Reducing Friction in the Creative Adoption of Software Artifacts

Emilio Dorigatti

May 14, 2018

1 Introduction

That science is an important driver of economic growth and national wealth was already postulated in 1800 (Rosenberg 1974), and many studies, such as (Henderson & Cockburn 1994) have confirmed this link. Technological development is what connects scientific research to economic growth, and an important factor to accelerate the rate of technological innovation is the rapid diffusion of the knowledge generated from scientific works. Scientists disseminate their works through different mediums, such as conferences, personal correspondence, professorship, and pubblications. This last method was shown to be by far the most important one to quickly spread information (Sorenson & Fleming 2004), and thus generate, ultimately, economic value, for the simple reason that, without pubblications, discoveries spread only through the personal social network of the authors, typically very localized.

Nowadays, most researchers make use of computers to analyze data, and often write their own code to perform at least parts of their work; in Computer Science, however, the code itself is often the main outcome of research. In other quantitative fields, such as physics, the primary product of research is a new idea, appropriately validated with data, and adoption happens by means of extending the idea with new insights, and validating them with more data. In contrast, in Computer Science, adoption requires using the code produced by other researchers, to extend it, to use it as a component in a larger system, and so on. It is therefore very important to make it as easy as possible for other Computer Scientists to adopt one's work.

1.1 Research Question

The Open Science movement stresses the importance of opening the research process, and the diffusion of innovation is based on effective communication of information. Undoubtedly, this can be put in practice in different ways depending on the research field, because different disciplines operate in different ways and produce different research outcomes. By restricting ourselves to a specific field, we can formulate the following

Research Question: How can the authors of a research project in Computer Science improve their work to increase its adoption by like-minded creative individuals, and how is adoption affected by these actions?

1.2 Delimitations

The research projects that we refer to are those which produce a tangible and verifiable solution to a problem, in form of an algorithm that must be run by a computer in order to generate actual value. In this context, any idea, no matter how clever, remains only an idea, and the only way to generate value from it is to transform it into an algorithm and implement it, so that it can be used by a computer system to solve actual problems. Computer Science is a very broad field, and not all research works in this field fit this description, but, for the lack of a better term, we will improperly use it with the sense described above.

Another restriction that we apply is to consider only adoption for creative purposes, by other scientists and engineers, either professionals or amateurs, such as students; the point is that this work is adopted to be extended, improved, or used to build something else. We do not consider commercialization strategies to maximize adoption by end users, who only make a passive use of it, and we do not consider adoption by businesses either, since a company has very different needs and requirements compared to a creative individual.

Given that we are considering tangible outcomes of these research projects, the diffusion can be measured with the rate of adoption, a measure used by Everett in his diffusion of innovation theory

(Roger 2003), described later in this document. Even for creative use, software can be adopted in many different ways: using it as a component in a bigger system, applying it to a novel use case, resolving issues, extending it with new features, are all possibilities that are covered by the meaning of adoption that we consider in this paper.

1.3 Purpose

There are many possible ways for Computer Scientists to make their work easy to adopt by other researchers. In this work, we identify some actions that Computer Scientists can do to foster the diffusion of their work, and apply a general framework for the diffusion of innovation to qualitatively understand the merits of these actions.

1.4 Research Methodology

Due to the limited scope of this work, it is not possible to perform a full quantitative data analysis to determine the factors that affect adoption of a software product. Instead, we will perform a literature review to find studies that identified some of these factors, possibly in fields other than Computer Science, and relate these factors to our research question. The outcome can be seen as a set of guidelines that researchers can use to increase the impact of their work, inspired by empirical evidence as well as the author's own experience. The identified factors will be qualitatively evaluated against Everett's framework to determine what aspects of an innovation they affect.

2 Background

The study of the diffusion of innovation was popularized in Roger (2003); what follows is a very brief summary of the main points of this theory that are useful to answer the research question presented previously. According to Everett, diffusion is essentially a form of communication where the subject of interest is an innovation. The essential factors in a communication process are information being transmitted, the channel through which it is transmitted, the time that it takes the

information to reach the involved actors, and the social system to which these actors belong. All these factors are discussed in the next sections.

2.1 The Innovation

The main feature determining whether an invention is innovative is not so much its newness, as measured by, for example, the time since it was invented, but the *perceived* newness by the receiving actor. Everett uses innovation and technology as synonyms, referring to a piece of information that essentially reduces uncertainty, in the sense that, after an actor gets to know the technology, the added information advances the actor's understanding. Usually, an innovation is composed of two components: a software aspect, comprising the underlying idea and informative content of the innovation, and its hardware aspect, which embodies the actual realization of the technology; not all innovations have a hardware aspect. Given the context of this work, it is important not to confuse the hardware and software aspects of an innovation as defined by Everett, and the hardware and sofware terms used to describe a physical computer and the code it runs. In Computer Science parlance, an algorithm corresponds to Everett's definition of software, and its implementation in code corresponds to Everett's definition of hardware. In the remainder of this paper, we will stick to the Computer Science meaning of the terms, algorithm and implementation, and use software as a synonym for the latter, unless explicitly stated.

2.1.1 Characteristics of Innovation

An innovation can be characterized by several subjective traits, that can be perceived in different degrees by different actors, and that determine the rate of adoption:

- *Relative advantage* is how much better the innovation is perceived, compared to the idea that it takes over;
- *Compatibility* is the degree by which the innovation is consistent with values, beliefs, norms, needs, etc. of the potential adopters;

- *Complexity* is the perceived difficulty in grasping the essence of the innovation and productively put it into use;
- *Trialability* is how easy it is for new users to try the innovation in the setting they need it;
- *Observability* is the degree by which the advantage brought by the innovation can be quantified.

All these characteristics are positively associated with the rate of adoption, so that innovations that score highest in all these traits are the ones that spread more quickly.

2.2 Communication Channels

The diffusion of innovation can be described as, essentially, a communication process, and the channel over which such communication is performed has an important role in determining how an innovation is perceived. Mass media channels diffuse information using mass mediums such as television, radio and newspapers, whereas interpersonal channels involve direct, face to face exchange of information between a smaller group of individuals. The recent widespread adoption of the internet makes these definitions somewhat obsolete, but the gist is that in mass media channels the information is broadcasted to a wide audience, not aimed at particular individuals.

Previous studies showed that an individual learning about an innovation evaluate it based mainly on the its subjective evaluation from other individuals who have already adopted it, and the trust that this individual places on previous adopters. This phenomenon is known as *homophily*, and essentially states that most effective communication occurs between homophilous individuals, i.e. individuals who are similar in a large number of attributes.

2.3 Time

Time is an important factor in the diffusion of innovation, and can be used to characterize several stages during the diffusion process, such as the time it takes for an individual to be informed of

the new technology since its first introduction, how long it takes for it to decide whether to adopt an innovation, and when to communicate it to ther individuals. Moreover, a central characteristic of diffusion, namely the rate of adoption, is defined as a function of time. Adoption refers to the fraction of individuals in a population that are making use of the innovation, and follow an s-shaped curve, increasing slowly at first and accelerating until the system almost reached saturation, at which point the rate of adoption gradually slows down and, eventually, stops.

2.3.1 The Innovation-Decision Process

The innovation-decision process is the sequence of steps followed by an agent, from when it first learns about an innovation to when it reaches a decision on whether to make use of this innovation or discard it. Everett breaks this process down into five steps:

1. *Knowledge:* when an agent is first exposed to the innovation, and gains a basic understanding of it. During this stage, the agent seeks the software information of the invention, as defined by Everett, in order to find what the innovation is and how it works. Mass media communication channels are an effective way of transmitting such information;

2. *Persuasion:* when the agent is forming an opinion of the innovation. In this stage, the hardware aspect of the invention, as defined by Everett, is the main driver that helps the agent decide to which extend the innovation is applicable in its situation, if at all, and its advantages and disadvantages. Interpersonal communication also plays an important role in this stage;

3. *Decision:* encompasses the activities that lead the agent to reach a decision regarding the adoption of the innovation, such as trying the innovation in a simple setting. This stage can result in either adoption or rejection;

4. *Implementation:* when the agent, after deciding favourably towards the innovation, puts it into use in its specific situation;

5. *Confirmation:* when the agent seeks evidence that its implementation of the innovation gives the expected results. This step can result in the rejection of the innovation, in case it did not result in the advantages the agent expected.

Persuasion and decision might appear overlapping in scope, but they answer two very different questions, respectively "is this innovation any good?" and "do I want to use this innovation?".

2.4 Social System

Everett defines a social system as a set of agents that interact and cooperate with each other in order to solve a common goal. Diffusion of an innovation always occur within a social system, and is affected by the structure of this system, i.e. the interaction patterns of the agents. An example of this are communities, defined as groups of agents which interact much more frequently with members of the same community than with members outside of it. One would expect, then, that information spreads more easily within one community, and takes more time to reach agents in separate communities. Another factor that affects the diffusion of innovation in a social system are the system norms, intended as the established and accepted behavior patterns, such as traditions; it is easier for a social system to adopt innovations that align with the values accepted in the system (this is related to the *compatibility* trait discussed earlier). Opinion leaders have a strong influence on the opinions of the other agents of a social system, therefore they can significantly affect the diffusion of innovation, either by promoting or by blocking it; note that there is a connection between the norms of a social system and the attitude of its opinion leaders toward change. Finally, change agents are those agents who have the authority to impose change on the other members of the system.

3 Results

As is evident in Everett's framework, reducing friction is one of the most effective ways to improve adoption and foster the diffusion of innovation. We identified five broad categories of actions that

achieve this in Computer Science: open access, technology, documentation and comments, tutorial, and use case; each will be discussed in the following sections.

3.1 Open Access

Here, we adopt a more broad definition of open access than the one commonly used when talking about scientific work: we consider varying degrees of openness, according to what assets are made available by the authors, whereas the commonly used definition of open access refers only to the publication. Cumulatively, in order of importance, they are:

1. *Publication:* the most basic form of Open Access regards the publication of a scientific paper describing the fundamental idea, or ideas, that allow the problem, subject of the research, to be solved, as well as the benefits of this idea over previous or alternative solutions, and the evaluation results, proving the benefits of the contribution with respect to competing solutions;

2. *Algorithm:* in Computer Science, it is often not enough to present an idea in natural language, because of its inherent ambiguity. Presenting the solution to the problem in a formal language, be it a diagram or, even better, pseudo-code, is fundamental both to eliminate possible ambiguities (not all readers/writers are native English speakers!), and to specify details that are irrelevant to the idea, but very important for its practical realization;

3. *Parameters:* many algorithms are controlled by parameters that can be tuned and customized to better suit a specific application; the authors should specify the exact values used in every test they perform, and how these values are chosen. This helps readers better understand the trade-offs involved, and possible ways of tuning the parameters themselves;

4. *Code:* often, implementation details are not described when discussing the contributions for brevity requirements, even though they can make a difference between successfully implementing the algorithm, and failing to do so. Moreover, there is often a considerable amount

of supporting code, used to prepare the inputs for the algorithm, to evaluate it, and to analyze the results of the evaluation. Letting other people access the code ensures they are able to reproduce the claims in the publication, as well as readily adopt it in their own work;

5. *Data:* the data used in the evaluation, both input data (if applicable) and the raw results, not the summarized version appearing in the paper. This is important when the authors use data that is not already available to the public, for example when they create a synthetic dataset. In this case, it is important to know how the synthetic data was created, and to have the possibility of altering this process, since the results presented in the paper may depend on it. Sharing data can either be trivial, when the authors use the accepted benchmark data in their field, or pose significant challenges, e.g. because of the size of the data, or be impossible, for example when the data is protected by industrial secret.

The added benefit of these stages is not linear, and they are not necessarily followed in this order. We are focusing on research projects that produce algorithms as the main outcome, and, as we argued previously, the only way to create value out of an algorithm is to implement it so that it can be run by a computer. It follows that giving access to the actual code is by far the most beneficial step that authors can take.

Relating this to Everett's framework, we find that every increasing level of openness increases the trialability of the idea, and, ultimately, the observability of its advertised benefits. Every new step, until the fourth, aids potential adopters in trying the idea in their setting with less and less effort, since it reduces the amount of work they have to perform. Moreover, most stages of the decision process can benefit from increased access to the resources mentioned above; from persuasion to confirmation, reducing adoption friction can be a determining factor in many situations. For businesses, which are always trying to optimize their processes, it translates to reduced economic costs, as less workers, possibly less skilled, are required to incorporate the innovation into the business's operations; this relieves the business from hiring specialized staff, and allows workers to be allocated to other tasks. For students, who are short on time to complete assignments and have incomplete technical capabilities, it can transform adoption from impossible to viable. For

researchers, who often are short both on time and money, it allows them to perform more work in less time. In general, the more work is available, the less time is wasted on redoing the same things.

Proper sharing of software requires knowledge in strategies for licensing it, a notoriously complex and intricate topic (Morin et al. 2012). Nonetheless, given the utmost importance of sharing code, (anecdotal) evidence that this is already a diffused practice in the Computer Science community, and the fact that the next sections build on the availability of code, from now on we will assume it is available.

3.2 Technology

Nowadays, programmers rarely write everything from scratch; instead, they make heavy use of code written by other programmers. This re-use is what enables any non-trivial piece of code to be written relatively easily, without requiring expertise in a number of specialized fields of Computer Science, and multi-million euros budgets, mostly wasted in reinventing the wheel. With the umbrella term *technology*, we refer to everything that is not implemented from scratch by the authors of an innovation, and that is required for the innovation to function properly and to be extended. Examples of technologies include the programming language, the frameworks and libraries used, as well as infrastructural components supporting the software, such as the operating system(s), the storage system(s), the computer hardware, and so on.

Often, there are many competing technologies that can be used to perform any given task; choosing one among them is very important, because they are not entirely compatible with one another. Switching to a new one later on often requires a lot of work, and, possibly, a redesign of the system, or parts thereof. This is very clearly related to the compatibility of an innovation with the existing system used by the potential adopters, as well as their knowledge about the technologies used to implement the innovation. This, in turn, reflects on the implementation stage of the innovation-decision process, as higher compatibility is reflected in less work to integrate the innovation with the other parts of the systems. These factors should also be considered in the decision of whether to adopt the innovation, although this heavily depends on the computer literacy of the

decision makers themselves. Note that, through considerable development effort, it is possible to offer an interface with multiple technologies at the same time, such that adopters of a project can use any of them.

Some technologies can be so innovative and disruptive that the hype surrounding them affects the knowledge and persuasion stage, as well. Recent examples of this phenomenon are the Blockchain (Nakamoto 2009) and Deep Learning (Krizhevsky et al. 2012): everybody is talking about them, because they are the solution of every problem! Or so the saying goes… This cargo cult is, in part, the result of ignorance and/or irrationality on the part of potential adopters, and, in part, because of the very disruptiveness of these innovations, so the community has to learn proper use cases by applying them to all sorts of problems. In spite of this, we feel that evaluating the goodness of an innovation on the basis of the technologies it uses is very superficial, and using a specific technology in order to reach the largest amount of people is more of a marketing decision than research.

3.3 Documentation and Comments

In the Computer Science field, documentation refers to a textual description in natural language of the parts composing a software, describing how they work, what they need to work correctly, how they can fail, and so on. All of this is clear to the programmer who wrote a particular piece of code, but a programmer seeing this code for the first time often has to put considerable effort into understanding this information, which is obviously essential in order to be able to use and extend such software.

Writing documentation is notoriously a boring activity for most programmers, since it feels like writing obvious trivialities, and is not as exciting as writing code. However, even the programmer who wrote a particular piece of code might have difficulties understanding it months later; this speaks of the importance of having a description of the operation of a piece of software that is easier to understand than the software itself.

Documentation is often presented in a different medium than code, such as web pages, so that

users can handily browse it without having to read the code, which is often not interesting by itself. Sometimes, though, it is necessary to read the code itself, for example when the documentation is not clear or incomplete. In this case, the programmer who wrote the code can help readers understanding it by adding comments, short sentences describing a very specific piece of code. Good comments greatly aid the reader in gaining a deeper understanding of the algorithm, the flow of the code, and tough or obscure parts that are sometimes unavoidable.

Referring to Everett's framework, the availability of good documentation increases the trialability of the innovation, and impacts the implementation stage the most, since it is in this stage that the code of the innovation is built upon. Doing this requires understanding it very thoroughly, and, as argued above, documentation and comments are a great aid. The quality of documentation can also be a factor that is considered in the decision stage, since it can impact the implementation stage: bad or absent documentation will hinder the implementation and future maintenance of the software, whereas good documentation, on the contrary, will make it easier, quicker, and, ultimately, cheaper. Whether documentation is considered in the decision stage is, though, heavily dependent on the decision maker and her background; if the decision maker does not have experience in writing software, she is likely not going to consider documentation as important.

3.4 Tutorials

Whereas documentation is a detailed description of the components of the code, a tutorial is a description of how they can be used to create a very simple application; a tutorial is aimed at beginner users who know what the software does, but are not familiar with the structure of the code. Through the tutorial, they are shown what the main component are, and how they fit together. Often, a tutorial is a starting point that allows new users to quickly gain familiarity with the essential aspects of the code-base, and serves as a foundation for them to prototype new applications.

Tutorials mainly improve the potential adopters in forming an opinion about the complexity and the trialability of the innovation. Since tutorials usually interleave short snippets of code with explanations, it is easy for readers to foresee how the code needs to be modified to suit their desired

application. However, since tutorials showcase very simple applications, they are usually not suited to convey the relative advantage of the innovation over the existing alternatives, and, following the same line of reasoning, the impact of tutorials on the implementation stage of the innovation-decision process is very limited, since it only helps the very first steps of that stage.

3.5 Use Case

With use case, we refer to an application of the innovation to a complex and interesting problem, in order to showcase the merits of such innovation. Whereas a tutorial is a simpler application geared towards introducing the inner workings of the innovation, an use case is more complex and elaborate, whose purpose is to show the strengths of the new product, and the possibilities it opens to potential adopters.

In today's hyper-connected world, people are flooded with new information, and it is increasingly difficult to separate signal from noise. The only way to process more information in the same amount of time is to reduce the time spent on any particular piece of information; this means that, sometimes, an innovation is unfairly dismissed without careful consideration. In the so-called "attention economy", this can happen in seconds. We pose that an interesting use case is the most effective way to swiftly introduce an innovation to potential adopters, since it can often be summarized in a few sentences.

The main purpose of an use case is to introduce the relative advantage of an innovation, by demonstrating how much better a problem can be solved compared to the existing alternatives. A good use case also increases the observability of the innovation, by explicitly measuring the advantage it brings. Use cases can be used to reach adopters that would normally not be reached without it, for example by applying the innovation, originally devised in the research field F, to a problem in research field G. Furthermore, a good exposition of the relative advantages can persuade potential adopters of the goodness of the innovation.

Based on this, use cases are mainly a way to advertise the innovation. When discussing the technology aspect, we argued that using a specific technology *only* for marketing purposes is a

deceiving way of promoting an innovation. A use case can convey much more solid evidence of the merits of the innovation, and is, ultimately, a more ethical way of doing that, since it does not exploit people's irrationality.

Opinion leaders, by their very status, do not need to spend much effort in creating use cases, since their work will be taken seriously by everyone who is aware of their reputation. On the other hand, for most normal people, a great use case can be as important as the innovation itself, for getting others to give it fair consideration.

4 Discussion and Conclusion

The research question we set to answer at the beginning of this work was: *how can the authors of a research project in Computer Science improve their work to increase its adoption by like-minded creative individuals, and how is adoption affected by these actions?*

In light of the diffusion of innovation framework (Roger 2003), it is very important to reduce friction for adopters, so that they can adopt the innovation as easily as possible. We identified five categories of actions that the authors of an innovative project in Computer Science can take to reduce friction and increase the impact of their work, which are, in no particular order: open access, documentation, tutorials, technology, and use case. Table 1 summarizes the impact of these categories in the diffusion of innovation, with an *x* marking the cells where there is an effect.

One should not be tempted to simply count the number of *x*'s in each row and column to quantify the ease of affecting a certain attribute of the diffusion of innovation and the impact of the actions we defined on the diffusion. The reason for this is that the characteristics of innovation are not equally important for the diffusion, and the actions require different efforts to be put into practice; this means that every *x* has a different weight that should be considered in an aggregation. For example, an effective tutorial can often be produced in a few hours worth of work, therefore, even though from the matrix it appears to have a minor impact, its low cost makes it a worthy investment. Similarly, although the relative advantage is mostly conveyed through use cases, most

Table 1: What actions affect the innovation decision process and the characteristics of innovation, as defined in (Roger 2003).

	Open Access	Documentation	Tutorials	Technology	Use Case
Relative Advantage					*x*
Compatibility				*x*	
Complexity		*x*	*x*	*x*	
Trialability	*x*	*x*	*x*	*x*	
Observability	*x*				*x*
Knowledge					*x*
Persuasion	*x*				*x*
Decision	*x*	*x*		*x*	
Implementation	*x*	*x*		*x*	
Confirmation	*x*				

other actions offer opportunities to highlight the merits of an innovation over competing products. We refrain from defining these weights, and leave the issue open for further investigation.

Our work can be thought as siding with the pragmatic school of thought of Open Science (Fecher & Friesike 2014). Open Science advocates for open access to publications, data, code, education, and peer review; in this work we follow a similar line of thought, but do not argue about open education nor open peer review, as they are not relevant to our research question. The pragmatic view of Open Science states, in a nutshell, that the main purpose of Open Science is to enable more efficient creation and sharing of knowledge; as we argued, reducing friction is an effective way to foster the diffusion of innovation in the setting we considered in this work.

4.1 Assumptions and Limitations

An important assumption that we made is to consider the merits of the innovation itself fixed. Clearly, not all innovations are equal, and some are simply better than others. Moreover, we only considered the attributes in Everett's diffusion of innovation framework, but previous work has shown that other characteristics, such as functionality, performance, efficiency, perceived ease of

use and task productivity, may be more important; for example, Prescott & Conger (1995) highlighted the importance of these factors in the decisions of individuals belonging to the Information System unit of an IT business. However, the perception of these attributes can also be influenced by the categories that we considered in this work, and certainly interacts with the traditional characteristics. For example, performance and efficiency can be included in the relative advantage, and can be demonstrated in an use case.

The setting considered in this work does not include any business constraints to follow when evaluating an innovation, and focuses on small units of adoption with relatively simple mechanics. In more complex organizational settings, with IT innovations that are larger in scope and impact, the traditional diffusion of innovation theory is known to be inadequate (Lyytinen & Damsgaard 2001). Moreover, we only considered the innovation and decision aspects of Everett's framework, but the communication channels and social systems are also very important factors. For example, Lambiotte & Panzarasa (2009) studies the role of communities in social networks in the diffusion of information (and, therefore, innovation), and Yan & Gerstein (2011) studies how the awareness of a paper increases over time.

4.2 Why Share

A problem that is rarely considered in the Open Science literature is whether scientists will actually share their work or not. In an ideal world, sharing and openness are two basic pillars of science, but, in practice, this often does not happen. Haeussler (2011) showed that the willingness to share information is inversely proportional to the competitive advantage of such information, and that many factors related to the social capital of scientists, i.e. their connections with the community, affect information sharing practices. Andreoli-Versbach & Mueller-Langer (2014) confirmed that, although most scientists regard self-correction and replicability as very important, very few share their data in practice.

Good ideals are not enough to make scientists actually put in more effort than needed to share their research, and most of the actions that we discussed in this work do require a fair amount of

work to be put in practice. We cannot expect scientists to want their work to be adopted without proper incentives, and we agree with Fecher & Friesike (2014) in saying that Open Science requires an actual cultural change.

Because of the anecdotal nature of the results presented here, we stress the need of corroborating them with more solid evidence coming from qualitative investigations of this topic. Although previous works argued that the traditional diffusion of innovation framework is often inadequate, they were conducted in times where software was much less diffused than it is today. In the last decade, the adoption and usage of software changed dramatically, and the assumptions that were made in this work are now much less stringent than they were when those studies were conducted. Today, most software projects face fierce competition, and the playing field is much more even, with many alternatives of very similar quality.

Given that many guidelines are already being followed, to varying degrees, by many branches of computer science, there is ample opportunity for collecting and analyzing data related to this topic, by leveraging, for example, open source platforms such as GitHub[1].

References

Andreoli-Versbach, P. & Mueller-Langer, F. (2014), 'Open access to data: An ideal professed but not practised', *Research Policy* **43**(9), 1621 – 1633.

Fecher, B. & Friesike, S. (2014), *Open Science: One Term, Five Schools of Thought*, Springer International Publishing, Cham, pp. 17–47.

Haeussler, C. (2011), 'Information-sharing in academia and the industry: A comparative study', *Research Policy* **40**(1), 105 – 122. Special Section on Heterogeneity and University-Industry Relations.

Henderson, R. & Cockburn, I. (1994), 'Measuring competence? exploring firm effects in pharmaceutical research', *Strategic Management Journal* **15**(S1), 63–84.

[1] `https://github.com/`

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems 25', Curran Associates, Inc., pp. 1097–1105.

Lambiotte, R. & Panzarasa, P. (2009), 'Communities, knowledge creation, and information diffusion', *Journal of Informetrics* **3**(3), 180 – 190. Science of Science: Conceptualizations and Models of Science.

Lyytinen, K. & Damsgaard, J. (2001), What's wrong with the diffusion of innovation theory?, *in* M. A. Ardis & B. L. Marcolin, eds, 'Diffusing Software Product and Process Innovations', Springer US, Boston, MA, pp. 173–190.

Morin, A., Urban, J. & Sliz, P. (2012), 'A quick guide to software licensing for the scientist-programmer.', *PLoS Comput Biol* **8**(7).

Nakamoto, S. (2009), 'Bitcoin: a peer-to-peer electronic cash system'.

Prescott, M. B. & Conger, S. A. (1995), 'Information technology innovations: A classification by it locus of impact and research approach', *SIGMIS Database* **26**(2-3), 20–41.

Roger, E. (2003), *Diffusion of Innovations*, 5 edn, Simon and Schuster.

Rosenberg, N. (1974), 'Karl marx on the economic role of science', *Journal of Political Economy* **82**(4), 713–728.

Sorenson, O. & Fleming, L. (2004), 'Science and the diffusion of knowledge', *Research Policy* **33**(10), 1615 – 1634.

Yan, K. K. & Gerstein, M. (2011), 'The spread of scientific information: Insights from the web usage statistics in plos article-level metrics', *PLoS ONE* .