

# *The* **SIX LAWS** *of the* NEW **SOFTWARE**

*continued* ▶



by Dror Eyal

The first wave of software is over, it is doubtful that any one company will capture the market like Microsoft or SAP did. Not because the software they write isn't better or has less functionality, they've simply arrived too late. Most home consumers have all the software they will ever need, and most companies out there already have all the basic technologies they need to successfully compete right now.

I can hear their objections all the way down here, and I agree, your software is better designed, faster, has more features, is more user-friendly and can indeed make seven flavours of coffee. We have something similar, it isn't well designed, it doesn't have half of the features that yours has and no, it doesn't run on Service Orientated Architecture. We did however pay a small fortune for the per-seat licences, we have learnt to use it quite comfortably over the last five years and this is the system that our business runs on.

This view isn't limited to us — Northwestern University economist Robert Gordon, in a 2000 article published in the Journal of Economic Perspectives, argued that "the most important uses of computers were developed more than a decade into the past, not currently."

It's a fairly bleak view to be sure, but one that isn't unique to Mr Gordon. Many business executives are turning away from purchasing new technologies and looking for new ways to use their existing technologies effectively. Not because the new software entering the market

isn't better, but because the functionality that they need already exists in software that was bought years ago. Budgets for software expenditure are dropping and the accountants are starting to question why the software that was essential last year needs an upgrade this year. What this means to the average software developer is that the window of opportunity for selling into the corporate market and to some the degree the home market is getting smaller than ever before.

So does this mean that this is the end for the software industry? Obviously not, we will continue to develop better products, occasionally new technology will get developed and or a new idea will start a trend and software will get developed around it. Software that meets a new need will always be welcome. Who knew that we needed file sharing software before Napster turned the music industry on its ear? Or that social software and blogging tools were essential if your company was to be seen to be on the cutting edge?

No, it isn't the end, but for every tool that revolutionizes the industry and strikes a path into a new territory there are several hundred software companies out there trying to build a better CRM or CMS — the software industry equivalent of the mousetrap. Obviously it would be better if we all developed software that met a new need and created new markets, but just as obviously we can't all develop revolutionary new software. Most of the software being developed right now in studios around the world is trying to find a niche in existing and saturated markets. So how do you build software that stands out and can compete in this new environment? You build a tool based on new generation software laws.

**SINGLE IDEA:** The best way to succeed in the marketplace is to create software that fulfills a specific need. This may seem like an obvious point at first, but if you cannot explain to the end user what the software does in a single sentence it is probably too complex. Your first task is to ask yourself, “What does my product do?”

Most books about developing software start with the system architecture, some sort of UML diagram or a user requirements specification. In the new software, the place to start is to ask yourself what this product does? If you cannot describe this in a sentence to yourself, you'll have a hard time convincing others about the usefulness of the software to their situations. This software is used for writing documents, for easily updating a website without having to know HTML, it allows you to share files with other people. It can do anything, as long as you can describe it in a sentence.

By defining what the product does in a single sentence, you'll not only develop a product that has an explainable need, but your product will fulfill a defined need. If the idea behind your product is too complex you're going to face an uphill battle trying to communicate its purpose. It can be a specialised need, it can be a need that your customer base doesn't know that it has, but if that need can not be condensed into a single sentence you will lose your audience.

Most software vendors start off by developing a product which is stable and has a defined purpose. Once they are in the market they realise that there are competitors in the market with more features. They quickly make a list of the functionality the competitors have and

build it into their product, shifting away from the single idea. The product suddenly becomes complex to use, weighed down with features it requires more processing power, a longer install and more pre-requisites. The market stops using it. It's too complex, it needs too high a spec machine. The company tries to gain back the market share by adding more features.

The point I'm making is don't try to keep up with the Joneses by adding every new feature and functionality into your product. Users want simplicity, they want to know that the product will send and receive email. They don't really care that it reads RSS feeds and can automatically search your favourites folder for updates. This simplicity in functionality is something that Google exploited with its simple interface. While Yahoo!, Excite and the rest of the search engines were developing massive portals, adding functionality and features, Google had two buttons — 'Google Search' and 'I'm Feeling Lucky'.

One of the main reasons why people have been adding functionality into their software is not because the customers want it but due to software's non-physical nature. You see, software isn't a physical good like a shoe, it never wears out. In theory, once you have bought a piece of software to send mail you shouldn't ever need another email client. There's no natural repurchase cycle.

So if you're a software company how do you keep on making money off the same piece of software? The classical way of squeezing extra money out is to upgrade it. This is critical to the economics of most software. Unfortunately the customers don't always want it. How many features in PowerPoint do you really use? By adding extra features you can push up the price of the software, but you also leave a huge market gap for software that fulfills a specific need. Microsoft can afford to this, because they are Microsoft, you can't, and even they have had to compensate for this overshooting. When Office 97 was released unhappy customers forced Microsoft to issue a special program that enabled Office 97 files to be opened in Office 95. The customers preferred the old software because it had fewer features. In the new software the only upgrades that are made are those that are requested by the end users.

**COLLABORATE:** Forget enterprise systems that do everything possible within your field. They're too large, clumsy and require too much development time. Instead, create small discrete software that can collaborate seamlessly with the technology that the end users are currently using.

To develop an enterprise system that provides everything is a losing proposition. Rather, create small software that does not involve great investment in development and products that leverage existing software currently being used by your target audience and do not require armies of programmers. A small studio of three or four programmers should be able to turn out the kind of software I am talking about in about three to six months. By writing small software that doesn't take long to develop and collaborates with existing technologies, you can recoup the costs of that development many times over.

Unsurprisingly, much of the technology that is currently in place in major corporations is actually being used by their staff. The new software states that you need to capitalise on this by developing complementary software. If you develop a word processor, you probably won't win very much market share from MS Word, but what if you build a plug-in that allows you to easily add mathematical equations, or a plug-in that allows you to convert the document to PDF format, or one that allows you to convert a word document into an e-learning course with a single click of a button? Now you have a huge market of MS Word users who need that extra specialised functionality and are willing to pay for it. Most of the large software companies have developed extensive and well documented APIs. Use them!

Even if you are not developing a plug-in for another system, you must have the ability to seamlessly collaborate with the technology that is currently being used. Don't fool yourself: operating systems have collapsed and disappeared without a trace because they couldn't open an Excel spreadsheet. You have to take existing software into account or your software will die a soft death on a shelf somewhere.

In order to survive you must learn the art of collaborating. Don't try to recreate functionality that already exists in software tools that the user is currently using, rather leverage those tools and collaborate with them. The end user is familiar with those tools and they will be missed if you try to recreate them. If your system requires that you use a spreadsheet program, develop ubiquitous hooks that seamlessly allow the user to jump from your software into Excel and back. If your software requires that you have a word processing tool, develop your system as a MS Word plug-in, or take it one step better and develop your system as an MS Office plug-in. By extending the Office Suite your user can choose to either develop in Word or in Excel.

**DISAPPEAR:** No matter what kind of software you are creating, you have to simplify the interface. The greatest software in the world is useless if it is too complex to use. Decrease the interruption of the user experience by reducing the user interface to the point where only the essence is showing.

Buyers have become much more demanding about usability, and by usability they mean that the user experience they are used to should not be interrupted. Add an extra button or two to their user experience and they will forgive you, but try to add a whole new piece of software which interrupts the way that they are used to working and you will lose them.

We need to simplify the interface as much as possible. You want the user to not have to move away from her natural way of doing things and in order to do that your user interface has to disappear, to be unobtrusive, to be low-profile. If your software converts from Excel to PDF, then all it really needs is a single button to do its core function. If it must have management modules of some sort where you manage the settings of these conversions, these should be easily accessible through one or possibly two buttons.

Every Information Architect will tell you that the average person can't remember more than x amount of buttons and y amount of abbreviations. This is all fine and well, but most small software companies don't have the benefit of an Information Architect so I'd like to add another important rule, let's call it the Gates effect. It states that if you are uncertain about any element of the user interface, just think, 'What Would Microsoft Do?'

Microsoft's usability has improved quite a bit since they moved to XP, but more importantly the majority of the people on this planet are working on Windows or Windows clones. It's a subtle effect, but subconsciously in the user's mind when they want to close the interface they will look at the top right hand corner for an small x icon. By developing to a standard which the user is used to, you are not only decreasing time to competency but also decreasing interruptability.

As technologists, we hold all sorts of knowledge that is tacit. We mostly don't realize that we possess it, and we don't realize that our end users don't. We don't realize that most people don't know that you should not repeatedly click the enter button because commands are repeatedly being sent and must be processed. Acknowledging the gap between technologists



and end-users is nothing new, and for years we have been arguing that software must be usable by “my mom.” That the software must provide an ‘out-of-the-box’ experience. The way to do that in the new software is to make the interface disappear, to not interrupt the user experience.

Yes, the user interface is important as it is the core of the user experience but it shouldn’t disrupt the user to the extent where they have to minimize or close their regular software and switch to yours. Software like netviewer, a desktop sharing tool provides hooks that allow the end user to change from her desktop to another without splitting the user experience, has completely disappeared. Zero interruptability.

**SIMPLIFY:** Do I have to go through a course to work with your technology? If so, you are already out of the market. I don’t have time and I already have something similar which I’m used to.

Most people who are going to buy your software are going to have had experience of software. Even my grandfather from Haifa has had email for years now. To the average person, commercial web sites like EBay and Amazon are really sales software with an intuitive interface that the average web user can easily understand and use without going on a course. They know that a piece of software does not have to be difficult to use and doesn’t always require training. My grandfather orders books on Amazon. You can’t send my grandfather on a course, and if he doesn’t immediately grasp your software he won’t use it.

At the company I work for they were looking for a Powerpoint to Flash converter to use in the development of online material. Everyone in the company is proficient in using Powerpoint, while the company’s main method of content delivery via the web is flash. A tool that allowed

people to work in a medium that they were familiar with and could relate to, while at the same time produce material that could be used online was ideal. After considering a number of options which included products from some of the big software companies, they settled on a product that had a single button. No need for extra training, just one ‘Convert to Flash’ button.

The tool was a success because it retained the familiar interface everyone was used to in Powerpoint, it didn’t require anyone to learn anything new, it had an easily conveyed function inscribed on its single button and most importantly it was easy to use. They could have gotten software with extra functionality which let the user edit the individual slides, add voiceovers or other rich media functionality, but the added interfaces would have required some training and getting used to by the authors. This tool had a non-existent learning curve and functionality that didn’t require any training.

**RELEASE:** Start creating and releasing your software now. Think prototypes, iterative releases and user base. Don’t spend your time on writing business plans, designing a website and choosing logos. The competition is moving a lot faster than you may think.

First mover advantage is not as important as various authors make it out to be — Amazon was the third online bookshop — but releasing a piece of software into market where there are dozens of competitors is hard to recover from. While you were designing your website, your competition was releasing beta versions of their software into the market, testing the

waters and refining the prototype based on user requests, blog reviews and actual user experiences.

It is very difficult to get a user to switch from ‘bad’ software to ‘good’ software once they have invested time in mastering the software. In the new software you develop a working prototype which performs the core functions in your system and release it. Release it in alpha, release it in beta, it doesn’t matter, just release it. Let the people who are actually going to be using it give you feedback. If you’ve only spent three months developing it, it is not going to hurt as much to be told that a quarter of the functionality is not going to be used.

It has happened far too many times that software has been designed according to “theoretical” best practice or some product manager’s notion is of what the users want or how they might perform their work. This results in fully developed systems that might match the workflow in an industry but do not take into account the idiosyncrasies and shortcuts that people in any industry develop over time. It’s the idiosyncrasies and shortcuts that will increase your rate of adoption.

It repeatedly amazes me how people will reject a piece of software and it will fall out of favour because it doesn’t have a certain shortcut or the terminology is different to what they are used to. Some people swear by AOL’s Instant Messenger because it has an icon that allows you to express a cheesy grin while Windows Messenger doesn’t. Its minor but it can be the difference between adoption and rejection.

We recently had a company who were interested in our Content Management System to update their website. They had been using a blogging tool to create a daily updated front page and were unhappy that the system we were offering did not have a permalink system, we tried pointing out that due to the nature of our system the URL of the page was equivalent to the permalink. They were not convinced, so we added a small hyperlink at the bottom of

every page which linked to the very same page. The link was called permalink. They have since switched their entire operation to our system.

Developing software by evolution is not something new, the software industry has always had iterative releases based on market demands. The new software just releases it a little earlier, lets the users have more say in the final design. Systems like the Wikipedia work on the same principle.

The Wikipedia is an online encyclopaedia that has no editors, and anyone can contribute to it. Anyone. The old way of thinking would suggest that it would end up being a confusing mess of misspelled words, incorrect entries and spam. As it stands the Wikipedia is one of the most informative knowledge bases online. By releasing early and letting your user base make some of the decisions you are harnessing the same power to develop a better software. One which is more suited to the needs of your end user.

**COMPLY:** Find the relevant international standard in your marketplace and comply. This will enforce good architecture and keep your product on track when your customers will want it to integrate with their legacy software. You know they will want you to integrate.

Most industries these days have standards that facilitate communication between different systems by defining language standards and their application. These standards add a uniform interface to heterogeneous systems, allowing them to connect and share data and functionality without requiring modifications to their internal workings. In essence, these standards

define an agreed upon language so that the system doesn't have to worry about which software the request comes from, only that in response to a kind of request it should return a certain kind of response.

This veil which is placed between the software and other systems it collaborates with erases the incompatibilities among various software companies and allows them to interoperate more or less seamlessly. You don't have to worry what kind of system is on the other side of that request, it could be someone's open source Property Management System, a Point of Sale System developed in India or SAP HR. As long as they all comply with the same standard they are all equal.

For the online learning industry this standard is known as SCORM and what it facilitates amongst many other things is the exchange of information between content vendors and content delivery mechanisms. An educational institution can buy any SCORM compliant content, safe in the knowledge that their content delivery system is SCORM compliant and will be able to deliver the content.

HTML is a standard for the web. All software vendors who develop software that either views, displays or edits HTML comply with the standard, which means that content developed on Dreamweaver will not only be viewable on Internet Explorer but can also be reopened and re-edited by Frontpage. Macromedia, who developed Dreamweaver, doesn't need to have ever tested on Microsoft's product, they both comply.

This interoperability means that whatever your software does it will interoperate with any other software that complies with the same standard. If you don't comply, your users will be locked into a proprietary software system which requires you to develop proprietary tools to do everything. Going back to our HTML example, your browser developed by Microsoft will only display content developed by Microsoft, thus limiting not only the market but also the

usefulness of your product. Would you use a browser that could only display pages created by same software vendor? Why does your software require you to?

The second, often overlooked benefit to complying with international standards is that if you comply then most of the workflow of current systems in the industry, is documented for you. There is less chance that you will have that moment when your stomach rapidly condenses, as you realise that you developed a function while forgetting that crucial element which everyone in that industry knows is essential.

We had that kind of moment recently while developing an application for the travel industry which uses the OTA standard. Somehow the ability to allow a person to book multiple rooms was left out of the system, not a big deal you might think, but the entire core of the system had to be rebuilt at a cost of months. A look through the OTA standard would have shown a list of details that should be sent through when making a booking. Number five on the list? Number of rooms.

# info

---

## ABOUT THE AUTHOR

Chronic multitasker and software devotee Dror Eyal is dedicated to the creation of useful software. He has been developing software for more years than he can remember, both as a software analyst and as a programmer, and has had developed several award winning software packages. Some of those packages have contributed to the mass of software that shouldn't have been built, while others have helped to push the boundaries of the software paradigm.

Over the last couple of years, Dror has developed his theory of what it takes to compete in today's software marketplace based on work in commercial environments as well as the field of software art. It has been referred to by various reviewers as 'guerrilla software techniques', 'how to turn software into a product' and 'the new software'. He prefers the new software.

## DOWNLOAD THIS

This manifesto is available from <http://changethis.com/12.SixLawsSoftware>

## SEND THIS

Click here to pass along a copy of this manifesto to others.

<http://changethis.com/12.SixLawsSoftware/email>

## SUBSCRIBE

Learn about our latest manifestos as soon as they are available. Sign up for our free newsletter and be notified by email. <http://changethis.com/subscribe>


# info

---


## WHAT YOU CAN DO

You are given the unlimited right to print this manifesto and to distribute it electronically (via email, your website, or any other means). You can print out pages and put them in your favorite coffee shop's windows or your doctor's waiting room. You can transcribe the author's words onto the sidewalk, or you can hand out copies to everyone you meet. You may not alter this manifesto in any way, though, and you may not charge for it.

## NAVIGATION & USER TIPS

Move around this manifesto by using your keyboard arrow keys or click on the right arrow ( → ) for the next page and the left arrow ( ← ). To send this by email, just click on .

## HAVING PROBLEMS SAVING TO DISK?

First, make sure you have the latest version of Acrobat Reader 6 which you can download from <http://www.adobe.com/products/acrobat/readstep2.html>. If problems persist, it may be due to your Acrobat Reader settings. To correct the problem (for Windows), a reader, J. Hansen, suggests going to your Acrobat Reader Preferences > Options > Web browser Options. Check the “Display PDF in Browser” option. Then click on Save to Disk .

## KEYBOARD SHORTCUTS

	PC	MAC
Zoom in (Larger view)	[ CTL ] [ + ]	[ ⌘ ] [ + ]
Zoom out	[ CTL ] [ − ]	[ ⌘ ] [ − ]
Full screen/Normal screen view	[ CTL ] [ L ]	[ ⌘ ] [ L ]



# info

---



## BORN ON DATE

This document was created on 24 January 2005 and is based on the best information available at that time. To check for updates, please click here to visit <http://changethis.com/12.SixLawsSoftware>

## COPYRIGHT INFO

The copyright in this work belongs to the author, who is solely responsible for the content. Please direct content feedback or permissions questions to the author: [fox@polka.co.za](mailto:fox@polka.co.za)

This work is licensed under the Creative Commons Attribution–NonCommercial–NoDerivs License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

## ABOUT CHANGETHIS

ChangeThis is a vehicle, not a publisher. We make it easy for big ideas to spread. While the authors we work with are responsible for their own work, they don't necessarily agree with everything available in ChangeThis format. But you knew that already.