The Cloud, the Store, and Millions of Apps

Anders Fagerjord

The 1.5 million apps for the iPhone can be used for thousands of purposes. Many are cloud-based services, even more are games and simple utilities. The idea of Software as a Service is to have one instance of a program running on a central server, and only one browser to access these programs. From mobile devices it is more effective to access services from apps than from browsers, meaning that every user will need many apps. Moreover, hardware sensors are equally or more important to apps than cloud access. Rather than thinking of apps as software services, we should describe them as actors in a network where developers, users, and Apple's hardware, programming environment and App Store are important parts.

"The Web is Dead" was the slogan that covered the entire front page of *Wired* in August, 2010. Mobile apps provide "simpler, sleeker services that just work," editor Chris Anderson wrote (2010b). Tim O'Reilly responded that "it's the backend that matters," pointing to the fact that popular services like Twitter, Google, or Facebook are run in large server centres which can be reached from web sites and native apps alike (Anderson 2010a). These servers, called the *cloud*, are used by many of the most popular apps. We store our documents and data in the cloud, sometimes sharing it in social networks, sometimes keeping it private. They are available to us from any screen we use, from the little telephone and the mid-sized tablet to the desktop computer and even the 50-inch TV screen. We still call them telephones and TVs, but we use them for the same services. It is the cloud, the backend, that matters, it seems.

Parts of the cloud, or some clouds to be more precise, are *Software as a Service* (SaaS) sites, where users can access computer systems running in central data centres. Instead of installing the software on their own machines, users access the systems through a web page. In my university, I file my travel expenses in one web site, read and write formal correspondence in another, and write drafts of papers with colleagues in Google Docs, which is also a web site. All are accessible from a *thin client*, my Web browser. As the thin client already is in my computer (and virtually all other computers) I only need to keep that one program up to date, and do not need to install and upgrade a lot of others. It is presumably easier for me, and it saves the university's computer department the work with purchasing upgrades and distributing them to all employees. The main system exists in only one instalment in the data centre, and may be updated at any time, without the need to distribute copies to all users.

Most of the time, however, I find that the web sites are slow and generally difficult to use. I often long for "simpler, sleeker services that just work," to borrow Anderson's words once more.

While desktop computers increasingly are used to access remote software via a Web browser, mobile platforms are used without the browser, instead favoring a myriad of native apps. We will untangle this somewhat in the following, and realize early that the *cloud* is a nice, simple metaphor for a complex actor network. A short essay like this can hardly treat one network, let alone several competing networks, so I will focus on apps made for Apple's iOS, running on iPhones and iPads.

To describe a complex network like this, we need to be careful in the use of words, especially as a term may be understood differently by different sets of actors.

For a programmer, an *app* is an abbreviation for any application program. Here, we will use app as in a more restricted sense, which we believe is more in line with everyday language: an app is a small program for a mobile device, downloaded from a central distributor, an app store.

The term *service* is crucial for a book on SaaS. Here, we will have to move away from the everyday understanding of service, and limit it to the use within *Service-oriented Architecture* (SOA) engineering, as defined by The Open Group:

> A service is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports, etc.) and: is self-contained, may be composed of other services, is a 'black box' to consumers of the service. (The Open Group 2013)

It should be added that these services are made available over a computer network. Are apps made of services, being just thin clients, gateways to the clouds? The truth is that some are, but far from all. To understand apps, we need to realize they are actors in a network that we will try to describe in the following.

Coordinating Sensors

Sweeping generalisations about apps are common, but an app can be most anything, from simple a utility to a complex game. Apple's App Store contains map applications, medical diagnostic tools, exercise journals, recipe books and diet journals, banking apps and bus ticket apps, unit converters, calculators and the simple flashlight. The only common aspect seems to be the device: Apps are software applications for mobile devices. Let us then begin the description of the app networks with the iPhone itself.

When the iPhone was introduced, Apple announced it as three new devices combined: An e-mail device, a music player, and a phone (Apple 2007). We may still tend to think of the iPhone as a remediation (Bolter and Grusin 1999) of the telephone, but the technical specification of an iPhone makes it very clear that it is much more. It is a pocket-sized computer with several network connections: GSM telephony, 801.11 Wi-fi, Bluetooth, USB, and in the 2014 models even Near-Field Communication (Apple 2014a).

Input can be given via the high-resolution touch screen, a microphone and camera on both the front and the back the phone, and a few buttons. Output is given through the screen, three loudspeakers, a vibrator, or a powerful LED light, and more loudspeakers and screens can be connected with wires. It is important not to forget the sensors: GPS, proximity sensor, barometer, an accelerometer, a three-way gyroscope for compass and movement, an ambient light sensor, and a fingerprint scanner in some models.

An iPhone app is a small program that uses some of these network connections, input/output and sensors for a purpose the user finds useful or entertaining. An app can make calculations, based on input from the user or the sensors, send and receive data over a network, and output the results to the user, and simultaneously send the results over a network. The most

popular apps are in fact thin clients for Web services, such as Facebook and Google Maps. They use the network extensively, and most calculations are performed on the remote server, "in the cloud". But other popular apps, for example Angry Birds, perform calculations on the iPhone, and use the touch screen and the loudspeakers of the iPhone for interaction. Yet other apps rely on other sensors, such as Sleep Cycle which uses the accelerometer to monitor how the users move while sleeping, or VitalSigns which calculates the pulse and breathing rate of the person in front of the camera by analysing the image. The 2014 Apple Design Award winner Device 6 is an interactive story midway between a game and a book, using only the touch screen and the loudspeakers, while Flashlight uses only the touch screen to switch the LED flash on and off. Sleep Cycle, VitalSigns, Device 6, and Flashlight do not communicate with any server, they run in isolation on the iPhone.

To state that there is no software, only services, would be to narrow down this multitude to only a few kinds of apps. I find Liestøl's perspective more fruitful: That we are moving into the age of sensory media (Liestøl et al. 2012). I believe this transition needs to be studied extensively, but for this essay, we need to move on in our description of the network; from the apps running on the device to the app developers and the environment they work in.

Here be Software

Kittler received some attention for the provocative article title "There is no Software" (Kittler 1992), where he argues the many layers of computer software are only masking the underlying hardware of the computer. In all its technological determinism, the article is mainly a critique of modern computers' Cartesian foundation. Kittler could code in several programming languages, and knew very well that software is the quite tangible result of labour, often tremendous labour. Its layered structure makes this

labour more efficient, and instead of analysing it away, a software studies approach should focus on these different layers, and see how power is distributed throughout.

One does not design an app by combining web services. Apps for iOS can only be made with Apple's XCode programming environment for Macintosh computers. It includes two languages and 70 different frameworks programmers can draw on, including interface buttons and other elements, cloud storage in Apple's server parks, a database system, graphics engines for 2D and 3D development, and interfaces to other parts of iOS, such as notifications, address book, calendar, maps, camera, and photo editing software. These frameworks are similar to services both in being standardised design patterns that developers can rely on through a relatively simple interface, and in being "black boxes", as developers do not need to understand their inner workings.

There are frameworks to support all the three main operations we outlined above; local calculations, access to the sensors , and access to Apple's cloud services. Programmers may earn money by using Apple's frameworks for purchases within the app through App Store's payment service, and for banner ads inside the app. Cloud storage in Apple's iCloud is available through another framework, and sharing via Facebook and Twitter is done via yet another.

XCode is a powerful actor in the network: It regulates what can be done, what is simple to do, and what simply can't be done, and thus has power over its developers. Zittrain uses the iPhone as a prime example of a "tethered" device that can be remotely controlled by its manufacturers, in opposition to a "generative" device that can be made to do anything (Zittrain 2008, chap. 2–3). This division is too simple. Apple can control some aspects of iPhones through software updates, and some of the frameworks and services that developers may use can be remotely controlled. Developers have still found the freedom to create 1.5 million apps available in the US store, which seems quite generative. Apple's

frameworks rarely lock developers in, but they provide roads of less resistance. Large corporations like Facebook operate their own services that their apps use. Smaller developers will have to develop their own services, or they can take the simpler route and use Apple's. Rather than using a dichotomy of generative/tethered, we should follow Kittler's example (if not his conclusions) and study the degrees of freedom available through the software layers.

Software as a Service is often pictured as an architecture that makes programming simple. Apparently, developers do not need to code, just assemble different services, like a child connecting Lego bricks. Programming for iOS programming is a far way from this. Just to create the traditional beginners' "hello world" message requires a list of different files, most of which are unintelligible for a beginner. 500 million iPhones have been sold (Costello 2014), only 350 000 of the owners have registered as developers, and many of these developers (we do not know how many) have never uploaded an app to App Store. One could imagine a phone so easy to program that users would create a flashlight app, not download one, but the iPhone is not that product.

App Store: The Obligatory Passage Point

Just as XCode is the only programming environment, Apple has a monopoly on distribution; developers can't just send apps to their friends. To test a new app on an actual iPhone, the developer must purchase a $99 per year license from Apple (Apple 2014b). The app can be tested on a few devices only, and can only be distributed further via Apple's App Store. This is the main node in the iOS network we are describing, and what Callon (1986) would describe as an "obligatory passage point."

App Store contained close to 1.5 million apps at the end of 2014. It is a place for small businesses, as discussed by Snickars (2012)

and Flueckiger (2012), although major services power the most popular apps (App Annie).

Apple reviews every app before it is allowed into the App Store, and the "App Store Review Guidelines" (Apple 2015) contain 179 rules. Apple controls that apps are reliable, safe, and consistent with the iPhone interface guidelines. Apple also protects its market position, and "apps or metadata that mentions the name of any other mobile platform will be rejected" (rule 3.1). Violence, racism, sex, medical advice and mentions of drug, alcohol, or nicotine use are all strictly governed. This has spurred a debate on censorship, as witnessed by the Wikipedia page "Censorship by Apple" (Wikipedia contributors 2015).

Apple collects a fee for every review. Approved apps can be distributed for free, or the developer can choose to sell it, in which case Apple keeps 30 percent of the revenue. To download an app, users must submit their private Apple ID and password, and charge paid apps to the credit card associated with the account.

Apple is by far the strongest power in these meetings with developers, software, registration fees and credit card companies, these "trials of strength" (Latour 1988). Developers also have power, however. The iPhone had not been the success it is without this tremendous creativity on the part of the developers, as Snickars (2012) has shown. Users on their side judge, one by one, which apps they want to install and use, which is no small power, as the competition for downloads is strong. When users choose which apps to keep, they arbitrate in the trials of strength between the other actors.

Mobility and Ubiquity: Clients and Clouds

We have drawn a quick sketch of the app network, indicating some power relations. We now can return to the question of SaaS. App development is not mashing up services by the inexperienced. Still, apps may connect to Facebook, Twitter, Google's

many services, and personal storage clouds like Evernote or DropBox. This is *ubiquitous computing*: Your data is always with you; the clouds are always over your head. But the idea of the one thin client for all software is lost. Although the mobile phone is powerful it is too slow for the advanced client-side scripts that modern web services use. Mobile telephony networks are also much slower than broadband connections in desktop computers. Efficiency is a major reason to create an app rather than using the telephone's web browser. Apple's Objective-c is more efficient than JavaScript, and gives the developer more control over the many software frameworks and hardware sensors. Another reason is the tiny screen: The browser has a few lines of user interface (known as "chrome") that eat up precious space. Facebook on the Safari browser is shown with the address bar on top and the back button and other controls on the bottom. The Facebook app can use the whole screen, and is at the same time more efficient.

Cloud computing on the phone is not one, but many thin clients. Each of these must be installed and kept up to date, and while the App Store software can notice users of available updates, SaaS's main promise of no installs, no upgrades is lost.

Conclusion

Apps will not kill the Web. While there are some overlaps between web sites and apps, there is a considerable number of apps that never have been, and never will be web services. Anderson's point is that a lot of what is now available as commercial services on the web, such as news and social media, can be delivered more efficiently and reliably on apps tailor-made for each platform. It should not be a surprise that the media industry is what is most visible from Anderson's perspective as an editor of a print magazine. Amateur participation is for Zittrain and others the strength of the Web, and the one aspect that makes it a unique technology.

Amateurs make many apps, but most apps are probably made by professional programmers in their spare time. To create an app is to enter a network of, Apple's programming languages and the Xcode application, Apple's approval service, Apple's App Store, users, and the iPhone itself.

Apps are more than services, they are applications that put the iPhone's computing facilities, network connections, sensors and output devices to use for purposes that do not provoke Apple, and that users find meaningful.

I would like to thank Anders Sundnes Løvlie, Frode Guribye, Kjartan Michalsen, and Johannes M. Ringheim for insightful discussions. The Department of Media and Information Science, University of Bergen kindly lent me the office space where I wrote this text.

Bibliography

Anderson, Chris. 2010a. "The Web Is Dead? A Debate." *Wired,* September 17. Accessed December 19, 2014. http://www.wired.com/magazine/2010/08/ff_webrip_debate/.

Anderson, Chris. 2010b. "The Web is Dead: Long Live The Internet." *Wired,* September 17. Accessed May 27, 2015. http://www.wired.com/2010/08/ff_webrip.

App Annie. "iOS Top App Charts." Accessed April 14, 2015. https://www.appannie.com/apps/ios/top/?_ref=header&device=iphone.

Apple. 2007. "Apple Reinvents the Phone With the Iphone." *Apple Press Info*, January 9. Accessed December 19, 2014. https://www.apple.com/pr/library/2007/01/09-Apple-Reinvents-the-Phone-with-iPhone.html.

Apple. 2014a. "Iphone 6: Technical Specifications." *Apple iPhone.* Accessed December 19, 2014. http://www.apple.com/iphone-6/specs/.

Apple. 2014b. "iOS Developer Program." *Apple Developer.* Accessed December 19, 2014. https://developer.apple.com/programs/ios/.

Apple. 2015. "App Store Review Guidelines." *Apple Developer.* Accessed April 14, 2015. https://developer.apple.com/app-store/review/guidelines/.

Bolter, Jay David, and Richard Grusin. 1999. *Remediation: Understanding New Media*. Cambridge, MA: MIT Press.

Callon, Michel. 1986. "Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay." In *Power, Action and Belief: A New Sociology of Knowledge?,* edited by John Law, 196–223. London: Routledge.

Costello, Sam. 2014. "How Many Iphones Have Been Sold Worldwide?" *about tech*. Accessed December 19, 2014. http://ipod.about.com/od/glossary/f/how-many-iphones-sold.htm.

Flueckiger, Barbara. 2012. "The Iphone Apps: A Digital Culture of Interactivity." In *Moving Data: The Iphone and the Future of Media*, edited by Pelle Snickars and Patrick Vonderau, 171–183. New York: Columbia University Press.

Kittler, Friedrich. 1992. "There is no Software." *Stanford Literature Review* 9 (1): 81–90.

Latour, Bruno. 1988. *The Pasteurization of France*. Cambridge, MA: Harvard University Press.

Liestøl, Gunnar, Anne Doksrød, Šarunas Ledas, and Terje Rasmussen. 2012. "Sensory Media: Multidisciplinary Approaches in Designing a Situated & Mobile Learning Environment for Past Topics." *International Journal of Interactive Mobile Technologies* 6 (3): 24–28.

Snickars, Pelle. 2012. "A Walled Garden Turned Into a Rainforest." In *Moving Data: The Iphone and the Future of Media*, edited by Pelle Snickars, and Patrick Vonderau, New York, NY: Columbia University Press.

The Open Group. 2013. "Using TOGAF to Define and Govern SOAs: Service-Oriented Archtecture Defined." *The SOA Source Book*. Accessed May 27, 2015. https://www.opengroup.org/soa/source-book/togaf/soadef.htm.

Wikipedia contributors, "Censorship by Apple." *Wikipedia: The Free Encyclopedia*. Last modified February 22, 2015, 14:04 CET. Accessed April 14, 2015. http://en.wikipedia.org/w/index.php?title=Censorship_by_Apple&oldid=648325693.

Zittrain, Jonathan L. 2008. *The Future of the Internet and How to Stop it*. New Haven, CT: Yale University Press.