done on a "work for hire" basis which means you own the final product exclusively
- Make certain that all coding being done for you is being done legally, which means both (a) using legal versions of licensed software and or (b) using open source products and or (c) not directly copying copyrighted code from a third party source, where that issue applies.

In terms of payment, we recommend using an escrow service where that is practical, and using an electronic service in any case. Most freelancers are used to a starting payment and a completion payment, but payment in phases or thirds for longer projects is not uncommon.

Some programmers are entrepreneurial and will suggest some form of "works now pay later" scheme, a form of equity in your product. While in some cases this may make sense for everyone, as a general rule we prefer the clean simplicity of pay as you go.

The Importance of Testing

All software needs to be tested at several points during its development both for functional integrity and also to make certain that the people who will be using the software will be comfortable using it.

There are a lot of specific methods and means of testing – in these pages we can only underscore its basic importance to <u>any</u> software project!

Once you know what to build and how to get it built, and you have a working product, you'll want to start marketing it!

Step 4: Finding Your Target Markets Online

Depending on who is the intended market – all home PC users on the one hand, a specialty business clientele perhaps on the other – you can find customers a number of ways.

First you will have your own contacts. Use them!

Then you can use low-barrier-to-entry sales channels like eBay (see our report on eBay for getting started).

You can and should send your software to relevant online journalists to consider reviewing and writing up – especially if you are focused on a special industry or region where a large percentage of people are likely to read the same things. In the paper industry for example there are only about four or five magazines everyone reads, so a review in one of them is likely to get a tremendous amount of attention.

You can also post free or trial versions of your software on the major websites people use to find such items, and let the market find you.

While there are many places to do this, one of the best was and remains CNET's Download.com (http://www.download.com).

If you visit and look at the download statistics, the numbers can be staggering – there are products with download numbers in the tens of millions! Of course there are also some with numbers in the thousands or less.

The way to get your software onto Download.com?

Upload.com/Download.com

Why, logically enough, Upload.com (http://www.upload.com)!

CNET makes it very easy to participate and you can choose from a combination of four programs to list your software.

1. Listing Packages – multiple levels of listing and posting privileges for your software ranging from free to "basic" and "premium" options. Basically you select a package, pay a monthly or annual fee, and agree to a revenue share with them much like a sales commission.
2. Merchant Services – an enhancement to your listing package(s) this enables ecommerce and other useful things to make your software commercially viable online. As of November 2005 CNET charges a flat 8% for a wide variety of powerful tools.
3. Pay Per Download – borrowing a model you may be familiar with from Google's AdSense and similar programs, this is a way to "compete" though bidding for a Top 5 or Top 10 position, ensuring higher visibility and more downloads.
4. Button Program – a way to link your sites and potentially others' sites directly to your product listings on Download.com.

While there are many competitors to Download.com none get anywhere near the traffic, and CNET has become a major "review" brand unto itself (look at software boxes the next time you're in the store – plenty of them tout high-star ratings from CNET).
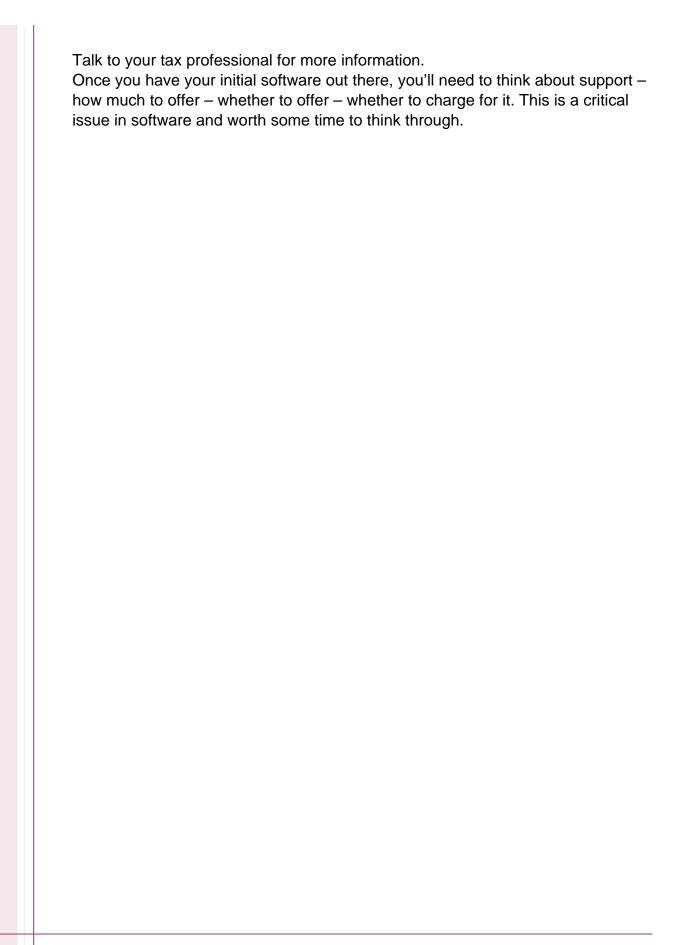
Selling Your Software

If you use a system like Download.com to distribute your software you can handle purchasing directly through them.

If not, or if you have other reasons to use a separate payment system, you can sell software directly, through various third parties, through a PayPal store, on eBay and other auction sites, or by placing ads in publications (online or off) that are likely to reach your intended audience.

If your software works with something else or is an add on to something else, you could explore creating an OEM ("original equipment manufacturer") relationship, where your product would be bundled with others. On a per sale basis the compensation will be much lower than the retail market but volume can easily make this a more profitable route for the right kinds of software.

Regardless of how you choose to sell, remember that software is a product sale, so be sure to take into consideration sales tax, particularly if you are delivering the software to the same state in which you reside or have your business entity registered.

Talk to your tax professional for more information.

Once you have your initial software out there, you'll need to think about support – how much to offer – whether to offer – whether to charge for it. This is a critical issue in software and worth some time to think through.

Supporting Your Software

Even if you don't plan a huge call center in India, most software needs updating. Most software needs to offer some technical support. Think about whether your product will need either or both and then think about the best way to offer them in a way that keeps your product profitable.

We know a database company that built its database platform in Corel Paradox in about 1990.

Paradox is a very old-school system that you may not even have heard of, but it does what it does well. The key to this product is the data, which is updated regularly on a subscription model, not the interface, which has been updated twice in 15 years and may never be updated again. As long as it works, the clients in this case don't care.

In this case, Corel licensing (for the runtime version every buyer of this database needs) is dirt cheap. Paradox is rock solid and pretty much always works, even if its feature set is very old and clunky under the skin.

So this company is in fact selling information, not a "software product" though it is sold and licensed as both.

With more than 200 customers their support staff is a single person who handles phone and email inquiries but is also updating the data continually. The software support costs, by leveraging a solid, unfashionable, database system that only works on DOS/Windows machines are kept exceedingly low.

These particular answers may not work for you, but whatever your market, customer base, and product requirements might be, think about these issues:

- Will the software do forever what it does now, or does it address a changing need?
- If the need changes, does it change yearly, every five years, or some other timeframe?
- How easy is it for a reasonable average computer user to install and use the software?

Have a plan for what you think your market will require, and solve as many problems in advance as you can.
- Thoroughly test and debug your software

- Distribute an early aka "beta" version to some real users and get their feedback for the final release (buttons that make sense to you and your programmer might not make sense to them)
- Use a good installer program that does everything automatically – XP and Mac users now expect "smart" installation, and this is not a corner we recommend cutting
- State what your support policy is and stick with it – if its email only with a 24 hour response, say so, and everyone will be happy!

Creating Your Software "Brand"

Put some time into thinking about what to call your software, and whether you plan to expand on it later. While software that meets a real need is essential, and it has to work, it should have some branding to it – and try to be unique while still getting the idea across.

In the area of system maintenance for example, we have found no less than 20 products with the word "mechanic" in their names, from at least eight publishers. So do you find a way to use the word mechanic? We say no. You'll never standout, but you could try expanding the metaphor – Software Pit Crew for example.

If you have big plans, you may want to work with some branding or marketing resources – you can find plenty of them on Elance and Guru as well.

While there is no way a short report could fully equip you to become the next Bill Gates, we want you to recognize that these days making software is easy to do, easy to market, and you don't really need to know much about programming to do it.

The following are the five most common questions we have been asked recently.

In simple English what are the basic steps for developing software?

There are four essential steps no matter how you look at it:

1. Define requirements – what you want to make and what it will do
2. Develop code to meet those requirements
3. Test the code, debug/fix/enhance it
4. Distribute or deploy the finished software product

Explain what Open Source means?

Open Source refers to a broad-based approach to software development and distribution that began in the 1960s and took the form we think of today starting around 1985, then more emphatically in 1998. As the Free Software Foundation put it 20 years ago Open Source software is "free as in free speech and not free as in free beer," though in many respects elements of many open source platforms are in fact free in the latter economic sense as well.

Open Source means the source code for a piece of software is readily available for free.

As a general proposition, if you can develop your software using Open Source tools and standards instead of commercially licensed ones, you will have lower development and licensing costs, though these relationships can get complicated and in many cases open source software is only financially "free" to individual users, not commercial ones.

There is also a strong anti-commercial – or at least anti BIG commerce – thread through the open source movement.

Pepsi and Coca Cola consider their "secret" recipes the most valuable things they have, but meantime, some wise guys called Open Cola have reverse-engineered their drinks and posted recipes that result in startlingly similar beverages…

Do I need to pay some kind of ongoing license fees for commercial development products I use to make my software?

Generally, no, unless you actually use pieces of someone else's software within your own – such as the runtime version of Corel Paradox we mentioned earlier, which has to be licensed for each instance it is resold.

Licensing the *output* of development tools would be like Microsoft trying to assert rights in Stephen King's next novel because he wrote it using Word!

What do I need to worry about in terms of liability with my software?

Two things mainly.

The software should do what you say it does when used the way it is supposed to be, and you have be very careful in your End User License Agreement (EULA) in listing out what uses are legally acceptable and how much liability your company is willing to accept – i.e. none except the price of the software in the event of a failure.

Consult your legal advisor in helping craft your EULA.