The Digital Ecosystem: An "Inherit" Disruption for Developers?

Jorge Vega, Jon Mikel Zabala-Iturriagagoitia and José Antonio Camúñez Ruiz

1 Preface

First of all, however knowledgeable about programming or technology in general you might be, we would like all our readers to feel comfortable with the following text. I'm not a PhD or university lecturer but Jorge Vega, a developer, so I'm letting you know from the word go that this is not going to be like any of the other chapters you might have read in this book. We'll only be including a few bibliographical references and most of them will be links. We'll do our best to use language similar to that in the rest of the book even though academic terms and developers' jargon have little in common. We'd like this chapter to be easy reading, written in a straightforward style, where we can speak directly to the reader, like we would at a talk or debate or just a friendly chat in a coffee shop.

As you may have noted this different style is already present in the title of the chapter. We have included the word "inherit" in it, in a nod to the language of developers and to the meaning of the word in relation to the Darwinian jargon of inherited characters. The aim of this chapter is to show developers' vision of the dynamics that occur in the Internet ecosystem. Up to this point, you've read about

J. Vega (✉)
Senior Front-End Developer, Bilbao, Spain
e-mail: jvega300@gmail.com

J.M. Zabala-Iturriagagoitia
Deusto Business School, University of Deusto, Donostia-San Sebastian, Spain
e-mail: jmzabala@deusto.es

J.A.C. Ruiz
Department of Applied Economics I, University of Seville, Seville, Spain
e-mail: camunez@us.es

M. Gómez-Uranga et al. (eds.), *Dynamics of Big Internet Industry Groups and Future Trends*, DOI 10.1007/978-3-319-31147-0_5

the characteristics of the big business groups that dominate said ecosystem. But in this chapter, we'd like to take a look at the other side of the coin and analyse what is happening in the world of developers, what they do and how the Internet is seen from the thousands of start-ups that form it. Do developers think that there is a disruption that is the same or equivalent to what the big Internet companies create in other scopes? That's the question we will try to answer. We're fully aware that hundreds of thousands of developers and start-ups form this new ecology which is constantly expanding, so this should all be taken as the view of just one developer and may well not agree with that of many other digital developers or entrepreneurs.

Like all good computer freaks (and proud of it!) we developers who are now over 35 have been tremendously influenced by the great movie sagas and science fiction movies. And like all good trilogies, the chapter will start with a story, which we'll gradually set in its proper context. The second part of the chapter includes Sects. 2 and 3. Section 2 pays special attention to the evolution of programming languages over the last decades because it is key to understanding the evolution of developers' logic. It also gives an overview of the different developer profiles. Section 3 covers developers' growing empowerment in the Internet ecosystem and the increasingly important role they play in it. Section 4 shows us a particular vision of the possible future scenarios that we might expect, from the perspective of developers and start-ups and their role in the Internet ecosystem as a whole. Section 5 centres on the opportunities that might arise from the emerging Big Data context and the role that developers may play in it. The chapter ends by analysing the weaknesses of the epigenetic (i.e. EED) approximation for studying developer-related dynamics in the previous scenarios and raises the possibility of their being studied from a quantum approach.

2 What Being a Developer Means

I went to a talk by Carlos Barrabés in San Sebastian about 15 years ago, during which he explained his experience with his online shop.[1] He talked about how he had hired some consultants and when he told them about his online business idea, they started picking it apart, and tried to change the direction of the project he had in mind. Finally, he got so fed up with snags and not being able to put his idea into practice that he decided to get rid of them and hire a developer, with whom he started his online business. And that very business has, with time, become a global success.

[1]Carlos Barrabés is an entrepreneur who was born in Benasque (Spain), a town in the middle of the Pyrenees in 1970. Barrabés ran a small mountain gear shop and set up the first online shop around 1994 (i.e. sale via the Internet) in Spain and one of the first in the world. See: http://www.barrabes.biz/.

As I listened to him, the question I asked myself, and would now like to ask you, is: who was the developer in that story? The first consultants, the developer (i.e. computer engineer) who set up the project that Barrabés had in mind, or Carlos himself? Besides laughing at the insults the speaker launched at computer engineers, the only thing I remember about that talk was coming out with the feeling that everybody is "a bit of" a developer. More than anything else, I remember how the speaker highlighted that everything was possible in the digital world and the most important thing is ultimately for each person to be engaged enough with the organisation they belong to so that they give their best to achieve an overall final result that satisfies all the parties concerned. Somewhat like what first year Economics or Business Administration students are taught in the lectures on the optimising individual in microeconomics.

To start with, I would like to get a few things straight and clarify what a developer is and isn't by pointing out the differences between developers and programmers. Programmers are the ones who write codes. Full stop. They don't give a care about the purpose of their code except for its function declaration. However, developer is a broader term. A developer is anyone who has a professional profile that requires defining and creating (i.e. modelling) a product or service, and doesn't have to be a programmer to do so. I realise that this idea is completely subjective and not everyone may agree, but…. I guess that's why people discuss these things, isn't it? That is the great advantage of science; you can have informed debate where as many points of view as there are disciplines converge (or collide). In any case, that term referring to disciplines is being questioned due to the prevailing interdisciplinary nature of today's world and the Internet ecosystem is no exception to this.

Here I am in 2015 writing this on a laptop that has an operating system (OS) with many programmes and routines where thousands of lines of code are processed in a second. Actually, this chapter is being written gradually using different platforms and devices that are synchronised. So depending on where I am, or how inspired I feel, and many other factors that I'm not going to mention here, there are days I write from my laptop and other days from my tablet and, when on the underground, from my smartphone. Sometimes, I even write on a paper napkin in a café and later include that digitally in the text or talking directly to any of the previous devices, depending on how illegible or legible the napkin might be when checking it at home. For instance, I used two editing software for this text, an online one and a desktop one. Both allow me to edit the same text without losing any information from various Android devices and my Mac, all 100 % compatible. As I mentioned, all these programmes or apps run on different hardware but thanks to online services, I can create the illusion that all my data magically appear everywhere. What's more, if the apps can run in a browser, we see that the information is available from almost anywhere (e.g. a terminal, a browser, a mobile OS). I never knew writing could be so difficult. When I was first asked to contribute to this book, I thought writing a chapter couldn't be too different from writing lines of code. One of the conclusions I've reached is writing code and writing a book or scientific articles are just about as different as developers and programmers.

Needless to say, the Internet is just one of the current areas of technological development where developers play the role they always did. Today's gurus might have forgotten that it took a group of developers to make an alarm clock just 25 years ago. Developers, however, don't just focus on developing code because hardware is essential. As Quintero (2015) pointed out, investment in hardware has grown exponentially since 2010. I imagine that with the connection leap from our machines to our communication terminal (i.e. the Internet of Things), this investment will get even bigger because a lot of the equipment we now have in the home will have to be updated.

For instance, in business information management, we are shifting from Enterprise Resource Planning (ERP)-based environments, which would be somewhat like business resources planning systems (i.e. automate the company's processes in any of its departments via different modules) to cloud Software as a Service (SaaS) systems, where the software is online and we connect to its management capabilities without having to use ultra-expensive machines to do so. That's how new business models arise in code generation or for companies that develop these technologies that didn't exist before such as[2]:

- **Infrastructure as a Service (IaaS)**: In this case, processing capacity (CPU) and storage are contracted. Our own applications can be deployed in this environment if we choose not to install them in our company to avoid costs or due to lack of knowledge. Servers manage them and all expenses become variable costs for customers, so that they only pay for what they use. Examples of IaaS could be Amazon's Elastic Compute Cloud (EC2) Microsoft's Azure.
- **Platform as a Service (PaaS)**: Here, an applications server (where our applications will run) and a database are provided so that we can install the applications and run them. You usually have to observe a series of restrictions to develop apps for a server, concerning programming languages for instance. An example of PaaS could be the Google App Engine.
- **Software as a Service (SaaS)**: This is what is commonly called "the cloud". It's an application for end users who pay a rent for the use of the software stored in said application. This means that users don't have to buy software, install it, set it up and maintain it, since all this is done these by the SaaS. Some examples of SaaS could be Google Docs, Zoho or Office365.

Yes! Whoever you are, reading this chapter, no matter what they have told you… there is not really a cloud! That image of something they say is called "the cloud" is nothing more than a room full of machines. This is nothing new. It's an old concept.

I often meet people (many of them are university lecturers) who define a group of services as something tangible (I mean "the cloud" concept). That really bothers me. Honestly, the people who should be educating Internet users and people in

[2]See: http://www.xatakaon.com/almacenamiento-en-la-nube/cuando-hablamos-de-la-nube-que-es-iaas-paas-saas (last access 11th October 2015).

general are pushing an empty misleading concept. There is one concept that has been essential my whole life, and that is learning. This is true at least for me and most of my colleagues. As I was searching for a little bibliography for this paragraph, I ran into the "cerebral plasticity" concept, which is nerve cells' ability to anatomically and functionally regenerate as a result of environmental stimuli. Learning is all about achieving the best functional adaptation to the environment. That's what epigenetics is all about, isn't it?

Let's take a short pause here. The learning process must necessarily be connected to teaching, for us to understand it easily. When an apps critic or a technology journal talks about the cloud, or online services, when a salesperson is going to sell services to a company, both parties (teacher and student, salesperson and customer) should do our best to understand exactly what we are selling and hiring. And that's what I mean when I said you can't sell empty concepts like "the cloud". They have to be correctly explained and neither party should be misled, especially in business relationships where resources are invested and expenses and profits are everyday matters.

So, why do we give things such ridiculous names instead of boosting and conveying existing knowledge? It's as if we said that five-year-olds are naughty, annoying, etc. when we know that it's simply not true. Nobody can stick a child in a category just because of their being a child. If we extrapolate this idea to users, the same goes for services. Not all of them can be put in the "cloud" category. There are different services, adapted to meet certain needs, although generic services do exist. That kind of scalability is what online services provide. And I say "online services" because they may vary widely. I'm not using the term "cloud" as if it were just one service.

So far, I have tried to explain how, in general, we have shifted from closed environments to being exposed to a vast information network to help readers understand the delocalisation of typical hardware in big companies (e.g. Microsoft Windows) has created new business models for the entire world. And in turn, we have somehow stopped wanting to understand how things work and what they are for. It is true that our field is moving faster than Fernando Alonso's McLaren (2015) but, as digital citizens, we should at least try to explain the advances in the most accurate, friendly and interesting way possible. And we should also attempt to understand what their purpose is or what they can really be used for, both from a personal as well as a business perspective. But as we said at the start of the chapter, our aim is to show how developers view the epigenetics carried out by the big business groups in the Internet ecosystem. From the time I was asked to take part in this book, epigenetic dynamics have been on my mind. Despite seeing indicators that lead me to believe that the world of developers will be affected by the epigenetic model on the one hand, the lack of truly disruptive elements makes me wonder if the analytical framework provided by EED is really the right way to analyse the reality of developers. On the contrary, an alternative model might be needed to understand the other side of the coin in the digital ecosystem.

This inner struggle started out with the personal experience I have had with Marketing and my vision of it. I don't think an actual disruption exists at the

development level, but rather that the reality of developers follows evolutionary logic more similar to Darwin's (see Chapter "Introducing an Epigenetic Approach for the Study of Internet Industry Groups" in this book by Gómez Uranga, et al.). The reality of most developers is pretty far from the aggressive marketing used for products and services and counterposes the benefits of development itself, which is the Internet companies' progress and advance. I think where there is fierce disruption is at the economic, social, institutional and business level but not so much at the development level. So, these companies, and therefore their marketing, are nurtured and grow thanks to the hours devoted to development by many people who are still working with systems that are more conventional than the final product. This sounds like the description of a big company although it isn't so different from a start-up, except that its investments are infinitely lower in every sense except for development and that is where good developer performance can "change the world".[3] So, I do agree that the Internet ecosystem is a two-sided coin. On one side, we have the big multinationals in the ecosystem, which have been referred to as GAFA in this book (see Chapter "Epigenetic Economics Dynamics in the Internet Ecosystem" by Zabala-Iturriagagoitia et al.). And for that side of the coin to exist, the other side where the thousands of start-ups and millions of developers are found needs to be firmly established.[4] However, I'm not so sure that the analytical model we can use to explain the dynamics of some (i.e. GAFA), is the right one to explain the evolutionary dynamics of the others (i.e. developers).

3 Developers: Classification and Evolution

I didn't study biology. Or economics or engineering. Therefore, I have ideas about what I have read in the authors' conversations that may not agree with theirs. And I accept that anybody should feel free to criticise and disagree with the views in this chapter as well. As Kuhn (1962) pointed out, scientific paradigms are often challenged by new models, leading to scientific breakthroughs. That's why science is distinctive for fostering widespread debate.

I understand an epigenetic dynamic to be evolution that occurs in response to the environment, particularly when it behaves like a high velocity environment where sudden disruptive changes take place. We are what we are thanks to our grandparents' diet, the pollution where they lived, the jobs they had and the same

[3]All you have to do is see this clip from "Silicon Valley", a series I highly recommend. View it at: https://www.youtube.com/watch?v=J-GVd_HLlps.

[4]According to IDC's 2014 Worldwide Software Developer and ICT-Skilled Worker Estimates (IDC 2013), the total number of software developers in the world is about 18.5 million. Around 11 million of those would be professional developers, and 7.5 million would be hobbyists (i.e. coders building software in their spare time for their personal entertainment, student developers, contributors to free and open-source software projects, and unfunded entrepreneurs).

goes for our parents; genetic legacy after genetic legacy shared with other humans and the environment. Bearing this concept in mind and getting back to our subject, the first thing we should do is take a look at the history of computer code. Understand what it was for, the environment where it worked, its aims and above all, what developers were doing at the time. In other words, we need to know the developers' original DNA.[5]

Looking back, we find that (programming) languages were originally used for a series of very specific activities with a minimum diffusion capacity in a very highly controlled ecosystem. As time passed, technological progress has made the ecosystem bigger, having reached today's global Internet. Several years ago, every developer had to learn many languages, which ranged from the simplest to the most complex. I think we all began programming in C. Or at least we had to go through that stage to learn the basics. A terminal…. and then we went right on to print code on a screen! Simple. From C, we moved on to C++, Turbo C, etc. They were all evolutions based on the same (relatively simple) pattern. Watch out, though. If you are a developer and didn't start with the above, I think it's a little dangerous because you lack the basics, the essence of the simple. There is nothing like seeing your first code printed on a terminal.

Taking into account how these languages evolved (Table 1), we can see that some "ghettoes" were created where languages were being embedded. Either because of the need to adapt the hardware to the language or because the language had to evolve thanks to the progress made in the hardware.

From the above list, we see how the first languages centred on computation in closed environments with very limited objectives. We should bear in mind that computational power was established thanks to the development of smaller transistors and chips, so we can put more in the same physical space. Moore's Law has been in force for no less than 50 years. Imagine an upward curve from 0 to 100 where we easily understand that speed has undergone an exponential increase. Going back to languages, the newest ones are oriented, and later adapted, to work on open systems (i.e. the Internet) and more specifically, on clients and servers.

We are witnessing a key moment in history, although we often don't see it that way. Mobility is changing everything. However, it requires the device concerned to have a good battery that won't run down because of the processor, without overlooking continuous connection to the Internet, which is causing some changes on servers' infrastructures (see Chapter "4G Technology: The Role of Telecom Carriers" in this book by Araujo and Urizar). This calls for a language that is light and can be extended to any hardware support.

I'm going to explain what the client/server parts are, for those readers who are not familiar with this. It's important to understand this point to be able to continue from this point and grasp what you are actually doing when you are holding

[5]This stage would be equivalent to what Gómez Uranga et al. identify in Chapter "Introducing an Epigenetic Approach for the Study of Internet Industry Groups" in this book, where they introduce the three-stage methodological approach of the EED, as the "Analysis of the environment and identification of the genomic instructions which are transmitted over time".

Table 1 History and influence of programming languages

Language	Named after	Year	Creator	General purpose	Primary uses	Used by
Fortran	The IBM Mathematical Formula Translating System	1957	John Backus (IBM)	High-level. For numeric and scientific computing (as an alternative to assembly language). Oldest programming language (still used today)	Supercomputing applications (e.g. weather and climate modelling, animal and plant breeding, computational science functions)	NASA
Lisp	List processor	1958	John McCarthy (MIT)	High-level. For mathematical notation. Several new computer science topics (e.g. tree data structures, automatic storage management, dynamic typing, and self-hosting compilers)	Algorithmic language development, air defense systems	Etsy uses Clojure, a dialect of Lisp[a]
Cobol	Common Business-Oriented Language	1959	Short Range Committee (SRC)	High-level. Primarily for business computing. First programming language to be mandated by the US Department of Defense	Business software (e.g. finance and administration systems), but also banks, insurance agencies, governments, and military agencies	Credit cards, ATMs
BASIC	Beginner's All-purpose Symbolic Instruction Code	1964	John George Kenny and Thomas Eugene Kurtz of Dartmouth (SRC)	High-level. Designed for simplicity. Popularity exploded in the mid-1970s with home computers. Early computer games were often written in Basic, including Mike Mayfield's Star Trek	Home computers, simple games, programmes, utilities	Microsoft's Altair BASIC, Apple II

(continued)

Table 1 (continued)

Language	Named after	Year	Creator	General purpose	Primary uses	Used by
Pascal	After French mathematician/physicist Blaise Pascal	1970	Niklaus Wirth	High-level. For teaching structured programming and data structuring. Commercial versions widely used throughout the 1980s	Teaching programming Object Pascal, a derivative, is also commonly used for Windows application development	Apple Lisa (1983), Skype
C	Based on an earlier language called "B"	1972	Dennis Ritchie of Bell Labs	Low-level. Created for Unix systems. Currently the world's most popular programming language. Many leading languages are derivatives, including C#, Java, JavaScript, Perl, PHP, and Python	Cross-platform programming, system programming, Unix programming, computer game development	Unix
Ada	After Ada Lovelace, inventor of the first programming language	1980	Jean Ichbiah	High-level. Derived from Pascal. Contracted by the US Department of Defense in 1977 for developing large software systems	US Department of Defense, banking, manufacturing, transportation, commercial aviation	NSTAR, Reuters, NASA, subways worldwide
C++	Formerly "C with Classes"; ++ is the increment operator in C	1983	Bjarne Stroustrup	Intermediate-level, object-oriented. An extension of C, with enhancements such as classes, virtual functions, and templates	Commercial application development, embedded software, server/client applications, video games	Adobe, Google Chrome, Mozilla Firefox, Microsoft Internet Explorer

(continued)

Table 1 (continued)

Language	Named after	Year	Creator	General purpose	Primary uses	Used by
Objective—C	Object-oriented extension of C	1983	Brad Cox and Tom Love of Stepstone	High-level. Expanded on C, adding message-passing functionality based on Smalltalk language	Apple programming	Apple's OS X and iOS operating systems
Perl	–	1987	Larry Wall of Unisys	High-level. Created for report processing on Unix systems. Today it's known for high power and versatility	Computer-generated imagery, database applications, system administration, network programming, graphics programming	IMDb, Amazon, Priceline, Ticketmaster
Python	For British comedy troupe Monty Python	1991	Guido Van Rossum of CWI	High-level. Created to support a variety of programming styles and be fun to use	Web application, software development, information security	Google, Yahoo, Spotify
Ruby	The birthstone of one of the creator's collaborator	1993	Yukihiro Matsumoto	High-level. A teaching language influence by Perl, Ada, Lisp, Smalltalk, etc. Designed for productive and enjoyable programming	Web application development, Ruby on Rails[b]	Twitter, Hulu, Groupon
Java	For the amount of coffee consumed while developing the language	1995	James Gosling of Microsystems	High-level. Made for an interactive TV project. Cross-platform functionality. Second most popular language (behind C)	Network programming, web application development, software development, Graphical User Interface development	Android OS/apps

(continued)

Table 1 (continued)

Language	Named after	Year	Creator	General purpose	Primary uses	Used by
PHP	Personal Home Page	1995	Rasmus Lerdorf	Open-source. For building dynamic web pages. Most widely used open-source software by enterprises	Building/maintaining dynamic web pages, server-side development	Facebook, Wikipedia, Digg, WordPress, Joomla
JavaScript	Final choice after "Mocha" and "LiveScript"	1995	Brendan Eich of Netscape	High-level. Created to extend web page functionality. Dynamic web pages use for form submission/validation, interactivity, animations, user activity tracking, etc.	Dynamic web development, PDF documents, web browsers, desktop widgets	Gmail, Adobe Photoshop, Mozilla Firefox

Source Veracode (2013)

[a]Etsy is an online marketplace for artists, designers and crafters, so that peer-to-peer e-commerce relationships focused on handmade or vintage items and supplies, as well as unique factory-manufactured items can be exchanged. Clojure is a general-purpose programming language with an emphasis on functional programming. It runs on the Java Virtual Machine, Common Language Runtime and JavaScript engines

[b]Ruby on Rails is a web application framework running on the Ruby programming language under MIT license

your terminal and you run actions on the applications. Besides, if something can be explained and someone learns, as we said earlier, so much the better, so here I will put those ideas into practice.

Readers who already know this can skip this paragraph. Imagine how often we open a browser and key in our favourite newspaper. Services represent the client, or even better, the programming of n elements, which enables the browser to use it as the language to communicate with whatever there is on the Internet. All the programming runs on my machine and the device uses software in the language that best suits to it so as to process the information got from the Internet. The server part covers processes that run on several machines across the world which work to supply us the information that we, as clients, ask for. We've all seen how the computer slows down when we load a webpage. That is because the information being interpreted as it reaches us. It's actually being "translated, understood and displayed in a comprehensible way", thanks to the code that has to run. When the computer slows down, that means there are problems or errors, and depending on the operating system, this will determine how angry we will get.

Imagine Mariano Rajoy, Pedro Sánchez, Albert Rivera and Pablo Iglesias in a meeting, or any other politician from your country. Since they are incapable of understanding each other, they have a translator on hand. They all speak Spanish but don't understand each other so suppose a question like "How is Spain doing?" comes up, which would be the information needed from the server part, the interpreter. Depending on its programming, it will make these deliver a response, according to how the information has been interpreted. I'll leave their responses to your imagination.

These two concepts, client and server, are necessary to understand the next section. As developers, please allow me to focus on the developers' ecosystem from my perspective as front developer. New specific jobs have been created which depend on the working model we embed in our online application (I say application so as not to use webpage or online business) or the language to execute on the client/server part.

If we take a look, there are currently many development frameworks (we understand frameworks as utilities) to develop a web application, for instance, what we know as a single-page application (SPA). Backbone, AngularJS, Ember or React, are all based on Javascript, a language from 1995 (see Table 1), but these evolutions enable a native browser format for their interpretation. However, the most important point is their ability to work on the "Model view controller" where:

- **The Model**: shows the information with which the system operates. It manages all access to such information, both queries and updates, also implementing access privileges that have been described in the application specifications (i.e. business logic). It sends to the "view" the part of the information that is requested for viewing at each moment (usually by a user). Information access or manipulation requests reach the "model" via the "controller".

- **The Controller**: responds to events (usually user actions) and sends commands to the "model" when some request is made concerning the information (e.g. edit a document or an entry on a database). It can also send commands to its associated "view" if a change in the way the model is displayed is requested (e.g. displacement or scroll for a document or the different entries in the database). So we could say the "controller" acts as an intermediary between the "view" and the "model" (see Middleware).
- **The View**: generates a presentation based on changes in the "model" (information and business logic) in a format that can interact (normally the user interface). It therefore asks said "model" for the information it should show as output.

When we load information on the browser, we wait to obtain a result. We're not aware of this action and are outside the application structure that runs internally both on the client/server. Increasingly, lighter workloads for the client and greater access to the processes that run on the server are being searched for. Since the latter are online, they may have higher level of computation. Although, as I said, for us (as users) it is invisible. We are only aware of our webpage's load time.

A further step in recent years, the arrival of Node.js,[6] is changing the very architecture of the applications, which is always done under the model view controller. Node.js enables you to have Javascript in the server part and not only in the client when this was and has been exclusively used in the client part. Disruptive? It may be, but it's applying a language in a familiar environment, which changes many things but is perhaps just one more step rather than a huge leap. The change is taking place gradually, the runtime environments have not been made obsolete by the new ones.

Having seen the intangible part of the code, I'd like to make a few remarks about programmers (i.e. who are not developers). In order to offer a possible classification of the many programmer profiles, I'd like to reflect on the how they are characterised (besides the fact of being fantastic technicians). This classification and the characterisation that goes with it may sound ironic, but if you have worked with code I'm sure you can identify the following descriptions:

- **The "Benito" programmer (From the TV series Benito y Compañia)**[7]: Maybe 80 % (Here's to Pareto!!!) of the programmers for SMEs and big companies fit in this category. They focus on debugging and solving problems that require speedy solutions, without needing a precise code. These programmers are vital to many semi-public companies and you can usually tell who they are when they ask the question: "well, it works, doesn't it?".

[6]Node.js is an open-source, cross-platform runtime environment for developing server-side web applications.

[7]Also known as the Benito Lopera Perrote (for Spanish readers) or the Mac Gyver (for international readers) of programming.

See: http://www.imdb.com/character/ch0169507/?ref_=nm_flmg_act_11 (last access 12th October 2015).

See: http://www.imdb.com/title/tt0088559/ (last access 12th October 2015).

- **The "Perfectionist" programmer**: These have a twin type in the design world. They can devote hours and hours, which ups the project budget. And when they deliver the project, you'd better not touch anything. They need to be kept close by on a short leash. Quoting the French writer Alphonse Karr (1808–1890), we could say this type of programmer "makes everything around him perfect, but does not strive to perfect himself". Also known as the Sheldon Cooper of programming.[8]
- **"By and by" programmer**: These are good programmers but are worn out from years of work and need to read a couple of sports dailies before they get down to doing anything job-related. They always say they'll have everything back to you "by and by". This doesn't mean they're not good at development, just that they don't feel like doing it when you need it. And then they are capable of creating all you need in record time. Also known as the Usain Bolts.[9]
- **"Technophile" programmer**: They always have the latest thing, on their cell phones, watches, etc., and know thousands of theory concepts in depth but are incapable of developing anything on their own. They are often seen as flies that flit around other technicians, distracting them and proposing ideas without knowing exactly what is being developed. Also known as the Antonio Recio of the wholesalers (for our Spanish readers)[10] or Milhouse Van Houten (the Simpsons).[11]

Of course, there are many other profiles, and hybrids of the previous ones, but I'm sure you've run into some of them if you've been working with programmers for a while. To tell you the truth, technicians (i.e. programmers) have also learned to evolve. You no longer have to be a Linux freak to be a good programmer. Actually, I really think that computer freaks are evolving towards a greater social awareness. Maybe because they understand the capacity of what they are developing or what they could actually achieve with their tools. In recent years, "hacktivists" are playing a bigger role and are more important thanks to the groups formed across the world.

I'd like to give some visibility to hacktivism (the term comes from the combination of hacking and activism) in just a couple of paragraphs. I won't go into any in-depth explanations since that is not the aim of this chapter. This socio-digital awareness is an indicator of what used to be the source of continuous jokes about their having no sex life but is now becoming significant. These movements are based on the capacity to join together digitally, and plan digital or social actions. Whether you agree with their activity or not is up to you, I only intend to show how developer groups have evolved. Epigenetics? I don't think we can deny that political and social concerns drive these groups to organise and carry out activities. But maybe what is most important is that, thanks to the developers in these

[8]See: http://www.imdb.com/character/ch0064640/ (last access 12th October 2015).

[9]See: http://usainbolt.com/bio/ (last access 12th October 2015).

[10]See: http://thecommentsection.org/viewarticle.php?id=5025 (last access 12th October 2015).

[11]See: http://www.imdb.com/character/ch0003035/ (last access 12th October 2015).

groups, there are tools that can make anybody a hacktivist one way or another, simply by being aware of the movement's ideas. Disruption? In my opinion, this is rather a gradual process that evolves hand in hand with society.

The broad field of development is generating an increasing number of new physical development nuclei causing the office ecosystem to begin to lose its original role. And that is where the business leaders-developers (i.e. entrepreneurs) that embarked on a path years ago came from, some of whom have been successful. However, the majority have had to accept their essential failure of not having become millionaires.

So, let's get back to languages after those brief comments on basic issues. Technological evolution has created several changes of direction. Nowadays, (practically) any programmer can create something with a beginning and an end, according to his needs, preferences or life projects. However, another technician may appear on the scene parallel to this and take advantage of part of the knowledge (i.e. code) created by the first person and bring out something entirely new or even focus that same code on different services. The number of opportunities depends on the enormous amount of available code. I often think the digital world is like the world of fashion, where cloth, designs, cuts and materials are all there, but each designer is capable of creating something totally different and sometimes unique. I believe that when language no longer depends solely on hardware, but on the service it is meant to provide, things will change in the development world.

4 Developer Empowerment

If you're a developer, of any type, and you have the mindframe to set up on your own with your technical knowledge, I feel sorry for you. There was a time for that. But today it's a lot more complicated. Not long ago, when they called you to do an online project, you just had to create a digital image like a showcase of what the company supplied. If you want to do something decent today, you have to upload the company webpage on the Internet, which means not only putting in their image but all of their organisation and customer management services.

If you want to do that on your own, I think your physical location is vital. The first thing to take into account is that living and working in a village or small town is not the same as in a big city. It's important to understand that the location of technology poles or clusters is important even if the network is global. Obviously, this also depends on your business aspirations. Several factors come into play here. For example, the first two authors of this chapter come from a town called Ermua between Bilbao and San Sebastian, in the Basque Country. Our town's industrial fabric is based on automotive parts production. And like ours, other towns nearby have a similar industrial fabric. European structural funds prompted many of these municipalities to develop strategies for conversion, we could say, toward manufacturing technologies (i.e. generally known as Industry 2.0 which has currently become the so-called Industry 4.0). Thanks to our "great" politicians

(another key factor), a lot of projects with the same focus suddenly appeared in the political arena. They were incapable of coordinating these projects or of thinking about the overall development and well-being of the area. The result was semi-occupied buildings and struggles to attract business to the area. I am making this remark because I'm sure there are excellent developers, who, due to the circumstances, haven't had the chance to move forward with all their potential and have simply been swallowed up by the ideas of political leaders. And I repeat, instead of having a global vision, finding out what is involved in becoming a technology pole (there are scientists and researchers such as the authors of this book who have spent decades working on this subject) and finding local initiatives to drive development, these politicians wanted to create a second-rate Silicon Valley. As if it were something as easy as Grandma's paella recipe.

I said earlier that some years ago, it was not difficult to work as what we normally understand as a "web developer" if you had some knowledge and a certain amount of self-confidence. The term was certainly unfortunate, it was really like stuffing all of a developer's abilities in a box and giving it a kick to mix everything together. Most of the demand for web developers was pretty superficial, and didn't look to go much further than virtual showcases of the contracting company's true business activity. However, in most cases, they lacked business logic, without any possibility of getting any real productivity from being online in spite of developers who often wanted to suggest initiatives that the companies did not (or chose not to) understand. A few clever individuals managed to become programmer-designers and carried out projects that we find very old-fashioned today.

In today's context, when developers aim to become entrepreneurs, they run into "limited" entry barriers to the ecosystem (aware that legislation plays a key role in this sense and varies according to the country). However, the main obstacle they face when trying to grow in the Internet ecosystem is the fight to survive (i.e. what Moore 1993, referred to as "predators and prey").

For those who are still restless and haven't wanted to become obsolete due to the new technologies coming from big Internet companies, web design has become a specialist field. For those who haven't, however, the job market is continuing to shrink. So, if you still don't know several languages and don't make a profit (or don't know how to), on what you create, you need to get out of the chair and get your brain going. It will be harder and harder to get customers, unless you are extremely lucky (i.e. meaning they have no idea of the Internet's possibilities and make do with what you offer, which is nothing because there are tools that can do your job).

There used to be a saying: look for something in the real world and create it in the virtual world cheaper and more profitably. But that is no longer the case. Everything changed when some developers created Google and found the fastest way to index content. When a developer created Facebook, he wanted to get students connected (and also to meet girls). Due in part to the freedom and ease offered by the network and the technical knowledge that they (not many) had, developers were able to create a business from some very simple ideas (e.g. search for information, connect people). And that, dear "web developer", was the

beginning of the end. The minute that investment funds saw that the solidity of small firms could create profits without large investments (in the initial stages) was when developers became entrepreneurs. And that is when empowerment came true.

That was a mirror for many but was still far from others' reach. However, computer freaks clearly lead their companies according to values that are very different from the existing ones (old-fashioned, greater inertia, bad habits and stagnant organisational routines) or from non-Internet ecosystem related values. And that is where the so called "dark side" of developers-entrepreneurs appears. The ability to reach a barren landscape first allows you to do whatever you like and build roads so that others can arrive, although they are obliged to take your route.

There was no jQuery when I studied programming 10–15 years ago.[12] It was JavaScript, full stop, and you could make animations, but looking back they now seem very flimsy. It was when there were webs with midi music and gif images that never stopped moving. Smartphones were only seen in films and what really thrilled me was learning ASP (Active Server Pages) and PHP, which I used to request entries and store them in databases. But now it's 2015 and I'm still programming in PHP, over HTML5. Thanks to JavaScript, we now use jQuery to manipulate interfaces and the data are no longer strictly structured so we can use JSON (acronym for JavaScript Object Notation, which is a light format for data exchange) and save the entries in a database like MongoDB. I start swearing when I have to work with Less or SaaS, for instance, but now looking back, I see that these new techniques (i.e. workflows) are nothing more than evolved concepts that we all knew and recognised.

For developers, the biggest evolutionary leap forward may not have arrived with the emergence of a new programming language, but with a language like HTML. In its progressive evolution to HTML5, its capacity has been enlarged and at the same time so has that of other languages, making it possible to advance together and take data manipulation to another level. That ability to make the other languages grow may be what I like most. It's a bit like a midfielder who makes incredible passes to the forwards or that point guard who runs the game and controls the tempo in a basketball match.

It's funny, in spite of several innovations in our developer world, not a one has been capable of creating a real disruption. And what I find even stranger is that thanks to these languages, these incremental innovations, new organisational and business models have actually emerged and have checkmated traditional concepts and business models. Don't forget that the term disruption, which is characteristic, although not exclusive to, dynamic, turbulent and high velocity environments (Eisenhardt 1989), involves a radical break (i.e. a paradigm shift) in a process of constant, progressive and gradual evolution.

Do developers think that there is a disruption that is the same or equivalent to what the big Internet companies are creating in other scopes? That was the initial

[12] http://jquery.com/ (last access 12th October 2015).

research question we were going to try to answer in this chapter. My conclusion is that there is not one. Current changes, the speed of such changes, new development patterns, etc., continue without being radically innovative. Developers adapt to the needs of the environment, which makes it a more Darwin-type movement than epigenetic. The following are needed for these dynamics to appear and consolidate:

- Sustained financial capacity from developers or start-ups that want to enter into such dynamics.
- The possibility of attaining certain amounts of intellectual property (patents, copyrights, creative common licences, free software, etc.).
- Access and preparation of their own "human capital" that enable them to penetrate and improve in certain specialist fields or areas of knowledge, such as, for instance, provision of engineering teams or legal counsel needed to defend their positions.
- Marketing research, as well as growth of potential users in different fields that the business group can target (i.e. dominant vectors). In this case, we would also have to consider competition from other business groups (i.e. GAFA, developers and start-ups) to compete on potential markets.

I believe that the epigenetic approach (i.e. EED) to the study of the dynamics observed in the Internet ecosystem makes it possible to clear up the dark side, which at the present time is being played by the same companies that have been capable of true innovation in a disruptive manner in human communication and in the Internet industry. This was, of course, the reason why Gómez-Uranga et al. (2014) introduced the concept of Epigenetic Economic Dynamics. However, it doesn't seem to be the most suitable method to study developers' dynamics (see Sect. 6).

When answering the question of which Internet companies are the most innovative or the biggest ground-breakers as per new ideas, I personally feel that IBM and Oracle, for example, have been much more innovative than the rest of today's GAFA for many years, and having seen the change coming, have been able to defend their market shares. Although I would have to point out that these firms are in sectors that are not as highly visible for most mortals. In spite of that, and not having economies of scale or scalability like GAFA (these phenomena are almost unheard of), these firms managed to revolutionise the world in which we live. Now GAFA are the ones trying to make us believe, almost compulsively, that there are only a couple of development models with a sole approach to making applications, and even that they are only for the goals set by their own marketing gurus. Let me explain, Google, Microsoft and Apple, whose development capacity has enabled them to create the technology base for applications development for their terminals, also oblige us to comply with their specifications (e.g. Application Programming Interface—API). Some are stricter than others, but you have to meet their "terms". Why this unwarranted attack on marketing agents? Because their aim as technology bases is to have their own area for profit-making activities and the marketing agents will sell us the good points but will hide the other part they don't want us to see (e.g. use of personal data for third parties, etc.).

And what was the catalyst that caused people not to give a toss about their privacy and rush off to give their personal information to GAFA? The popularity of

mobile terminals (remember mobility was changing everything). Mobile terminals have stirred up everything… business and people. Especially people, since the majority of applications look to entertain us or give us information (and naturally, capture our personal information), so people are ultimately the main target of all these applications and the focus of online services. So that we can get an abstract vision that we can understand, close your eyes, well no, on second thought, don't. You can't read like that so try dreaming while staying awake: we have an application on our terminal that visualises information on public transport. As we have seen earlier, there is no data on our terminal; it comes from server X and is interpreted by the app we have started up. This same application has an interface that works via a browser. So when our customer is in the development phase, he wants to be in the top search position. Google, for instance, has some guidelines that you have to comply with so that its online services place you in a high search position. This is the "for me or against me", which has always existed. However, we don't seem to mind too much because it's abstract. Although we then enter keywords to place our webpage in a top search position, when we access other webpages, we will see the advertising for goods or services we have previously searched. So now think about your privacy and what you'd like it to be in the present or in the future before you click on that dense text.

This is thanks to APIs (Application Programming Interface) which are the services we discussed earlier and which act as brains that prompt us to connect to them and offer us the capacity to work with their information. One of the latest reports by the market research company ComScore (2014) reveals how all the measurements for terminal apps are shooting up while desktop apps indicators remain stable. Think that having the terminal so near us, with our human obsession to look at the mobile terminal, makes developers, or more specifically, companies direct their products to these devices.

Access to the Internet, exposure of our information, granting our privacy so that our information is exploited globally, has been the newest secrets inflating the bubble. This time, I think they are being controlled by investment funds rather than the laws of the states where they are applied, with the aim of avoiding another bankruptcy like the dot.com one that occurred at the start of the new millennium. Now investors know what this is all about (i.e. often regarded as smart capital—Wriston 1998; Sorensen 2007).

A documentary on epigenetics that I revised a couple of years ago, when one of the editors of the book began to talk about his work on epigenetics and the possibility of writing a chapter for a book on EED, so that I could get a good grasp of the biology part, explained how a grandmother's terrible trauma was reflected in her grandchildren. The kids were exposed to a context similar to the one their grandmother experienced, which had a tremendously negative influence on her, and they automatically became stressed more quickly. We can understand that the very experience, the technological and regulatory chaos we live in, has brought about a change in investors' DNA, ensuring that the digital business model of exploitation and investment does not lead us to another disaster like the one in 2001.

5 The Future: The Evolution of Developers

What will developers be doing in 10 or 20 years? With a bit of luck, the same thing they are today, but adapted to a new environment (i.e. context), and if this good fortune bypasses them, they will be doing exactly what they are today unless they are capable of evolving. As I mentioned before, there are a lot of programmers and GAFA are creating APIs to work with their information and capacity to exploit it. So, on the one hand, we have top level developers focusing on standards development to serve as the base for secondary level development, such as that generated by apps. There's nothing bad about this a priori, but development is provided by big companies (i.e. GAFA) which are the ones arriving with new procedures that do no more than "build" the play area (i.e. we are referring to the APIs developed by the large Internet players). So will the future be controlled by a few big firms? Definitely. Particularly, and this where we get into one of those great conversations, when we take into account that the free code concept is possibly used more than ever by big companies. Just think that companies like Apple (but we could say any GAFA included in this book) create products that have such tremendous social and economic impact, for instance on their followers' image, and can even create social division between followers and non-followers. This prompts developers to adapt to their workflows, tools and hardware, and it is not with the aim of taking full advantage of third party applications, services or developers. It's the hardware they propose and above all, their operating systems that coordinate physical devices, the cloud and in general, all their users' public and private information.

We often find caricatures of these mega firms in series and films although if we take a closer look, their software has changed from being proprietary to being free. The Internet ecosystem companies have fought like wild beasts for patents and standards to keep their competitors in check and develop that same software. Thanks to free software, new business models can be developed and foster the creation of new markets.

We can enter "Web 3.0" in a browser and almost all the responses we find will have a common denominator, "data order". Well, for various reasons, I just don't like labels so I am going to try to rewind to understand the current state of what we now see every day. Did a Web 2.0 exist? I don't think so. Just because it occurred to some people (sorry, Tim O'Reilly)[13] to call something by a certain name doesn't mean it was true. Hundreds, even thousands of designers rushed out to copy what was understood as the erroneously termed "2.0 style". I saw marketers sell 2.0 projects which were 100 % the same, and the only new feature was a social media site plugin.

Of course, as a concept to mark an evolutionary point in history, it sounded good. But none of us are able to think that on Tuesday, 5 October 2004, the Web 2.0 started up. I think a lot of people would think "You are wrong, the Web 2.0 was socialisation, blah blah blah…". Fair enough. But was it like that before social media sites or was it the other way round? Did the concept come from the popularity and use of a certain type of projects?

[13]See: http://www.oreilly.com/tim/bio.html (last access 12th October 2015).

Hoping that the same doesn't happen with the Web 3.0, it is defined by the World Wide Web Consortium (W3C) as the Semantic Web.[14] And we are already making strides in that sense. If not, what is Big Data and why are we making so much noise about it? It is the first step in the above mentioned "data order". We are moving to an Internet of data or, in other words, massive information. These are the main characteristics of the Web 3.0[15]:

- ***Intelligence***. *The Semantic Web project known as Web 3.0 intends to create a method to classify Internet pages, a tagging system that not only allows browsers to find the information on the network but to understand it. By achieving this objective, users can access the Web to ask in their language, the Web will understand that language, and learn the result of the searches for the next operations. Although the fact of learning is just to save values and apply statistics to them; the change in the labelling is the important thing here.*
- ***Sociability***. *Social communities become more exclusive and complex. Social media sites increase as well as the ways in which they connect to their members. It begins to be considered normal for a person to have several identities in their virtual life and the possibility of migrating the identity from one network to another. I would like to remind that social networks are private companies. Are we going to continue giving our data to private companies? This increasingly resembles a dystopian film. We should learn, as discussed before, to apply the digital pedagogy and prevent abuses as we do in our physical life.*
- ***Speed***. *Video broadcasts on the network and the creation of portals devoted to this task, such as YouTube, are possible thanks to fast user connections. The main telecommunications operators have started to implement fibre optics for users with wideband connections up to 3 Mbps ADSL which would convert to speeds from 30Mbps to 1000 Mbps or even faster.*
- ***Open***. *Free software, standards and Creative Commons licences have become commonplace on the Internet. Information is freely distributed on the Web, preventing sole ownership. Capital gains on information are discontinued in favour of more democratic use.*
- ***Ubiquitousness***. *Personal computers will become obsolete due to the multifunctional nature of mobile phone and other portable devices. With the arrival of email on BlackBerry phones on desktops, Apple and iPhone are expected to include the Web. Small screens get bigger and higher resolution, enabling better visualisation of web content. The range of wireless networks and last generation phones increases, expanding network coverage.*
- ***User-friendliness***. *Internet users that visit a new website have to devote a certain amount of time to learning how to use it. The new design tendencies look for standards for a Web with more homogenous and more easily recognisable functions, besides creating spaces that users can set up however they like.*

[14]See: http://www.w3.org/standards/semanticweb/ (last access 12th October 2015).

[15]See: http://datateca.unad.edu.co/contenidos/MDL000/ContenidoTelematica/caractersticas_de_la_web_30.html (last access 12th October 2015).

- ***Distribution***. *Programmes and information become small pieces distributed by the Web and are capable of working together. Internet users can collect and mix these pieces to carry out certain actions. The Web thus becomes an enormous space that can run like a universal computer. Distributed computation systems—systems which connect the power of many computers in one entity—become a commonplace option of operating systems*.
- ***Tridimensionality***. *Tridimensional spaces in the form of virtual worlds as game and online courses will become increasingly common. There will be new devices to move around the Web, different from keyboards, the mouse and optic pencils*.

However, I find that the drawbacks, obstacles or difficulties that have to be overcome for its successful implementation are very serious and specific[16]:

- *The decentralisation of Web management offers developers the freedom to freely create tags and the ontologies they need to make their webpages sensible. However, the downside of this freedom is that various developers could use different tags at the same time to describe the same things. This could enormously complicate the comparisons for the machines due to the possible ambiguity of terms to refer to the same thing*.
- *There is criticism about what is philosophically known as the "identity problem", which centres (in its computational transposition into the Semantic Web) on whether an internationalized resource identifier (IRI) only represents the web resource that it makes reference to, or, in contrast, the implicit concept in the referenced web resource. (e.g. The IRI shows the path of the webpage of an institution. Does it really represent the institution in itself or just a webpage written about it?). The point is important when establishing trustworthy sources or resources considered "axioms" from which knowledge taken can be inferred to be true*.
- *Finally, the biggest obstacle of all is the Semantic Web's dependence on establishment of adequate ontologies and rules to give it meaning. Building ontologies requires a great deal of work and is actually the central issue and where most of the work to build the Semantic Web is done. Will companies and people be capable of devoting the necessary time and resources to creating adequate ontologies so the existing websites can "understand" the Semantic Web? Will their ontologies maintain and evolve as the content of their websites change*?
- *Some sceptical developers disagree with the approach that the Semantic Web should be totally dependent on establishing ontologies and rules, to the point of arguing that the project is unfeasible because of its huge dimension. They contend that the task of creating and maintaining such complex descriptive files is too much work for most people and furthermore, that companies are not likely to devote the necessary time and resources and add the necessary metadata to the existing websites so that they can work properly on the Semantic Web*.

So we see identity problems, creative freedom, possible personal rights at the digital level, etc., are basically the same problems we have today, but in another

[16]See: https://sites.google.com/site/groupccygv/wiki-del-proyecto/web-2-0/hacia-la-web-3-0-la-web-semantica (last access 12th October 2015).

similar context, not very different from our reality. I fear that as it happens at present times, there will be a consortium of actors that recommend certain actions, patterns, etc., but without getting in the activities of the dominant players. Instead of leading the progress, they may simply serve as a collector ideas and developments and provide a forum for standardisation. I'd like our readers to be aware that we are on the verge of changes that could go one way or the other, leading to very different things. Gartner (2014) stated that in 2014, 73 % of the people interviewed were going to invest in Big Data projects in the following years (in 2013 it was 63 %), so we are at the gateway of new dilemmas about how information is handled on the Internet. In this respect, Google, for instance, is doing its homework and now has tools to process large amounts of data very quickly, which gives it the capacity to provide them to generate new applications for third party firms and thus extend the field to the following source of data, which is things. And the rest of GAFA are doing the same thing.

6 Making Big Data Known

So what does Big Data mean at the developer level? Actually, not very much for a technician working as a front-developer in a medium-sized firm. There is something alarming, though. Right now, code developers have access to almost anything to run a project, a database, frameworks, space to place files on a server. However, as of now, access to data organisation is not going to be free. It will be on order from a big Internet firm. And this will divide companies that work directly with data from those that have to rent them, instead of their being freely accessible to anybody.

So we'll be facing the same contradictions we have today (i.e. Does Google manipulate the data it displays in searches?), with all the questions that keep coming up. Everything will continue to be the same until what I foresee as a new turning point, which could be when "things" send data directly to our services.

Lohr (2015) points out how IBM, the hardware and software manufacturer, announced that a considerable sum of money was going to be put into Big Data. More specifically, into the Big Data free code software, Apache Spark. That will trigger another cycle of data explosion, communication, interaction and we will have a 3–4-year period in which to form part of this new data flow, with communication between objects emerging soon afterwards. This will make man more passive than ever as per socialisation and interaction with other human beings.

And what will developers do then? If we are active, we should learn to manipulate those data, since many, although not all, future companies will have a business model based on this type of services. So becoming a provider of these services could be very profitable.

Imagine what the HTML5 standards and changes have meant to some. It has allowed the creation of a large ecosystem, an entire landscape, down to the very last detail. Then the development teams went in and suggested putting in a pipe

network and began to change the way data were handled. They made non-relational (i.e. NoSQL) databases popular and generated frameworks to handle them, thanks to their strengths and new improvements. Let's not forget that the world contains an increasing amount of software, whereas hardware is shifting from an inert object to become a semi-intelligent being, due to the intangibility of software. This is starting to open a gap between people who are capable of understanding, interacting, working and negotiating with intangible information and those who can't. In just a few years, the rift will exist between people who have a close relationship with intelligent objects and those who don't. And when the time comes, we'll see how society is fragmented and what social groups are formed. Just give it time.

Going beyond this, it's plausible to think that the Big Data system itself will become an axis of the Internet. To offer a simile, Big Data services will play a similar role to today's browsers, but perhaps at a more concealed level from the final user but which developers will be required to know about. Who knows? A couple of young people who avoid investment funds might come along and turn out to be capable of coming up with a new focus for data use and create a totally alternative business model. They might become a major Internet company and force developers to adapt and evolve with them. Anything is possible.

And on top of that, revising the Web 4.0 guidelines (yes! Web 4.0!) people like Nova Spivack say that around 2020–2033, network intelligence will take a huge leap forward, which is the inspiration for Web 4.0 and it will be similar to human reasoning. Like today, we will make the sum of many services be regarded as a standard. The question, as I mentioned earlier, depends on whether it's going to be an open model or one by request run by the big firms that exist then. Depending on this, developers should learn to use new frameworks and methodologies to run what our customers want or what firms demand, in the event that we create a start-up. Whatever happens, I don't think it will very different from today's reality, but probably from another environment or context. There will still be top level developers and others who are pulled along by the tools or language evolution developed by the former.

So, after all that we have talked about, I return to the question: is it disruptive? I don't think so, but what we do see increasingly is a pattern and, in our case, GAFA are like a sun that erupts from time to time, resulting in a true disruption. And, as I said, it is becoming a pattern and eruptions are fantastic to look at but are extremely dangerous.

7 How Can We Study the Evolution of Developers? an Analytics Proposal Based on Quantum Physics

In the previous chapters (mainly Chapter "Introducing an Epigenetic Approach for the Study of Internet Industry Groups" by Gómez Uranga et al.) we extensively developed the EED model which was later applied to GAFA. However, in the third

section of this chapter, we reached the conclusion that the developers' dynamics seem to fit Darwinist characteristics more than epigenetic. As a result, although the analytical framework of the EED does adapt to the study of the big Internet business groups' dynamics in an interesting manner, we don't believe this happens with the other side of the coin, which is developer dynamics. This, in turn, leads us to think that, in view of the characteristics of the Internet ecosystem and the size of the population which is the object of study (i.e. 18.5 million developers around the world) perhaps we need to think of an analytical framework that will enable us to understand the phenomena that occur in the world of developers.

In this section, we intend to introduce a new analogy which is related to quantum physics to apply it to the case of developers and attain a more robust analytical, conceptual and empirical understanding of their characteristics and dynamics. Of course, this is only a methodological proposal which must be strengthened and applied, so there remain many stages that must first be studied and many challenges to be overcome with this 'cross-fertilization'. First, bringing in new concepts implies the need to develop new analytical approaches. Second, these conceptual approaches need to be translated into methodological tools. Third, in order to validate these new approaches, it is necessary to gather data from the different actors that are operating and shaping the Internet ecosystem (i.e. developers and start-ups). Gathering these data is a task in itself. There are millions of start-up companies constituted by developers, scattered across the globe and which are very small in size. Following them constitutes a difficulty itself as their traces are not observed in the market from the moment of constitution, but rather when they are acquired by large players. In this respect, we will develop what fundamentals/properties of this latter discipline could be imported to our fields of study. We therefore feel that the application of epigenetics to study the behaviour of the big Internet business groups through EED could be supplemented with an appropriate use of a quantum approach for the case of developers.

As noted, in order to understand the dynamics of the Internet ecosystem, it is necessary to know the dynamics of developers, which materialise in the creation of new technology start-ups. Developers (i.e. entrepreneurs) are key in explaining epigenetic dynamics. Examples of GAFA absorbing entrepreneurial ventures include Facebook acquiring Instagram, Whatssap or Oculus; Google acquiring Nest; or Microsoft acquiring Mojang for instance (see Chapter "Epigenetic Economics Dynamics in the Internet Ecosystem" by Zabala-Iturriagagoitia et al.).

Developers are becoming the cornerstone of the Internet's rapid development and the abrupt growth of the large industry groups dominating it. The literature increasingly emphasises the relevance of entrepreneurs in employment generation (Autio et al. 2014; Bruton et al. 2013; Engelen et al. 2014; Mazzucato 2011). However, in spite of the key role developers play in the dynamics of the Internet ecosystem, there is hardly any evidence regarding which dominant vectors are guiding developer activity, their economic impact both in terms of employment generation in the geographical areas where they are located, the generation of added value, the challenges they encounter when facing competition from GAFA, or their strategies in relation to intellectual property protection. We believe the

principles of quantum physics could be adopted to systematically explain the evolution and changes in the orientation of developers and how they are influenced and, at the same time, affect the changes in the Internet ecosystem.

Quantum physics is characterized by three principles: quantum superposition, entanglement and collapse.

- Quantum superposition: Schrödinger's uncertainty principle, determines that a particle is in all the states it could potentially be in. This is illustrated by the metaphor of Schrödinger's cat, which is alive and dead at the same time (i.e. a particle staying in all possible states at the same time).
- Quantum entanglement: this principle shows how a set of particles cannot be defined as single particles with defined states, but rather as a system with a single wave function. The strong relationships between the particles (entanglement) make the measurements done on a system appear to instantly have an influence on other systems that the original system is intertwined with, no matter what the separation among them is. In other words, a particle affects the system as a whole. Accordingly, the distribution of probability of the particle being located in a concrete state is dependent on the system as a whole.
- Quantum collapse: refers to the transition of a quantum system from a superposition of states to a concrete state. It is related to the quantum superposition principle, inasmuch as a particle, which can potentially be in any possible state, when making an observation on it, will collapse to a concrete state with a defined value. The process is also known as collapse of the wave function or collapse of quantum states, and the probability of collapsing to a given state is determined by the wave function of the system before the collapse.

The relationship established between the dynamics of these three quantum principles and developers is as follows (Table 2). Initially, given the horizontal character of the software industry and the generic capabilities required in it, every developer could be oriented to all potential activities and industries (i.e. quantum superposition). However, developers opt for certain dominant vectors (Suárez et al. 2015). This decision to focus on certain markets would be equivalent to the quantum collapse. As a result, the introduction of developers into certain markets alters the situation in which that market or sector showed during a previous state (quantum entanglement).

Application developers are becoming increasingly important, not only for the dynamics of the Internet ecosystem, but also for new employment creation and economic growth. The dynamism of current societies is based on the development of applications and on entrepreneurship to a greater extent (Glassdoor 2015; Newbert et al. 2008). Developers have multiple directions or dominant vectors they orient toward. Depending on their location and the characteristics of the environment in each location, they will collapse into these vectors with different probabilities. The key lies in identifying the dominant vector that may guide the activities of entrepreneurs (i.e. developers) and which rely on the higher efficiency in each location (Zabala-Iturriagagoitia et al. 2007). Finding out which the dominant vectors are in each territory, policies (e.g. entrepreneurship, innovation,

Table 2 Relationship between the quantum properties and developer dynamics

	Millions of developers	Billions of electrons and particles
Quantum superposition	A developer can orient to all the potential states it could potentially be in	A particle is in all the states it could potentially be in
Quantum entanglement	Developers depend on global relationships and requirements	Particles lose their meaning as isolated elements
	New firm entry (created by new developers) has an immediate influence on the ecosystems, altering their previous situation	They are precisely defined by their entanglement with other particles
Quantum collapse	When making an observation on the start-up, this will be specifically defined in a particular state from all the existing options it could initially be oriented to	When observing a given particle or element, a perfectly defined state is created

Source Own elaboration

employment, education, regulatory, tax, etc.) may imply greater effectiveness when supporting these entrepreneurs' activities.

Naturally, as in all entrepreneurial processes, a large share of the new entrants fails. This is where the quantum probability becomes important. In the stage prior to entrepreneurship, there is superposition of states since the entrepreneurial firm can both fail and survive. Therefore, the quantum analysis would be equivalent to a probabilistic analysis. The novelty of the project lies in that there are millions of developers, but a minority of these succeeds and becomes firms of a certain size and reaches a certain degree of success on the market. It is here that the quantum approach meets EED, as the difficulties entrepreneurs face are, to a great extent, due to the dynamics of GAFA. A large share of the developers has great interest in being acquired by GAFA, since they know they cannot outcompete them due to their financial power. What is more, these large groups often even "own" their developers (e.g. through the organisation of huge contests or hackathons). As we have described in this chapter, GAFA often act as "lodestars" that guide the action of the developers themselves. That is why we have often referred to the Internet ecosystem as the two sides of the same coin (Perks et al. 2012).

In order to study the dynamics of developers and their start-ups, it would be possible to rely on the use of quantum Bayesianism. Quantum Bayesianism was introduced in 2002 by Caves et al. (2002), unifying quantum physics with probabilities. The adjective Bayesian is due to Bayes' theorem and the conditional probabilities used in inference processes. In Bayes' theorem, evidence (or observation) is used to infer the probability that a hypothesis may be true. The basic idea behind Bayesian probability is therefore a calculation of the consistency of the credibility of a certain hypothesis. In the Bayesian context, the term "degree of belief" is used, since the expert believes that something can be real (i.e. can take place) with a certain level of belief, and therefore cannot assert whether something is true or not. The actual beliefs come from external sources, and the researcher sets the degree to which something can happen by assigning an ex-ante

probability. As the system is collapsed according to different experiments, the researcher defines the probabilities for the different options to actually take place according to the observations of those experiments (i.e. the results of each collapse, which will be different).

Certain environmental variables may be significant in assigning these ex-ante probabilities, which may be defined by the researcher. A starting point for this definition can be Jeffreys' models (1961, 1973). Using regression models in which the target variable (i.e. the most efficient dominant vectors in each location) is a probability function, and in which variables related to the environment where the developers are located are used as explanatory variables, it is possible to reassign the initial ex-ante probabilities using Bayes' theorem. Consequently, the new probabilities are defined, which modify the initial beliefs according to the information provided by the data. The model will be amended as new variables related to the environments are introduced, obtaining the Bayes' factor, as a result of the likelihood ratio of the first and second models once the new variables have been included. This results in a series of nested models that seek efficiency in the probabilities of the decisions to be made by the developers concerning their activities.

The previous methodology requires identifying a set of systemic variables representing each environment. Some of these contextual variables might be:

- Level A: Quantitative indicators
  - General structural indicators such as those included in the Innovation Union Scoreboard.
  - Other relevant indicators related to entrepreneurship such as youth unemployment, new business creation, survival rates, sectors with higher growth rates, etc.
  - Most relevant economic sectors in each country.
  - Type of firms according to age, owner, size, R&D investments.
  - Availability of a trained labour force: share of the population with higher education, disciplines in which people in the country are more specialised.
  - Availability of venture, seed and risk capital.
  - Extent to which the Internet is implemented in the country, share of purchasing over the Internet.
- Level B: Qualitative indicators
  - Vertical priorities in terms the different governments' policy such as health, sustainability, manufacturing, energy, creative industries, etc.
  - Public support for entrepreneurial action: are there entrepreneurship policies? Are they focused on specific industries?
  - Presence or absence of large multinational corporations that may act as 'drivers' of their respective economies, which may attract not only developers and their start-ups but also other large corporations to the 'hot spots' where they are located.
  - Quality of the institutions.
  - Comprehensiveness and level of relationships among the different parts of the innovation system.

- Entrepreneur-friendly climate.
- Share of people with programming skills (coding, Big Data, cloud computing, mobile, data visualisation, user experience designers).

These potential variables are far from being comprehensive and this list should be considered a preliminary approximation. One of the weaknesses of this model is that there are multiple variables that cannot be included, such as the developer's subjectivity (e.g. whether its motivation is to grow or not to grow and just create a small amount of employment), the team size (particularly when the firm is constituted), family background, level of competence and skills, etc. These variables require other non-economic sciences such as genetics, sociology or political science, among others. The model also needs to be dynamic in order to capture the evolution in the different territories. This requires constant updating of the data, year by year. As indicated, the main challenge involved in the operationalization of this quantum approach is related to data acquisition, since many of the potential indicators listed above are not systematically collected. Therefore, quantitative and qualitative methods need to be combined to gather the necessary data. Finally, in order to assign ex-ante probabilities to the direction that developers may take, it is necessary to assign some weights to the chosen indicators, which also raises certain challenges as the weight of the structural indicators will vary from territory to territory.

The interest and potential usefulness of the quantum approach is manifold. First, it can be useful for policy makers to shape their policies, priorities, financial investments and the alignment among policy domains (e.g. environment, health, education, etc.). It also points out the environmental elements that need to be improved, supported, included or even eliminated so developers in their respective territories can have higher probabilities of success. Third, it can also assist GAFA in their diversification strategies, as it captures the dominant vectors on the one hand, and the type of vector that developers in different territories orient to. Fourth, the model can also be useful for the developers themselves, so as to know which locations are the most efficient, depending on the sector they want to focus on. Finally, the model can be of great interest for investors, enabling them to know where developers are, according to the sector they are interested in.

The previous quantitative and qualitative variables should embrace both the local and the global levels. Naturally, many of these variables are local in character. However, many others are global, not only because the dynamics of the GAFA are global, but also because the big venture capital firms, the lines of action (which are often defined at the European level) and even the Internet market are global. This does not preclude that local analyses of certain geographical contexts can also be carried out. With the very initial model we would be drawing thick lines, which identify the efficient ones? at the national level (a dominant vector for each country). However, ideally, the more variables at the local scale that could be included, the more accurate the dynamics of entrepreneurs and developers could be. It would be equivalent to zooming into see how the thick lines at the national level are also divided into thinner lines. However, this is very much dependent on the challenges involved in data acquisition and the level at which these can be gathered.

References

Autio, E., Kenney, M., Mustar, P., Siegel, D., & Wrights, M. (2014). Entrepreneurial innovation: The importance of context. *Research Policy, 43*, 1097–1108.

Bruton, G. D., Ketchen, D. J, Jr, & Ireland, R. D. (2013). Entrepreneurship as a solution to poverty. *Journal of Business Venturing, 28*, 683–689.

Caves, C. M., Fuchs, C. A., & Schack, R. (2002). Quantum probabilities as Bayesian probabilities. *Physical Review A, 65*(2), 022305.

Comscore (2014). The U.S. mobile app report.

Eisenhardt, K. M. (1989). Making fast strategic decisions in high-velocity environments. *The Academy of Management Journal, 32*(3), 543–576.

Engelen, A., Kube, H., Schmidt, S., & Flatten, T. C. (2014). Entrepreneurial orientation in turbulent environments: The moderating role of absorptive capacity. *Research Policy, 43*, 1353–1369.

Gartner (2014). Survey Analysis: Big Data investment grows but deployments remain scarce in 2014. http://www.gartner.com/newsroom/id/2848718. Accessed 19 Oct 2015.

Glassdoor (2015). http://www.glassdoor.com/blog/jobs-america/. Accessed 19 Oct 2015.

Gómez-Uranga, M., Miguel, J. C., & Zabala-Iturriagagoitia, J. M. (2014). Epigenetic economic dynamics: The evolution of big internet business ecosystems, evidence for patents. *Technovation, 34*(3), 177–189.

IDC (2013). 2014 worldwide software developer and ict-skilled worker estimates. International Data Corporation.

Jeffreys, H. (1961). *Theory of probability*. New York: Oxford University Press.

Jeffreys, H. (1973). *Scientific inference*. Cambridge, England: Cambridge University Press.

Kuhn, T. S. (1962). *The structure of scientific revolutions*. Chicago: University of Chicago Press.

Lohr, S. (2015). IBM invests to help open-source big data software and itself. http://bits.blogs.nytimes.com/2015/06/15/ibm-invests-to-help-open-source-big-data-software-and-itself/?_r=0. Accessed 19 Oct 2015.

Mazzucato, M. (2011). *The entrepreneurial state*. London: Demos.

Moore, J. F. (1993). Predators and prey: A new ecology of competition. *Harvard Business Review, 71*(3), 75–86.

Newbert, S. L., Gopalakrishnan, S., & Kirchhoff, B. A. (2008). Looking beyond resources: Exploring the importance of entrepreneurship to firm-level competitive advantage in technologically intensive industries. *Technovation, 28*, 6–19.

Perks, H., Gruber, T., & Edvardsson, B. (2012). Co-creation in radical service innovation: a systematic analysis of microlevel processes. *Journal of Product Innovation Management, 29*(6), 935–951.

Quintero, C. (2015). Who invests in hardware startups? http://techcrunch.com/2015/09/12/who-invests-in-hardware-startups/. Accessed 19 Oct 2015.

Sorensen, M. (2007). How smart is smart money? A two-sided matching model of venture capital. *The Journal of Finance, 62*(6), 2725–2762.

Suárez, F. F., Grodal, S., & Gotsopoulos, A. (2015). Perfect timing? Dominant category, dominant design, and the window of opportunity for firm entry. *Strategic Management Journal, 36*, 437–448.

Veracode (2013). The history of programming languages. infographic. https://www.veracode.com/blog/2013/04/the-history-of-programming-languages-infographic. Accessed 19 Oct 2015.

Wriston, W. B. (1998). Dumb networks and smart capital. *Cato Journal, 17*(3), 333–340.

Zabala-Iturriagagoitia, J. M., Voigt, P., Gutiérrez-Gracia, A., & Jiménez-Sáez, F. (2007). Regional innovation systems: How to assess performance. *Regional Studies, 41*(5), 661–672.