The Next Significant Breakthroughs in Software as a Service

Julien Chaumond

Writing a paper on the next breakthroughs in Software as a Service demands some serious creative thinking, because of both the pace at which software as a whole, and SaaS in particular are evolving right now, and of the pace of adoption of these changes by a wide audience. For these reasons, some of the points we'll identify as the next breakthroughs in this paper, may very well be implemented and become mainstream in a few months. We'll first tackle the subject of **quantitative** breakthroughs, and then try to see how **technological** and **qualitative** changes can further change the industry landscape, both in general consumer and business software.

Quantitative Breakthroughs

We should first ask ourselves whether, and when, SaaS is going to *numerically* breakthrough – ie, when it is going to overwhelmingly compete with Traditional software models. Agreed, as Timothy Chou put it, SaaS "has already happened" and it's already a multi-billion industry. However the Nifty Nine, promising as they might be, still don't truly erode the power of traditional software companies. What they undeniably did, though, was to initiate a wind of change that's probably going to change the way big companies think their innovation and production processes.

The first related question to that emergence is whether all software can be delivered as SaaS. This question really makes as little sense as the related question of

whether all software can be open-source. What I mean is that without a prevalent definition of the term “open-source” it isn’t reasonable to even try to answer. If it means “free to get and duplicate”, then of course all software cannot be open-source – for example, game producers wouldn’t give their 100-million-dollars games out for free. If it means that at one point, every piece of software will either contain a part that’s open-source, or be built using significant open-source, then the assertion is probably true. The same reasoning is true for SaaS. If we incept the term as a description of the architecture of a system based on clients that are just terminals, and a central server owned and operated by the software company, then it doesn’t make much sense, because some companies will always need to be in control of their data from start to end, either for privacy issues, or for bandwidth issues – depending on the kind of data processed it might not be smart or even feasible to ship all of it to a server far away, this argument being brought about by those who mock SaaS saying we’re going back to the mainframe architecture of the birth of computers. However, if you mean that most companies are going to move from a software model where projects are monolithic, have very long iteration cycles and are basically sold as one-time goods, to a delivery model where software is upgraded transparently every few days or weeks, where different pieces of software smoothly work together in the same enterprise, and where customers regard support as an essential asset, then you’re likely to be right.

In that sense, and as noted by Rahul Sood and Scott C. Bolick from SAP during the November 27 lecture, the analysts’ prevision are that by 2010, SaaS will be a $20 billion industry and will have gained 50% share in the sector of business applications. So,

at that point, SaaS products will, as a whole, be the size of Oracle, or twice the size of SAP. That's what the next economic breakthrough in SaaS is.

Qualitative and Technological Changes – Business Software

Now let's see what changes in behaviors and technology are likely to happen in the business software industry. First, one point that a lot of the lecturers emphasized throughout the class was the emergence of the extreme importance of **support**. While support was always important, the increase in complexity and specificities of the IT systems makes them more and more difficult to support if you didn't design them. That's why the current predominant scheme of software companies producing the systems, and consulting companies supporting them, is likely to evolve towards software companies selling their own support part, along with their software, for a recurring fee, which is the essence of SaaS. Another trend that's been around for a long time, but that will flourish under the reign of SaaS is the **standardization** and **inter-operability** movement. That is because software that has been designed to be shipped and upgraded incrementally already has to be inter-operable with previous versions of itself, so it is going to be compatible with other industry standards by default. Also, with the fact that software is paid for reccurringly, the cost of switching to a competitor is generally much lower and implies an even stronger need for standardization.

Qualitative and Technological Changes – Consumer Software

Finally we should have a look at the wider audience of general Internet users. Although this sector is not *fundamentally* different from the business software industry, it

is evolving much quicker and probably has a lot less inertia to emergence of new products. For these reasons we have to be very creative in order to find new paths of innovation. As Timothy Chou puts it, "*if you want to know what's next in the software business industry, just look at the consumer Internet*". The delivery model of most typical desktop activities has already begun to change, from office suite to personal calendar or even media player. And indeed, taking the office suite of applications as an example, if I didn't have Microsoft Office installed on my computer I would probably go online and use the exact same functionalities on Google Docs or any of its competitors. I never use any complex functionality besides just simple text typing and formatting tasks, and I guess lots of people are in my case. So, the next time I buy a PC, I'm actually not sure I will pay for Microsoft Office. And more and more types of application follow this trend. Monolithic, permanent software tend to disappear in favor of either small clients that are updated automatically, or browser-accessible applications, in just about any kind of applications. The only applications that are really left in the desktop paradigm environment are those that are really data-intensive and that need to be very responsive, such as single-player 3d games, audio editing, or movie editing.

And, we can actually have an interesting insight on the next breakthroughs in software just by looking at which types of applications are still predominantly desktop-based.

First, media players are still predominantly desktop-based. However they begin to migrate online – see for example lala.com, where you can upload your own music library and then stream it from wherever you are. In that case **mobility** is a key factor: you can listen to your music from your laptop at home, from the office's dektop, and on the go on

your mobile phone, without bothering to synchronize all the devices you use. In the future, as more people get multiple access points to the internet, we expect to see a lot of applications that will simplify the desktop paradigm by avoiding those tedious tasks of **synchronization**. For example, there is a huge demand for such a simple app as synchronizing your passwords, plugins, or bookmarks, across different instances of Firefox. Most of the time, configuration settings – for applications or even for your OS – on all your machines are the same as well, so when you change a setting on one machine you would like to be able to propagate these changes to your other(s) machine(s) as well.

Then, movie editing software is still predominantly desktop-based, for some good reasons: it is most of the time, both very computationally intensive, and very data-intensive. However, in a few cases, these characteristics don't apply. For instance, if you plan on putting your video online on Youtube, you don't really value high resolution that much, and then both the size of the data and the complexity of the computations needed decrease drastically. That's why sites such as jumpcut.com start to appear. Here the key adoption factor – in addition to not needing to buy and install software – is the **simplicity** of the process. Most users aren't video editing gurus, and just want to apply basic effects that work. Another good example is 3d modeling with Google Sketchup incarnating the SaaS model (6 versions in 6 years) as opposed to the big and complex 3d modeling software like Maya or Blender. The **quick iteration time** of SaaS makes their life easier by taking much more easily into account the users' feedback and demand, thus valorizing the so-called "user experience". In that respect applications of the future will present more of a continuous flow of changes to the users, based on what they really ask for. Quoting Brian Behlendorf, "Software is like lettuce", i.e. it's not something that you can

put on the shelf for a long time. Both the Facebook API and the applications that rely on it are perfect examples of that, since deep changes are made to the API every week based on developers' feedback, and deep changes are made to most complex applications every day based on users' feedback. Projects and companies that emphasize this level of **interaction between developers and users** will probably become more and more widespread in the next years.

This brings us to my last point which is the feeling of **trust** of the users. In my opinion this is going to be a key differentiator between software producers in the future. Let me explain what I mean. SaaS is all about software that can really brings a *service* to a user. The user doesn't want the app to just improve his efficiency, but also to change the way he lives by doing things that he doesn't want to bother doing and taking decisions for him depending on complex parameters. An example is travel planning, which is one of the most painful activities today. If the feeling of trust between the user and his application exists, the application can decide which tickets to buy, depending on some preset users' preferences, and can even pay automatically for them. The same is true for any cumbersome activity, such as Christmas shopping. Instead of wasting a day to surf 1,000 different sites for presents, why don't I just write a list of possible presents for all the members of my family, and then let my application decide which ones to submit to me for approval, from any vendor's website, and finally pay for all the things that I chose, in just one step. Obviously the key factor here is the trust of the user. On the first level, this trust depends on the **control** that the user has on the app's decisions. But more deeply, this trust depends on the **transparency** of the software engineering and of

the company that’s behind it, exactly like the trust you put in your own personal assistant when you are lucky enough to have one.

As a conclusion, breakthroughs in SaaS, and in software in general, will involve support, standardization, smart interoperability, automatization, simplicity, ease of use and deployment, and trust. These factors will contribute to the upcoming drastic changes in the Software industry, and in the lives of people involved in writing code, selling it, supporting it, and those using it.