Master → Ph.D. transfer report

Method diffusion in large open-source projects

Martin F. Krafft <martin.krafft@lero.ie>

http://phd.martin-krafft.net

16 Nov 2007

Lero – the Irish Software Engineering Research Centre
CSIS, University of Limerick, Ireland
Under supervision of Prof. Brian Fitzgerald

Contents

1 Introduction

1.1 Motivation and objective

I have been involved with Debian for over a decade. At first, it was only a hobby, occupying a small fraction of my leisure time. Then, teaching Debian to others became my main source of income. Along the way, my fascination with the project kept building up. The Debian Project is a highly successful project, run entirely by over 2 000 volunteers: it produces several computer operating systems, the most popular of which provides over 20 000 software packages for eleven different hardware architectures. Furthermore, the Debian System is the basis for around 150 derivative distributions.

Despite these successes, the project is currently stagnating.[1] As I started to become more and more involved with the project, I ran up against walls and fell into holes that had started to appear over time. After 14 years years of age, it is starting to show that some of the project's processes did not scale to meet the tremendous growth the project has enjoyed since its inception in 1993. Endeavours, such as library transitions, currently require dozens of developers to work hand in hand, and often stall because of bottlenecks or lack of coordination. Similarly, day-to-day tasks, including package maintenance, consist of a dozen of tedious and thus error-prone tasks, which are anything but integrated.

It seems strange that developers of a system as technically sound and universally applicable as Debian are still doing by hand what the computer could be doing instead. Tasks like the aforementioned could be streamlined and optimised with tools geared towards distributed cooperation, if consistently used.

Some efforts in this direction already exist, but they are not being adopted by the Debian community as readily as they should be. Because the Debian Project is made up entirely of volunteers, noone can be told what to do or how to approach their

[1] This is not to say that the project is failing; rather, its processes are not as effective as they could be.

tasks. Instead, Debian Developers (DDs) prefer to choose their own methods, based on criteria which range from technical benefits to ideology, from pragmatism to sheer stubbornness. In addition, once settled, a developer is often unwilling to change his/her methods at a later point in time, or may only be able to do so through considerable additional effort and not be able to assess whether the effort will pay off.

Through my research, I strive to find a framework which can be used to assess diffusions of development methods in the Debian Project. The framework should capture factors which have an effect on the developers' decision to adopt or reject a development method. Given a technology which improves the workflow of an aspect of Debian development, the framework can be used to estimate the rate of diffusion of this technology within the project. If this rate is unacceptably low, the framework can offer suggestions on where to concentrate efforts to speed up the process.

I hypothesise that such a framework can be used to streamline Debian development and thus increase the efficiency of every contributor. Moreover, the framework should be applicable to other volunteer projects, and management science, where a trend exists towards treating employees as volunteers by giving them increased amounts of autonomy.

1.2 Outline

In chapter 2, I deliver a profile of Debian. While I could write volumes more, I confine myself to a short introduction (section 2.1) to explain the most crucial terms (section 2.2) before I sketch one of the core workflows in Debian, package maintenance (section 2.3); I highlight problems and offer hypothetical solutions along the way. In section 2.4, I touch upon the most prominent character traits of the stereotypical DD, which are undoubtedly similar to the traits of participants in other Free/Libre/Open-Source Software (FLOSS) projects, but are in many ways subtly different as the Debian Project focuses on quality and freedom more than any other project. The project has been the subject of a number of scholarly publications, which I collect in section 2.5.

Chapter 3 is devoted to the study of diffusions and split into two parts. I start with a short history and the development of the most relevant terminology (sec-

tion 3.1). The second half is concerned with frameworks: I identify the characteristics of the framework needed for my research and present a number of existing frameworks as a foundation for the development of a new framework, should that be needed.

Then, in chapter 4, I describe my research approach, which combines grounded theory (section 4.1.1) with a Delphi study (section 4.1.2). I partitioned the approach into four phases, which I present in section 4.2.

Finally, I offer a statement on research output to date (section 5.1) and outline my work plan throughout completion of my endeavour (section 5.2)

1.3 Acknowledgements

I would like to thank Gabriella Coleman, Assistant Professor at the Department of Media, Culture, & Communication of New York University, and Prof. Brian Fitzgerald of Lero, University of Limerick, for their valuable feedback on a draft of this report.

2 Debian

Debian is on of the largest FLOSS projects [Amor-Iglesias et al., 2005b]. As such, it has been subject of a considerable number of studies. In addition to scholarly works, vast volumes of text have been written on the subject in online publications, weblogs, and last but not least, publicly-archived mailing lists. The purpose of this chapter is to introduce Debian and one of its core workflows, and present an overview of existing literature.

2.1 Introducing Debian

Debian is primarily the name of a project, christened after its founder Ian Murdock, who is married to Debra, thus Deb-Ian (/'debiən/). The Debian Project is a globally-spread, non-commercial "association of individuals who have made common cause to create a free operating system" [Debian Project, 2006]. The project came to life 14 years ago to publish Debian GNU/Linux, the second oldest GNU/Linux distribution after Slackware Linux.[1]

The Debian Project is possibly the largest and most complex FLOSS project [Lameter, 2002, Amor-Iglesias et al., 2005b]: 1049 people from 52 countries are registered as official developers. In addition, an uncountable number of sponsored maintainers, translators, testers, and, last but not least, users provide their input and contributions around the clock. [2]

"Debian" is also used synonymously to refer to the concepts shared by a family of operating systems produced by the Debian Project. Debian GNU/Linux is the most prominent of these operating systems, and the Debian Project's main product: a distribution,

[1] The GNU/Linux distribution time-line: `http://futurist.se/gldt/` [26 Sep 2007]
[2] I estimate the number of two-or-more-times contributors to be around 2 000.

which integrates the Linux kernel, GNU user-space,[3] and Debian system administration concepts and techniques.

I have published a book on Debian, which documents these concepts and techniques. The book also sports chapters on the project, development techniques in use, and traits of its community [Krafft, 2005, chs. 1, 2, 4, 5, 9, 10]. For my dissertation, I research adoption behaviour among the group of developers and contributors, which my book does not cover. Nonetheless, it serves as comprehensive source of background information on the topic and portions of the following chapter are based on its contents.

Debian GNU/Linux is a *free* computer operating system. Debian takes an extraordinary and somewhat radical approach to Free,[4] which it documents in one of its fundamental documents, the Debian Free Software Guidelines (DFSG) (see section A.5), a part of Debian's Social Contract (see section A.4). Herein, the project promises "that the Debian system and all its components will be free according to these guidelines[, and that the project] will never make the system require the use of a non-free component." As I shall mention in section 2.4, this adherence to freeness and freedom is a fundamental motivation for the project's members.

To date, Debian GNU/Linux is made up of over 20 000 software packages, most of which are available for all of the eleven supported architectures. Amor-Iglesias et al. [2005a] find "Debian [to be] one of the largest software systems in the world, probably the largest."

2.2 Terminology

Please refer to section A.2 for additional terminology.

[3] GNU (GNU is not Unix) is a FLOSS project that pre-dates the Debian Project by a decade and strives to provide free, open-source implementations of the basic user-space applications, which hold the system together.

[4] The word "Free" is often, but not necessarily, capitalised to emphasise that the author wants to refer to both, "free as in beer *and* speech." In the Debian Project, the word is not capitalised, but still implies both meanings.

2.2.1 Developers and contributors

Traditionally, the Debian Project distinguished only between developers and users, while developers are also users. Over the years, as the project grew and more people contributed in an ever increasing variety of ways, additional roles came into existence [Wallach et al., 2005, Coleman, 2005b, Michlmayr, 2004]. Please refer to section A.1 for details on the evolution of membership in the Debian Project and the history of the various roles. For this text, I identify only the four roles immediately pertinent to the development of the system and consciously leave out infrastructural roles, such as the system administrators:

Users — include anyone who consciously uses the Debian System, and thus all members of the aforementioned categories.

Contributors — are users who contribute to Debian by way of bug reports, patches, translations, documentation, *etc.*. Often, these people do not want to be involved with the project, or take on responsibility, but see the value in their contributions. Such users usually install snapshots of the system from the unstable archive.

Maintainers — are those in charge of a package in the Debian archive, or a member of a team maintaining a package. Traditionally, official developers would act as "sponsors" and upload maintainers' packages to the archive; in 2007, the role of Debian Maintainer (DM) was instituted, allowing maintainers to upload their own packages themselves.

Developers — are official members of the project and have the rights and obligations as stated by the Debian Constitution (see section A.1.4), including access to developer machines, the ability to upload packages to the Debian archive, and the privilege to cast votes in elections and general resolutions.

Maintainers and contributors are also developing the Debian System and are also affected by workflow choices. Thus, in most cases, when I speak of developers, I mean to include the other two groups. In the rare cases that I need to differentiate among them, due to *e.g.* upload privileges, I shall take care to classify developers appropriately.

Note that these are labels for status within the project and thus cannot be directly mapped to the well-known onion model, [Crowston and Howison, 2003], which categorises users according to their contributions. Even though the Debian Project is meri-

tocratic in that it only awards status to those who have contributed, the transition to a more privileged status may take a long time (*cf.* section A.1.5); in addition transitions to a status with lesser privileges basically never happen [*cf.* Jensen and Scacchi, 2005]. The above categories are purely functional and relate to the privileges associated with each. Thus, it does not matter that membership of various groups (or layers) has been found to be in continuous flux [González-Barahona and Robles-Martínez, 2003, Ye et al., 2004].

2.2.2 Volunteers

The Debian Project is a project driven entirely by volunteers. The project does not pay any of its developers,[5] nor does it let its sponsors or its legal entity have any influence in the project's technical interests.

Robles et al. [2005b] offer a comprehensive definition of what it means to be a volunteer, which I assume: volunteers are those who work "in their free time not profiting economically in a direct way from their effort. Volunteers can be IT-related professionals or not, but their professional activity is not the one they perform" when they act as volunteers. By their definition, "all maintainers in Debian are volunteers. Some employers of people who act as Debian maintainers in their spare time permit their staff to devote some of their time to Debian during work hours. Nevertheless, the majority of work by most Debian maintainers is performed in their spare time."

2.3 Maintainer fundamentals, challenges, and solutions

It is outside the scope of this text to cover development practises of the Debian Project. However, since development practices and processes differ greatly across FLOSS projects [Michlmayr et al., 2005], a short discussion is in order.

My research is about assessing diffusions of methods within the Debian Project, with the goal of increasing efficiency of our workflows. I concentrate not on the increase of efficiency, but on the conditions under which a method will be widely adopted. The

[5] The Dunc-Tank experiment (`http://www.dunc-tank.org/` [7 Nov 2007]) is one notable exception, but the experiment is said to have failed.

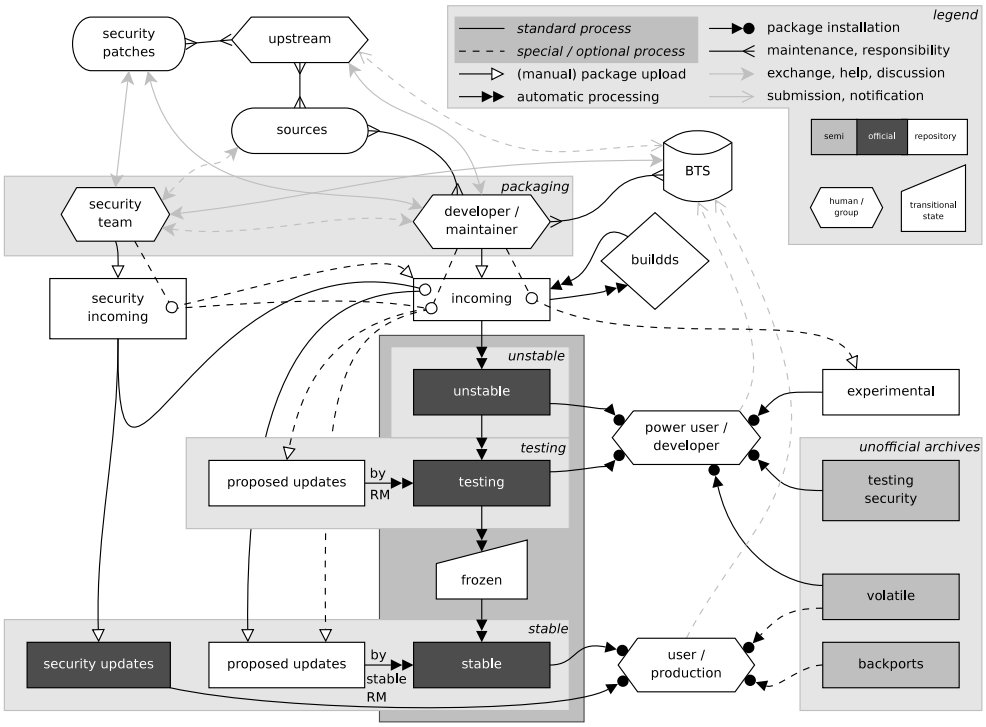

Figure 2.1: The life-cycle of a Debian package.

ratio between difficulty in understanding and putting a new method to use, and the benefits it brings will surely have an effect on the diffusion rate. In the following, I present a short run-down of the standard Debian package maintenance process as an example of a Debian development process in need of improvement. I also highlight some of the problems, and hypothesise solutions.

Figure 2.1 illustrates the various stages a Debian package traverses during its lifetime. A package is born when a user (or developer) finds (or writes) a piece of software, and wants to make that software available via the Debian archive. Through a process of "debianisation", the software is transformed to meet the requirements of the Debian System, and meta data are attached to ensure that the package fits in with all the other packages in the Debian archive (see section A.2.5). The package is then uploaded to the Debian archive, subjected to quality and licence checks, before it begins its lifetime as official Debian package in the unstable archive (see section A.2.4). The packager becomes the maintainer.

Every package in the Debian archive has an associated bug tracker.[6] As the package trickles through the archives and eventually becomes part of an official, stable Debian release, the number of users grows, and bugs will be found and reported. It is the maintainer's task to process these bugs, which can be either Debian-specific, or applicable to the upstream version[7] of the package.

In the first case, the maintainer might fix the problem, then prepare and upload a new revision of the package. In the package's changelog, s/he notes the fix and bug number, which causes the Debian archive management scripts to mark the bug report as done. To make life easier for other distributions, which may be tracking the Debian package, the maintainer could manually attach the patch fixing the problem to the bug report, but this is rarely done due to the extra effort involved; unfortunately, it cannot be automated reliably, a point to which I shall return shortly.

If the maintainer runs out of time (or is lazy), someone else may pick up on the bug report and work out a fix, attaching the patch to the bug report when done. Depending on the severity of the bug report, this person may then prepare a non-maintainer upload (NMU) to solve the problem in the archive as quickly as possible. In such a case, the patch may never end up in the bug log, as the act of appending the patch to the report as part of a NMU is requested, but not enforced. Once the maintainer again finds time for the package, s/he needs to retrieve the patch from the Bug Tracking System (BTS) and apply it, then record the act in the changelog before uploading the new package revision. Problems arise when a maintainer had completed half of the fix when time ran out, and then receives a patch from someone else, who was unaware of the maintainer's local efforts.

If the reported problem applies to the upstream package, the maintainer may similary fix the problem in the Debian package. It is then in the maintainer's interest to forward the patch to the software's author, so as to minimise the volume of differences s/he has to track between the original software and the version published in the Debian archive (patch management). Nevertheless, the maintainer will have to track the patch until a new upstream release incorporates it. Some patches against the upstream code may be Debian-specific and thus have to be tracked indefinitely, as they cannot be submitted to upstream.

[6] `http://bugs.debian.org/`*`packagename`*

[7] Upstream refers to the (external) source of software packaged for Debian; *E.g.* the GNOME project is Debian's upstream for GNOME-related packages. Debian users, as well as derived distributions, are sometimes referred to as downstream.

What this outline strives to illustrate is the complexity of the process and the number of related but separate tasks awaiting the maintainer in response to every bug report. Current Debian package maintenance involves many repetitive tasks, which hold a high potential for human error. In addition, as the complexity of a package increases (due to large numbers of bug reports and a slow upstream, or related to fundamental differences between the upstream software layout and Debian's standards), package maintenance becomes tedious and error-prone, negatively affecting the quality of the entire distribution.

In an ideal world, package maintenance, patch management, and bug reports would be integrated in such a way that patches are treated as separate, logical entities stored within the package. Each patch could be manipulated (applied, forwarded, removed) separately, and status changes would be automatically recorded in the associated bug report.

Using version control systems (VCSs) for package maintenance is a strong trend in the Debian Project, but without any standard practices; or rather, in the spirit of Grace Hopper: "the wonderful thing about standards is that there are so many to chose from."[8] This is not necessarily a bad thing, as competition fosters quality, but it dampens cross-package collaboration and hinders one-time contributions: users trying to fix bugs in a package as part of a bug squashing party (BSP) may have to first learn how to interact with the VCS. It also makes it difficult to automate the aforementioned integration of the BTS with the VCS.

This situation is a core motivation for this research. I have postulated improved workflows[9] and am part of efforts to integrate VCS with the standard Debian tools.[10] However, while availability of a solution is a prerequisite for its adoption, considerations are in order to drive the diffusion of a solution in the Debian Project; an assumption that any method, however advanced or appropriate, will be widely adopted in the project would be naïve, as it does not take into account the characteristics of the Debian Project, its community, and its members.

[8] This quote is widely attributed to Grace Hopper, but first appears in Andrew Tanenbaum's *Computer Networks* without attribution.

[9] `http://blog.madduck.net/debian/2005.08.11-rcs-uploads` [17 Oct 2007] and `http://blog.madduck.net/debian/2007.10.03_packaging-with-git` [17 Oct 2007]

[10] `http://kitenet.net/~joey/blog/entry/an_evolutionary_change_to_the_Debian_source_package_format` [8 Nov 2007]

2.4 Stereotypical traits of Debian developers

Coleman [2005b], by way of three examples, illustrates how Levy's seven infamous hacker ethics[11] [Levy, 1984] have been embodied and furthered by the Debian community, and the importance of acknowledging how these ethics evolve in new contexts, such as the Debian Project. For instance, Levy's "true hackers" did not have a concept of legality, which is a strong point of focus in the Debian Project (*cf.* section A.4).

When researching character traits of participants of FLOSS projects, one encounters the concept of developer motivations [Raymond, 1999, Feller and Fitzgerald, 2002, Weber, 2004]. In the context of ethics, Coleman [2005b] offers an alternative perspective: "developers commit themselves to an ethical vision *through*, rather than prior, to their participation in a F/OSS project" [[]her emphasis]p. 319 This (powerful) argument makes it possible to lump DDs together for common coverage of their character traits, while allowing for exceptions to be mentioned.

In my research, I concentrate not on ethics but on productivity and efficiency, and thus refer to Turkle [1984] for further treatment of the semi-obessive traits of hacker. Also, I touch the subject in [Krafft, 2005, p. 46ff]. The following is a very brief discussion of the most prominent traits, relative to my research.

I would like to partition the set of all DDs (including maintainers and contributors) into two classes: developers interested in performing their own tasks, and only these, and developers interested in the technical side of the project as a whole. By far the largest number of DDs falls into the former class, while only a single-digit percentage[12] care about the development processes in use throughout the project.

A DD is a volunteer (see section 2.2.2) and as such cannot be told what to do [Robles et al., 2005b]. On the other hand, many DDs, especially those in the latter class, are interested in new technology, and improving the efficiency of the workflows they

[11] (1) Access to computers – and anything which might teach you something about the way the world works – should be unlimited and total. (2) Always yield to the Hands-On Imperative. (3) All information should be free. (4) Mistrust authority – promote decentralization. (5) Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race or position. (6) You can create art and beauty on a computer. (7) Computers can change your life for the better.

[12] This number would have to be determined by the survey proposed in section 4.2.2; I currently estimate the number at around 200.

follow, and are likely to consider new approaches, sometimes even excited to follow the lead of others.[13] The distribution across adopter categories (see 24) seems to be similar to the spread identified by Rogers [2003, p. 281],[14] meaning that only a very few DDs innovate, while the majority tends to wait for the early bumps and holes to be fixed before adopting. Even though adoption tends to be slow in voluntary projects (*cf.* Rogers [2003, p. 29f]), levels of awareness and interest of a new technology tends to be high much earlier among members of the Debian Project. This is to say that most Debian contributors tend to spend a considerable time in the persuasion stage (see section B.1). The following quote is reminiscent of typical reactions in discussions about innovations in the Debian Project:[15]

> "I find your approach very interesting but probably not worth the trouble at the moment. Once the early adopters such as yourself have come up with a streamlined workflow for keeping Debian modifications in branches, I'll reconsider."

A related concept is absorptive capacity (AC) — the ability to recognise, assimilate, and apply external information to internal processes [Cohen and Levinthal, 1990, Zahra and George, 2002]. The Debian Project's AC depends on the AC of its members, which is limited mainly by the fact that contributors want to contribute, instead of learning new methods Stuermer [*cf.* 2005]. While in the business domain, where the concept of AC originated, a firm's innovativeness is vital and AC can decide between failure and success, but this is not the case in a volunteer project, such as the Debian Project, which exists without competition.[16]

For developers, a much more pressing concern is the perceived diminishing return from investing time into new methods. DDs are increasingly more aware of the problem of "yak shaving"[17] — spending (more) time on tools which can be used to solve a task, instead of simply (spending less time) getting the task done. Being aware of the

[13] *cf.* `http://lists.madduck.net/pipermail/vcs-pkg/2007-October/000039.html` [25 Oct 2007]

[14] Again, this is something to be determined by the survey of section 4.2.2.

[15] `http://lists.madduck.net/pipermail/vcs-pkg/2007-October/000032.html` [10 Oct 2007]

[16] This is not to say that Debian stands above its potential competitors, just that there are no hard facts, such as turnover or market share. Instead of money, the Debian Project is run on volunteer time and monetary donations, instances of neither of which are essential for its survival [*cf.* Robles and Gonzalez-Barahona, 2006].

[17] A true gem of the Internet: `http://projects.csail.mit.edu/gsb/old-archive/gsb-archive/gsb2000-02-11.html` [7 Nov 2007]

problem does not mean that an individual can avoid it and be pragmatic at all times.[18] It remains to be determined, possibly by survey (see section 4.2.2), how this problem affects adoption behaviour among DDs.

```
< madduck> is there a term for wasting time playing with and improving
           the tools one uses in accomplishing a task, instead of
           accomplishing the task?
< a_developer> I know exactly what you mean. fiddle with the tool for
               4 hours instead of just doing the 4-minute task
< another_dd> most of us do
< yet_another_dd> it's yak shaving
```

— #debian-devel, 7 November 2007

A trait shared by contributors to the Debian Project is the appreciation of technical excellence, which is often a source of "yak shaving." In my book, I described Debian poetically as "academically-inspired applied functionalism" [Krafft, 2005, p. 32] suggesting that Debian developers approach problems patiently and academically in search of solid, long-term solutions. Yet, the solutions have emerged out of the practical needs of Debian users. Consequentially, Debian's solutions are far more powerful and robust than needed for most situations, but these tools can be put to use in standard and complex scenarios alike. This is one of the reason why we call Debian the "universal operating system." In addition, high levels of peer review ensure code of outstanding quality [Michlmayr, 2005b], and Debian is a quality-oriented project without fixed release cycles or other forms of market pressure [*cf.* Halloran and Scherlis, 2002]. This orientation, I postulate, is one of the motivations for contributors to continue investing time into Debian. And this orientation may directly translate into the adoption development methods: if DDs prefer solid solutions and are not driven by market pressure, adoption behaviour within the group is likely to be more conservative.

Another motivation for participation is Debian's rigorous observance of freedom. The DFSG (see section A.5) express the project's dedication to building a complete operating system based entirely on free software. While one of the characteristics of a FLOSS project is the freeness of its produce, Debian GNU/Linux is the only GNU/Linux distribution which is built on a strong belief in freedom and does not suffer from licencing restrictions, which could restrict its users. Here, I can make the link back to the third of

[18] In fact, I hypothesise that much of Debian's core is affected by the issue in one way or another, and if it's only the distractions on IRC channels, which are central to Debian development, but also very "chatty."

the hacker ethics identified by Levy: all information should be free. This belief equally unites contributors to Debian [Coleman, 2005b, p. 3f21].

Other ideological effects relate to programming language and logic, and coding style preferences, which have been influential in an individual's decision to favour or disfavour a given solution.

2.5 Literature review

As stated in the introduction, Debian has been the topic of many investigations and articles. By far, most of the coverage comes via electronic publications, mailing lists, and Internet discussion forums. As one of the largest FLOSS projects, however, it has also been subject of books,[19] and a fair number of scholarly articles.[20]

Most commonly, the project is cited for its strict adherence to moral, ethical, and ideological standards [Raymond, 1999, Feller and Fitzgerald, 2002, Coleman, 2005a], and on licencing topics [Garzarelli and Galoppini, 2003]. It has also been studied in the context of project success measurements [Senyard and Michlmayr, 2004, Crowston et al., 2006], project management [González-Barahona and Robles-Martínez, 2003, Garzarelli and Galoppini, 2003, O'Mahony and Ferraro, 2004, Michlmayr, 2004, Robles and Gonzalez-Barahona, 2006, O'Mahony and Ferraro, 2007, Robles et al., 2007], release management [Michlmayr, 2005a, 2007, Michlmayr et al., 2007a], and quality assurance [Michlmayr et al., 2005, Michlmayr and Senyard, 2006]. The volunteer nature of the project has also received attention [Michlmayr and Hill, 2003, Michlmayr, 2004, Robles et al., 2005b, Robles and Gonzalez-Barahona, 2006, Robles et al., 2005c, Michlmayr et al., 2007b]. Robles et al. [2006b] have explored social network analysis techniques for community-driven software projects and used Debian as case study, and Sowe et al. [2006, 2007] study Debian's social network with a focus on knowledge brokerage.

The Debian package archive has served as basis for network analysis studies [Fortuna et al., 2007, Robles et al., 2006b], statistics [Robles et al., 2006a, Amor-Iglesias et al., 2005a, Robles et al., 2005a], and effort analyses [González-Barahona et al., 2001].

[19] `http://debian.org/doc/books` [8 Nov 2007]

[20] I have made efforts to bring together researchers of Debian with a mailing list: `http://lists.madduck.net/listinfo/academic-debian`

Thanks in large due to the high quality meta data found in the archive, Debian is beginning to gain momentum in software product line studies [Kojo et al., 2003, Bermejo and Dai, 2006].

To my knowledge, no articles exist on the Debian workflow, or method diffusion/adoption within the project.

3 Assessing diffusions

In Friedrich Dürrenmatt's satire *Die Physiker* (*the physicists*), the central character Möbius hides in a sanatorium in a hopeless attempt to keep his invention, the "system of all possible inventions," from the public, in wise anticipation of the potential of its abuse. The story unfolds to reveal his failure, building up to the climax moment when Möbius appeals to the public to consider the consequences of scientific research, because "what has once been thought can never be unthought."

Ideas spread. If an idea is widely accepted, it propagates faster from one person to the next; the process underlying this propagation is called diffusion. Not only ideas diffuse, but also products, strategies, theories, and the many other types of innovations, and diffusions are studied in numerous fields: politics, marketing, process improvement, and engineering, to name but a few. It is thus unsurprising that the study of diffusions, despite its young age, has received considerable attention and continues to be a busy research domain with a large volume of output.

The purpose of this chapter is to build a basis for the assessment and comparison of diffusions of methods among Debian developers. I start by developing a terminology used henceforth. Next, I define the requirements of the framework I seek for the research related to this thesis. I then briefly present a number of existing frameworks that have been developed to assess diffusions.

3.1 History and terminology

The origins of the study of diffusions are found in Gabriel Tarde's book *The Laws of Imitation* [Tarde, 1903]. Tarde, a French sociologist and social psychologist, who views society as made up of individual interactions between people, claims that "society is imitation" [*ibid.*, p. 74] of peers by peers. Core to Tarde's theory is the search for traits that determine the successful spread of an object of imitation, also referred to as innovation:

> "Our problem is to learn why, given one hundred different innovations conceived of at the same time — innovations in the form of words, in mythical ideas, in industrial processes *etc.* — ten will spread abroad, while ninety will be forgotten" [Tarde, 1969, p. 140].

To date, more than 5 000 research papers have appeared on diffusions [Rogers, 2003, p. xviii]; while almost all of these studies treated the respective diffusion as separate and did not consider potential insights gained from other diffusions, Rogers [1962] was the first to present a concerted approach (see section 3.2.2).

In the following sections, I introduce the core terms of innovation diffusion research. The terminology of the domain is very convoluted and definitions from business and marketing usually contradict their counterparts from the social sciences. Given the ubiquity of marketing terms, thanks to today's media, I deemed it appropriate to highlight the differences before taking a stand. Rogers [2003] is widely cited and accepted as as authoritative reference on the topic of innovations in the social sciences. A scholar of communication and technology adoption studies, he gained interest in the diffusion of innovations through exposure to rural sociology, and has since studied several thousand diffusions in copious different fields. While it is tempting to settle for his terminology, alternate definitions are occasionally more appropriate to this research.

3.1.1 Innovation and invention

Both in academic and industry circles, innovation is a buzzword with an unclear meaning. The word is overused for its seemingly positive connotation, when in fact is does not imply success, and politicians and marketing people are especially fond of its (ab)use. Scholars, dictionaries, and the media disagree over the actual meaning. For instance, the Oxford English Dictionary[1] double-defines innovation as (1) the action or process of introducing new methods, ideas, or products; (2) a new method, idea, product, *etc.*. As I shall show shortly, the social science literature takes a completely different perspective. Pierce and Delbecq [1977] provide an extensive treatment of the two words. The following concentrates on the aforementioned difference, which has been the source of confusion I most commonly encountered.

[1] `http://www.askoxford.com/dictionaries/compact_oed` [31 Oct 2007]

The crux of the disagreement may be the word new, and a related confusion over the use of invention *vs.* innovation: very often, an invention is referred to as an innovation. An invention is generally understood as something novel in the sense that it has never before existed. It is often the result of research and development [Fagerberg, 2004]. Kline and Rosenberg [1986, p. 279] acknowledge that an innovation is often perceived as a new product (an "invention"), but offer alternative interpretations: new processes, substitution of component materials in production, reorganisation of production leading to increased efficiency, and improvements in "instruments or methods of doing innovation"[2] [*ibid.*].

According to Schumpeter [1942, p. 89], an economist, an invention is economically irrelevant as long as it is not carried into practise. Earlier, Schumpeter [1934, 1939] had defined innovation as the commercial or industrial application of something new, thus suggesting that an innovation is what makes an invention economically relevant. Schumpeter does not take a clear side on what it means to apply or carry into practise an invention. In fact, his two definitions can be read to conflict with each other: a product is carried into the market by a marketing department, but applied by the end-user. The business literature seems to push innovations to the people, where an innovation is the "first commercialization of the idea" [Fagerberg, 2004]. The social sciences centre on the implementation; Amabile et al. [1996], who study organisational innovation studies, define innovation as "the successful implementation of creative ideas within an organization."

The adjective innovative is closely related. In the business world, companies (and even ideas) are often labelled innovative [*e.g.* Lazonick, 2004],[3] while in the social sciences, innovativeness is a feature of users who implement an innovation. The verb innovate and the lesser used noun innovator (someone who innovates) take similar meanings in the two domains: businesses are innovative if they offer new products or ideas. An organisation, studied by social science scholars, is innovative if it implements new products or processes.

By the definition of Rogers [2003], an innovation is only a perceived novelty, without any connotation of implementation or success [*ibid.*, p. 12]:

> An innovation is an idea, practice, or object that is perceived as new by an individual or other unit of adoption. It matters little [...] whether or

[2] For an explanation of what this cyclical definition means, please see footnote 2.
[3] Any newspaper's business section uses the (buzz)word in this sense.

> not an idea is "objectively" new [...]. The perceived newness of the idea for the individual determines his or her reaction to it. If an idea seems new to the individual, it is an innovation.
>
> Newness in an innovation need not just involve new knowledge. Someone may have known about an innovation for some time but not yet developed a favorable or unfavorable attitude toward it, nor adopted or rejected it. "Newness" of an innovation may be expressed in terms of knowledge, persuasion, or a decision to adopt.

Later in his tome, he explicitly defines the innovator (who innovates) as the person putting an innovation to use: "the innovator must be able to cope with a high degree of uncertainty about an innovation at the time he or she adopts" [*ibid.*, p. 282]. In comparison, Hughes [1989], who takes the perspective of marketers, claims that innovators "strive to increase the size of the system under their control and to reduce the size of the environment that is not" [p. 66]. In this context, I can see potential for a case that those first to adopt a technology may themselves be interested in marketing it, *i.e.* to get others to follow: especially in the case of innovations with complex network externalities (see section 3.1.3), an innovator will strive to increase his/her own utility by getting others to adopt.

3.1.2 Diffusion and adoption

Diffusion is about communication [Mahajan et al., 1990]. Rogers [2003] defines the diffusion of an innovation to be "the process in which an innovation is communicated through certain channels over time among the members of a social system. [...] Diffusion is a kind of social change, defined as the process by which alteration occurs in the structure and function of a social system" [p. 5f]. By this definition, a diffusion is always about an innovation, and hence "diffusion" by itself can be used in place of "diffusion of innovations."

The rate of diffusion is thus the speed of communication, or the rate with which ideas are passed from one member to the next. A diffuser is an individual or agency driving a diffusion. Thus, a company's marketing department, so-called change agents, individuals, and whole groups can take on the role of a diffuser.

In the marketing literature, diffusion is mostly assessed by the degree of acceptance of an innovation by the market collective [*e.g.* Rogers, 1976, van den Bulte, 2000, Ma-

hajan et al., 2000]. Here, a diffusion is successful if it yields widespread adoption. In this case, the term adoption refers to the psychological process leading up to a consumer's individual decision to accept an innovation [Ozanne and Churchill, 1971]. The marketing literature seems to be mostly concerned with successful diffusions, meaning diffusions that elicit adoptions.

Rogers offers a broader perspective, defining adoption to be one of the possible outcomes of what he calls the "innovation-decision process" [*ibid.*, ch. 5]: "adoption is a decision to make full use of an innovation as the best course of action available" [*ibid.*, p. 177]. Different from the marketing literature, his model also accomodates failure (rejection); Acceptance is sometimes used in place of adoption, especially in the context of the Technology Acceptance Model (TAM) (see section 3.2.2).

Potential adopters are those confronted with an innovation and who are not unlikely to adopt it; the subset of people accepting the innovation are the adopters. Rogers separates the members of the social system into five categories, according to their innovativeness: innovators are the first (2.5%), followed by early adopters (13.5%), the early majority (34%), the late majority (34%), and finally, the laggards (16%) [*ibid.*, p. 281ff]. Moore [1991] concentrates on the divide between early adopters ("visionaries") and the early majority ("pragmatists") and how to overcome it.

Adoption is not an act, but rather a process undergone by an adopter. Rogers splits the process into five stages [*ibid.*, ch. 5], which he derives from earlier research; as part of this research, he also validates the existence of stages in the adoption process [Beal et al., 1957]. Kwon and Zmud [1987] offer a more granular break-down of the same process into six phases.

Diffusions can be successful or unsuccessful. A common way to measure the success of diffusions is the rate of adoption, which Rogers defines as the "relative speed with which an innovation is adopted by members of a social system, [...] usually measured by the length of time required for a certain percentage of the members of a system to adopt an innovation." [*ibid.*, p. 23]. A diffusion may also be considered successful if a critical mass of adopters is reached [*cf.* Markus, 1990], which is the point when the number of adopters is high enough such that the rate of adoption starts to sustain itself. Once this point is reached, a diffusion is said to be irreversible [*ibid.*, p. 343ff].

Tornatzky and Klein [1982], Kwon and Zmud [1987], Fichman [1992], Swanson [1994] provide an extensive overview of diffusion research in the information systems (IS) domain.

3.1.3 Network externalities and excess inertia

The study of diffusions distinguishes between two types of innovations: those which can be freely adopted by members of a social system, without affecting any other members, and innovations, whose singular adoption has an effect on the rest of the social system.

The literature refers to effects of an innovation on the rest of the social system as network externalities, a term first used by Farrell and Saloner [1985] and Katz and Shapiro [1985]. The term[4] combines the meaning of network effects, commonly used in the communications field [Rohlfs, 1974], and externalities, an economic concept defined as "an effect of a purchase or use decision by one set of parties on others who did not have a choice and whose interests were not taken into account."[5] The literature does not offer a clear definition, but the Internet does: "Network externalities are the effects on a user of a product or service of others using the same or compatible products or services."[6]

One example of a (positive) network externality is the choice of media format of a video player: the consumer of a more popular format will benefit from a greater selection of products, because manufacturers are more likely to produce using the format that is more popular [Katz and Shapiro, 1986].[7] Markus [1990] develops a critical mass theory (of interactive media) in this context, which accounts for one technology (or medium) eventually displacing another (of lesser popularity).

Network externalities are not limited to positive effects: "If, for example, a telephone or computer network becomes overloaded, the effect on an individual subscriber will be negative" [Liebowitz and Margolis, 1994]. A decrease in utility with increasing number of users is called a negative network externality.

[4] Leibenstein [1950] coined the term bandwagon effect as "the extent to which the demand for a commodity is increased due to the fact that others are also consuming the same commodity."

[5] `http://economics.about.com/cs/economicsglossary/g/externality.htm` [1 Nov 2007]

[6] `http://economics.about.com/cs/economicsglossary/g/network_ex.htm` [1 Nov 2007]

[7] Although Katz and Shapiro [1986] use the example of VHS *vs.* beta video cassettes, which are nowadays antiquated, we are faced with the same dilemma choosing between HD-DVD and Blu-Ray media today.

Other negative effects of network externalities relate to the phenomenon of inertia. For example a larger number of users of one technology may prevent others from switching to another (possibly superior) solution because of the need to cooperate with those who use the current product. Farrell and Saloner [1985, 1986] call this effect "excess inertia." To make the characteristics of this form of inertia explicit (see below), I refer to it as excess lock-in inertia; Strader et al. [2007] call it cross-impact network externalities. Examples of such cases are numerous, especially in the information technology (IT) domain, and often involve (products by) monopolists, or years of established practise before a new product comes along: office suites, programming languages, version control systems, and sundry other productivity and development tools.

Network externalities can also work the other way and force users to give up their established practise in favour of another technology. The literature seems to provide no concrete examples, but the case of Microsoft Internet Explorer illustrates the point: after displacing Netscape's browser,[8] Internet Explorer became the *de-facto* standard of the World Wide Web, causing many designers and application providers to cater their sites for the browser by employing non-standardised extensions. In recent years, other browsers have regained market shares. Nevertheless, there continue to be sites which require the use of Internet Explorer and thus force users to employ that browser, even though they have long abandoned it. Internet Banking applications are but one such example. For lack of an existing term, I refer to such externalities as excess pull-back inertia.

3.1.4 Attributes, characteristics, facets, traits

An entity, such as an innovation or a diffusion, may have certain attributes which are characteristic of the entity. I use the words attributes, characteristics, facets, and traits synonymously.

3.2 Diffusion frameworks

When assessing or comparing diffusions, it is paramount to agree on a common set of comparable, normalised attributes, or else one might compare apples with oranges.

[8] Microsoft leveraged the ubiquity of its operating system to force diffusion of its browser, which later resulted in an anti-trust case.

Such a set of attributes, along with their domains and relation to each other is called a framework. The Oxford English Dictionary[9] defines a framework as "a supporting or underlying structure."

In this thesis, frameworks are used analytically to compare diffusions. An analytic framework is a skeleton which allows the characteristics of diffusions to be framed and aligned to facilitate comparisons: it is easier to compare the performance of two stocks with two data sheets issued by the same bank (and thus using the same presentation framework), than with separate data sheets using different frameworks, which present disjunct sets of characteristics.

A framework can be thought of as a template, with predefined fields to structure the data to be analysed or compared. A template is instantiated by populating these fields. The result is a representation of the data in form of an instance of the template, which can be compared to other instances.

The ideal diffusion framework references the attributes of diffusions whose domains are orthogonal: the value of any one attribute can change without affecting the values of the other fields. Furthermore, orthogonality in attributes reduces redundancy. Orthogonal attributes are sometimes referred to as principal components.[10]

The concept of a model is related to frameworks. The literature uses these words interchangeably. For instance, Venkatesh et al. [2003] provide a comprehensive overview of "models and theories of individual acceptance," all of which are labelled frameworks by their respective authors. I follow the literature and use the two words synonymously.

3.2.1 In search of a framework

The framework I seek to propose with my research needs to be able to capture the attributes of diffusions relevant to the Debian Project, specifically the social system of DDs (and contributors). Thus, it must pay attention to the traits of developers (see section 2.4) and incorporate the complexity of the project.

[9] `http://www.askoxford.com/dictionaries/compact_oed` [29 Oct 2007]

[10] The statistical technique "principal component analysis" is also known as "proper orthogonal decomposition."

The Debian Project is an organisation. Thus, a diffusion framework will need to cater to aspects of organisational diffusion and adoption [*cf.* Swanson, 1994], and respect the high level of interdependencies and network externalities any development method used in Debian will have to address. Furthermore, while the social setting is the same for any diffusion in the Debian Project, its countless subprojects and teams form individual social environments. Part of the considerations relating to the social setting are the various communication media (e-mail, mailing lists, discussion forums, the BTS, Internet relay chat (IRC), Web logs and the Web in general, and live meetings between developers at conferences and BSPs. Furthermore, the high volume of communication and related issues with information overload play a significant role that must not be underestimated. Finally, community traits and rules of conduct also need to be considered.

On the individual level, several factors are relevant to diffusions, including, but not limited to, ideology, pragmatism, absorptive capacity, and motivation. Opinion leaders, who are mainly those further up on the meritocratic chain, are an important asset in any diffusion, while change agents are less applicable. After all, the volunteer nature of any contributor is possibly the most important concern.

3.2.2 An overview of existing frameworks

Already in 1977, Pierce and Delbecq, scholars of organisational theory and management, reported that "during the past two decades a number of theoretical models of organizational innovation have appeared in the literature." Unfortunately, the situation has not improved today, as plenty of additional frameworks have appeared. Tornatzky and Klein [1982] seem to be the first to recognise that most of these frameworks overlap and tried to integrate them into a consistent framework. Few years later, Kwon and Zmud [1987] (again) realised that the models of IS implementation were (still) fragmented and made an attempt to unify them. In recent years, several other meta-frameworks have appeared, but failed to build up on each other and thus fragmenting the space even more [*e.g.* Frambach and Schillewaert, 2002, Wejnert, 2002, Fichman, 2004].

To my knowledge, no framework exists to structure diffusions in FLOSS projects. Chau and Tam [1997] provide a framework for the diffusion of open systems within businesses, which is the opposite of what I need: it assesses the adoption of open systems within authoritarian structures, not within FLOSS projects.

In the following sections, I briefly present a selection of frameworks, starting with the two best-known: Rogers' diffusion of innovations, and the TAM, neither of which seem suitable for the study at hand. I then describe two frameworks which I have found to be more promising, but as formulated, they may also not be suitable for my research; they could, however, serve as a good foundation for the framework I am designing.

In the following, it is important to keep Kuhn [1970] in mind, who postulates that one does not abandon a model just by thinking about it. Instead, one abandons a model when a better one has taken its place. Analogously, I cannot at this stage discard any model in favour of another, because I do not have a better model to take its place, yet. Even though after surveying the field, I am lead to believe that a new framework needs to be developed, the outcome of the Delphi study (see section 4.2.3) may be an existing framework. Thus, my intention is to work from the bottom up and then determine whether an existing model fits, rather than trying to apply different models in turn.

Rogers' theory of diffusion of innovations

Rogers [1962][11] is often cited as default literature for studies of diffusion of innovations. Following a cooperation with Beal and Bohlen [Beal et al., 1957], Rogers conceptualised a generalised model of diffusion. Around 1960, the number of publications on all kinds of diffusions increased sharply after independent studies suggested that the diffusion process was independent of communication infrastructure and seemed cross-culturally similar [Rahim, 1961, Deutschmann and Borda, 1962, cited in Rogers [2003]]. While almost all of these studies treated the respective diffusion as separate and did not consider potential insights gained from other diffusions, Rogers [1962] was the first to present a concerted approach towards the study of diffusions. His tome "summarized research findings to date, organized around a general diffusion model, and argued for more standardized ways of adopter categorization and for conceptualizing the diffusion process."

Drawing on over 3 000 studies in sundry domains, Rogers enriched and validated his findings and divided each of these elements into component attributes (or stages, as in the case of the third element), characteristics of diffusions that determine (or help to

[11] currently available as fifth edition [Rogers, 2003].

explain) their success. Rather than objective truths, these focus on perceptions and experiences by the potential adopters; according to Rogers [2003, p. 223], it is the "subjective evaluation [which] drives the diffusion process [...]."

Rogers' theory of diffusions is based on numerous generalisations, that he formulated during the course of his extensive work with agricultural innovations. Fichman [1992] identifies the following five generalisations as the most relevant:

- Innovation characteristics, as perceived by the adopters, determine the rate and pattern of adoption.
- Some adopters' are more innovative than others, and these usually have different personal characteristics than the laggards.
- The adoption process can be broken up into a series of 4-5 stages, and adopters are subjected to different influences in each stage.
- Certain individuals can influence others and hence influence adoptions (change agents and opinion leaders).
- The rate of adoption is an S-shaped curve when plotted over time, as is the cumulative number of adopters.

From these generalisations, Rogers identified four "elements" of diffusion:

1. Attributes of the innovation
2. Communication channels
3. Time
4. Social system

I briefly summarise these in section B.1; an elaborate summary is provided by Raghavan and Chand [1989].

Criticisms of Rogers' framework Rogers' framework has been validated by thousands of diffusion studies. It may thus seem a little far-fetched to speak of criticisms, but in the context of this thesis, the framework is not without limitations. Many of my criticisms may be rooted in its generality. Rogers' framework has been successfully applied in studies of fertility-control methods, political reforms, policy changes, agricultural practises, rural development, technology and IT, health, education, and many

more.[12] With such a broad range of applications, the framework has to be a loose fit in any single case.

In the following, I illustrate some of the limitations, specifically with respect to the requirements of the current study, for which I seek an analytic framework to compare diffusions within one and the same environment.

Static attributes The fourth element of Rogers' framework captures the social characteristics of the environment, which are likely not to change significantly across diffusions in the same environment. Furthermore, while the attributes of the communication channel may appear differently to any two adopters, they describe the communication patterns within the social system, which are slow to change.

Orthogonality A significant problem are overlaps between the attributes of the framework. Even though it is likely impossible to construct a completely orthogonal framework for qualitative data.[13] Rogers' framework lacks orthogonality, which may be a function of its generality.

For instance, the compatibility attribute of an innovation, which references social norms, is almost entirely a subset of "norms of the social system," another attribute in the framework. Similarly, the fourth element includes the communication structure of the social system, while the entire second element is devoted to communication. One may argue that in those two examples, one of the overlapping attributes merely expands on the other, providing more details, but then the framework still suffers from the problems related to redundancy of information.

Furthermore, if you compare just pairs of innovation attributes, the overlap is more striking: low levels of observability might cloud the assessment of an innovation's relative advantage, and high trialability might be annulled by high complexity.

Attribute domains The attributes of each of the four elements differ in their nature and domains, some of which do not render themselves for comparison. For instance, while each of the characteristics of an innovation can be represented linearly

[12] I refrain from citing representative publications, which would bloat the bibliography.

[13] In statistics, principal component analysis is a rigorous technique to reduce and reorder the dimensionality of a (quantitative) data set, but it only finds optimal solutions, *i.e.* solutions with the least amount of overlap between those dimensions. It is doubtful that one can come close to such results with qualitative data.

(*e.g.* high *vs.* low degree of relative advantage), the characteristics of the communication process are more complex: both types of communication channel have their merits, and the best adoption rates are exhibited by groups that are neither purely homophilous nor heterophilous, nor does either of them imply higher adoption rates than the other.

Variance vs. process theory While the first and fourth elements of Rogers' framework seem like candidates for analysis by variance theory, and the third element renders itself more for process research [*cf.* Mohr, 1982], it is unclear which approach to take for the second element; communication is a process [Dance and Larson, 1976], but Rogers identifies strongly interdependent variables of the communication channel in his second element of diffusion, rather than focusing on the process of communication over time.

A solution might be to analyse changes in the characteristics between subsequent phases of the process model.

Simplistic Rogers generalised only across diffusions in which little or no specialised knowledge about the innovations was required prior to the adoption [Robertson and Gatignon, 1986]. In addition, those innovations' consequences did not affect other members of the social system (at least not directly), and their potential adopters were able to decide independently for or against the innovations, according to Fichman [1992], who designed a model for choosing a framework according to the complexity of the diffusion (see section 3.2.2). In his words, "the opportunities to apply classical diffusion 'as is' may be rare indeed."

Furthermore, Rogers' framework does not include the degree of interconnectedness of communication networks, or the concept of geographical proximity, both of which influence the speed at which information can spread [Haegerstrand, 1967].

The Technology Acceptance Model

The TAM [Davis, 1986, Davis et al., 1989] builds on the Theory of Reasoned Action (TRA) [Fishbein and Ajzen, 1975, Ajzen and Fishbein, 1980], which in turn is a refined version of the Fishbein model [Fishbein, 1967]. The TRA is a model about the relation between attitude and behavioural intention, specifically as applied to "objects" (which could be

immaterial, such as a process): an individual may hold beliefs about a given object and as a result, form an attitude towards the object. An attitude may lead to intentions with respect to the object, and those intentions result in behaviour. Behaviour can be positive and negative, as long as it is explicit: said individual may start using the given object, or may refuse it. This decision may be influenced by social norms, another component of the TRA [Fishbein and Ajzen, 1975, p. 301ff], albeit one that is only minimally understood [*ibid.*, p. 304].

The TAM has been referred to as "the most influential and commonly employed theory for describing an individual's acceptance of information systems" [Lee et al., 2003]. It has emerged as a powerful model [Pijpers, 2001], robust and parsimonious [Venkatesh and Davis, 2000], and has been validated through a replication study, covering non-computer technologies, as well as computer software [Adams et al., 1992]. In addition, many extensions have been proposed: Chau [1996] distinguishes between short-term and long-term usefulness and found that while both have a net-positive effect on an individual's likelihood to accept, the short-term perception was most significant. Furthermore, Chau found that perceived ease-of-use had "no significant, direct relationship" with "behavioral intention to use a technology." Wixom and Todd [2005] integrated user satisfaction measures with the TAM. Bagozzi et al. [1992] focus on the importance of learning in technology acceptance.

Venkatesh and Davis [2000] defined TAM2 with additional theoretical constructs spanning social influence and cognitive instrumental processes. Venkatesh et al. [2003] reviewed eight acceptance models, including the TRA, the TAM, and the TAM2, and consolidated their salient features into the Unified Theory of Acceptance and Use of Technology (UTAUT). The UTAUT proposes four key constructs, which influence an individual's behavioral intention (performance expectancy, effort expectancy, social influence, and facilitating conditions), and four mediators (gender, age, experience, and voluntariness of use), each of which has an influence on the weight of each of the four constructs. The model outperformed each of the eight models on which it was based [Venkatesh et al., 2003]. Garfield [2005] successfully used UTAUT to study the acceptance of ubiquitous computing.

Criticisms The TAM is based only on two features: perceived ease-of-use and perceived usefulness, while the latter is influenced by the former "because, other things being equal, the easier the system is to use the more useful it can be" [Venkatesh and

Davis, 2000]. As such, it may be overly simplistic or broad, and not take many other factors influencing the acceptance decision into account.

Furthermore, Silva [2007b] has denied the falsifiability of the TAM (and the TRA). The TRA has striking similarity to psycho-analysis, as it is "very general, designed to explain virtually any human behavior" [Ajzen and Fishbein, 1980, p. 4]. Since the TAM is modelled on the TRA[14], it exposes the same similarity and is thus pseudo-scientific [Popper, 1972, p. 74ff]. Also, according to [Popper, 1972], the link between beliefs/desires, attitude, and action is logical, which "implies that any causal relation among them cannot be tested empirically and subjected to falsification" [Silva, 2007b].

Wejnert's integrated model of innovation diffusion

With a similar claim to Rogers [1962], that diffusions are mostly analysed in isolation from the insights of the others, Wejnert [2002] designed a conceptual framework by merging diffusion variables identified in other studies into a coherent structure. Her framework has three major components with various constituent variables:

- Characteristics of innovations
  - public versus private consequences
  - benefits *vs.* costs
- Characteristics of innovators
  - societal entity
  - familiarity with the innovation
  - status characteristics
  - socioeconomic characteristics
  - position in social networks
  - personal characteristics
- Environmental context
  - geographical setting
  - societal culture

[14] Silva [2007a] sees "TAM, TAM2 and UTAUT not as a discrete succession but as the progression of the same model."

- political conditions
- global uniformity

A slightly more thorough description of the individual variables may be found in section B.2.

Evaluation Wejnert's framework is rather young, which may be the reason that I have only found instances of the author herself using it, *e.g.* to assess the diffusion of democracy over the past two centuries [Wejnert, 2005]. Nevertheless, it is an interesting candidate, though not without shortcomings.

Other than Rogers' framework (see section 3.2.2), Wejnert does not consider the process of diffusion, but only characteristics of the innovation, the potential adopters, and the environmental context of a diffusion. It is helpful that she distinguishes between characteristics of the individual and the collective, but most of her variables overlap and lack in orthogonality, similar to what I have shown for Rogers' framework earlier (see section 3.2.2). For instance, members with high socioeconomic characteristics are likely to find themselves in central positions within the social network, thus boosting their social status.

Weijnert does not distinguish between interpersonal and one-way mass communication. It seems, however, that Rogers' second element, communication, might be a worthwhile addition to Wejnert's framework. Furthermore, Wejnert does not really distinguish between individual, collective, voluntary, and authoritarian decisions, another import candidate from Rogers' framework.

She regards consequences of an adoption to be a feature of an innovation, rather than the social system in which the diffusion is taking place, as does Rogers. Wejnert makes interconnectedness, network structure, proximity and level of communication explicit. Moreover, she caters for globally-spread groups, which in the context of my study is a very strong point.

Fichman's IT diffusion framework

Fichman [1992] finds that "IT diffusion research can diverge from classical diffusion assumptions due to characteristics of the technology (user interdependencies , knowledge barriers) or the locus of adoption (individual versus organizational). He proposes a

framework which maps classes of technology against locus of adoption to obtain four IT adoption contexts: type 1 technologies come with low knowledge burden and low user interdependencies, while type 2 technologies require high levels of knowledges and affect other users. The locus of adoption is either the individual or an organisation. For each of the four permutations, Fichman points at attributes from other frameworks. For instance, individual type 1 diffusions he leaves to Rogers, while he refers to *e.g.* Zmud [1982, 1984][15] for organisational type 2 diffusions.

Although not a framework by itself, Fichman splits the field of IT innovation diffusions along two prominent axes and encourages the use of *different* frameworks for each permutation. Unfortunately, I could not find a study making use of his work. However, the idea is contagious, and needs to be considered during the design of a framework for the diffusions of methods in the Debian Project, and additional axes (*e.g.* voluntary *vs.* authoritarian decisions) incorporated.

4 Research approach

Chapter 2 introduces the Debian Project and Debian GNU/Linux and illustrates the size and complexity of this project and the operating system it produces. In section 2.3 I outlined the workflow of Debian development and associated approaches, highlighted some of the challenges the project faces, and hypothesised some improved workflows. I followed with a set of traits commonly found among Debian developers, in particular their role as volunteers and related implications on project management, their strive for technical excellent, and their ideological concerns, as manifested by the strict adherence to freedom (see section 2.4).

The goal of my research is to determine the factors that affect diffusions of new approaches to common development tasks among Debian developers, and to structure these factors into a coherent framework, which can be used to predict whether a given development method will be adopted by the developers. I hypothesis that such a framework will enable diffusers to focus their efforts to increase the success of a planned or ongoing diffusion.

4.1 Overview of the research

I propose an exploratory, qualitative study which combines a Straussian grounded theory approach [Strauss, 1987, Strauss and Corbin, 1998] with a Delphi study [Dalkey and Helmer, 1963, Gordon and Helmer-Hirschberg, 1964, Linstone and Turoff, 2002]: first, I identify a set of factors from a variety of data sources, and then refine and realign these factors into a framework.

4.1.1 Straussian grounded theory

A grounded theory is a theory grounded in data: the theory emerges from the data as those data are systematically collected and analysed. The grounded theory research

methodology was proposed by Glaser and Strauss [1967], who criticised the "overemphasis in current sociology on the verification of theory, and a resultant de-emphasis on the prior step of discovering what concepts and hypotheses are relevant for the area that one wishes to research" [*ibid.*, p. 1]. Grounded theory concentrates on identifying (or discovering) the categories or concepts inherent in the data, and to derive a theory from those. The grounded theory approach is exploratory (the researcher considers options), rather than verificatory (the researcher tests and verifies an existing theory).

Glaser and Strauss [1967] provided ideas, but no recipes. Furthermore, their book lacked examples, resulting in confusion among their readers. In the following years, as Glaser and Strauss tried to straighten up the misconceptions, they diverged into two strains of grounded theory, mainly differing in their view on prior knowledge of the field of research. Glaser [1992, p. 70] makes it perfectly clear that scholarly and research knowledge of the substantive field have to be realised *ad hoc* (which was the original pretense), while Strauss and Corbin [1998, p. 42] anticipates bias and sees the development of a grounded theory as an iterative, two-way process: "the researcher is shaped by the data, just as the data are shaped by the researcher."

I chose the Straussian strain of grounded theory over the original grounded theory advocated by Glaser for three reasons:[1]

1. A central component of grounded theory is coding, or identifying categories. Glaser maintains that it is important to let codes emerge from the data, using "coding families" to structure the resulting codes [Glaser, 1978]. Strauss and Corbin, on the other hand, propose the use of a "coding paradigm" in a second pass through the data, after identifying a set of initial codes. The goal of this step is the identification of axes (key concepts, principal components; axes are orthogonal) in the codes [Strauss and Corbin, 1998, p. 229ff]. My goal is a framework of factors, and Strauss' approach seems like a better means to that end, even if Glaser's concern cannot be fully discarded: axial coding forces categories onto the data, rather than allowing them to emerge from the data.
2. Glaser and Strauss [1967, p. 37] suggests that the researcher "ignore the literature of theory and fact on the area under study, in order to assure that the emergence of categories will not be contaminated," although they were aware

[1] A fourth reason would have to be the polemic demarcation of Strauss and Corbin [1990] by Glaser [1992].

that no researcher could "approach reality as a *tabula rasa*" [*ibid.*, p. 3]. Strauss and Corbin [1998, p. 42ff] assume a more liberal stance and note that objectivity in qualitative research is "openness, a willingness to listen" [*ibid.*, p. 42]. They propose a number of strategies to deal with bias rather than to deny it.

3. Strauss and Corbin [1998, p. 52] encourage the use of non-technical literature, such as "letters, biographies, diaries, reports, videotapes, newspapers, catalogs (scientific and otherwise), and a variety of other materials." They furthermore postulate that non-technical literature "can be used to supplement interviews and observations. For example, much can be learned about an organization, its structure, and how it functions [...] by studying its reports, correspondences, and internal memos" [*ibid.*]. Glaser [1992, p. 31], on the other hand, argues that "there is a need not to review any of the literature in the substantive area under study [...] to not contaminate, be constrained by, inhibit, stifle or otherwise impede the researcher's effort to generate categories, their properties, and theoretical codes." Given my level of involvement with the Debian Project, the choice for Strauss' approach is an easy one.

While Glaser very much condemns the possibility of bias influencing the coding procedure, Strauss and Corbin do *not* require bias to exist. In other words, it is entirely possible, according to their theory, to let a code emerge from the data. In fact, it seems that the evolution of codes during the first phase of open coding, as well as during the process of axial coding is anticipated.

Grounded theory has been criticised for being extremely inductive, since it allows a researcher to freely modify the research hypothesis throughout the study. Furthermore, other researchers have attempted to disprove it; those arguments have been largely refuted by Mjøset [2005].

4.1.2 Delphi study

According to its creators, the Delphi method[2] is the way of finding "the most reliable consensus of opinion of a group of experts" [Dalkey and Helmer, 1963]. Surowiecki [2004] argues that the crowds have more wisdom than the individual because, given enough diversity in the members of the crowd, their averaged statements converge closer to the "right answer" than an individual's response. Nonetheless, [Linstone and

[2]The Delphi method was developed at the RAND Corporation in the 1940s.

Turoff, 2002, p. 5] argues that the question of whether the crowds are wiser than the few has not received the attention it deserves.

A Delphi study "may be characterized as a method for structuring a group communication process so that the process is effective in allowing a group of individuals, as a whole, to deal with a complex problem" [Linstone and Turoff, 2002, p. 3]. This process is an instance of moderated communication: a moderator (facilitator) serves a series of questions to the participants, who are also known as experts or panelists, and who return their answers to the moderator. The answers are anonymised and made available to the other participants. Then, the experts can modify their response in the light of the findings, and this continues as an iterative process until the group arrives at an agreed response, though not necessarily a single view. Alternatively, the facilitator may draw up a new set of questions, incorporating the returns from the previous round, instead of expecting the experts to modify their responses.

The Delphi study has a number of strengths: it is a transparent and democratic technique, asynchronous [Turoff and Hiltz, 1996], and is suitable for complex or broad problems for which no analytical techniques exist, but which can benefit from subjective judgements on a collective bases [Yousuf, 2007]. On the other hand, the process has been criticised as unscientific [Sackman, 1974], which was later refuted [Goldschmidt, 1975]. A number of criticisms have been brought forth again the Delphi method, which are still standing; Mullen [2003] provides a very good overview.

Similar to the grounded theory approach, the Delphi study is susceptible to bias. [Linstone and Turoff, 2002, p. 6] identify a number of problems which relate to the role of the moderator, who has to take care not to be suggestive and objectively formulate questions and summarise the participants' responses, making sure that disagreements are not dropped.

Another concern with the Delphi approach is the time and effort that it consumes. The correspondence should be regular and frequent to prevent losing participant's interest or context. At the same time, the facilitator will need to process large amounts of data, which grow with the size of the experts panel. An appropriate communication medium can help to reduce both, the effort and time requirement [Turoff and Hiltz, 1996].

A modified variant of the Delphi method, the wide-band Delphi approach, was developed by Farquhar and Boehm in the 1970s [Boehm, 1981, p. 335]; a wide-band

Delphi study incorporates discussions among experts before each expert submits his/her responses. Stellman and Greene [2005, ch. 3] provide an excellent summary.

4.2 Four phases of research

I partitioned my research into four phases, each of which builds on the result of the previous phase:

1. Collection of factors
2. Community survey to sort and augment the factor set
3. Delphi approach to fitting factors into a framework
4. Application and verification of the framework

4.2.1 Phase 1: Collection of factors

The first phase of my research is also the beginning of the grounded theory approach. The goal of this phase is an exhaustive, but structured set of factors which influence diffusions. I intend to use three sources of data: semi-structured interviews with developers, project forums and mailing lists, and the literature.

The role of the literature in factors collection should be clarified: Bajaj [2000] mentions the importance of "factors that are understandable to [*sic.*] the subjects [who] may not perceive them in the same way as academics." This is a valid concern, which the author addressed by obtaining factors *only* from interviews. He later used the literature while mapping the factors identified in his study to the ones found in the scholarly texts. I take a similar approach, but I also try to obtain subjects' judgements on the relevance of factors they have not mentioned, towards the end of the interviews.

Having been involved in the development of Debian for a long time, I often considered the questions of diffusion and adoption of new methods. Thus, I cannot

deny a significant amount of experience, which I bring into this research. To extract as much information from my subjects, and thus to minimise my bias' influence on the final set of factors, it is of utmost importance that I refrain from suggestive questions, presuppositions, or premature conclusions on the subjects' statements.

I have already conducted seven *ad hoc* interviews with peers. All these interviews emerged out of discussions over my research, thus with random subjects who had an interest in the work. Three of those were live, in the periphery of a conference. The other four I conducted over IRC. These interviews were premature in two ways: first, they were unstructured and I was not prepared enough to gather the subjects' thoughts on some of the factors identified in literature. Second, while aware of the bias problem and taking ample care not to influence my subjects' responses, I had not considered the issue enough to be able to act appropriately; I thus probably have influenced the responses. Nonetheless, these interviews provided valuable insights.

I kept track of subjects' responses on a publicly-accessible wiki.[3] So far, only few people know of this wiki, but my intention is to invite others to contribute by making changes or attaching notes, once I have gathered enough data. My rationale for keeping the wiki private for now is that on a wiki, a contributor is subjected to other peoples' responses and hence may fail to provide original input. However, once a considerable body of data has been collected, I postulate that inviting a group of people to chaotically reorganise and comment information can yield useful results. The wiki is based on a version control system, thus keeping track of all changes; in cases of destructive or improper participation, I can revert commits.

In future interviews, I want to concentrate on a subject's previous decisions and preferences. I want to obtain information about motivations behind previous adoptions, but I would also like to identify a subject's needs and assess, using hypothetical solutions, how likely the subject would adopt each. Instead of asking them about specific factors, or expecting them to label their experiences and expectations, I am looking for qualitative elements from which I can later induce the factors.

[3] `http://phd.martin-krafft.net/wiki/factors/`

4.2.2 Phase 2: Community survey

The goal of the second phase is to get an idea of the community members' perception of the relevance of each of the factors. I would thus like to administer a community survey, but this plan is not finalised. A survey is a serious challenge and often underestimated [Hyman, 1955, p. 2]. Additional considerations have to be made for web-based surveys [Hyman, 1955]. Nevertheless, I have a number of ideas which I would like to carry into practise, thus motivating a survey, if it proves reasonable in the context of this study. One possible approach could be to use the community survey as part of a conjoint analysis approach, which is a common method in mathematical psychology and marketing; Bajaj [2000] has ported the method to IS research.

As one of the largest FLOSS projects, the Debian Project has been subject of a number of surveys. While I have the impression that members of the project generally hold a favourable position towards research of our project, and thus surveys, a number of surveys have failed in the past, leading developers to write up advice to surveyors.[4]

Every survey I have seen thus far is either web- or e-mail-based. Web-based surveys suffer from number of problems, such as the requirement to be connected at the time when the survey is to be taken, rendering problems, frustrations with the user interface, and inability to judge the expected time until completion. In addition, once the survey is submitted, one's responses cannot be amended; none of the survey employed a reliable means of authentication.

Mail-based surveys solve a number of those problems: users can use their e-mail programme and editor of choice to participate in the survey, and the can judge the time it takes to complete the survey from the length of the mail message. Furthermore, electronic mail can be signed and thus authenticated, allowing for the resubmission of responses. Furthermore, an e-mail response can be made offline. However, e-mail messages are free-form text, allowing partakers of the survey to modify responses or answer outside the allowed set of responses, answer only a subset of questions, and attach notes. Additionally, character set and formatting incompatibilities can further complicate the analysis. Finally, spam filters can get in the way.

[4] `http://people.debian.org/~mjr/surveys.html` [6 Nov 2007]

I envision a means to conduct a survey, which is highly tailored for the target group: DDs, who are unafraid of the command line, and who appreciate technically sound implementations which do not restrict their freedom:

- I publish a console-based, interactive script, which can be trivially run on every Linux system, under a free licence. I also provide a Debian package to make installation even easier. It remains to be seen whether this can be done in a general way so that the package may be provided in the official Debian archive. A central instance of the script will be provided to let users partake, who do not or cannot install extra software.
- Each question asks the user for a response according to its description: multiple-choice responses, Likert scale rankings, and free-form text.
- The script allows users to freely navigate between questions and to amend responses. Interaction with the script should be intuitive and require as few key strokes as possible. Furthermore, the console should be used in such a way to put each question into context, at least with respect to the previous question.
- Comments may be attached to each question, and amended at any time.
- When done, the user instructs the script to submit the responses.[5] This can be done an arbitrary number of times. It is conceivable that users be asked for comment whenever they revisit questions and change their minds.
- The script uses a version control system as backend storage, and thus keeps track of changes across sessions and can merge changes to a central data-store.
- Commits are signed with the participant's cryptographic key,[6] and thus authenticated.

With an interactive script, it is also conceivable that questions are asked depending on previous responses. It is even plausible that questions are created according to the outcome of previous questions to get further clarification. My goal is for users to attribute relevance to each factor. Instead of a ranking on a Likert scale, it may be more fruitful to let them rank factors relative to each other. For such a use case, dynamic creation of questions could be highly appropriate.

[5] Submission may have to be automated to guard users from forgetting to submit them. However, this will not work in offline mode.

[6] As the set of participants is limited to developers of Debian, where cryptographic keys are integral components of the workflow (see section 2.3), it can be assumed that each participant is in possession of a cryptographic key.

As mentioned before, participation in the survey will be non-anonymous. Partakers will be asked to identify themselves with the same e-mail address they use when contributing to Debian by way of a cryptographic signature. This should encourage constructive participation, allow for followup questions, and put responses into the context of the partaker's contributions to Debian. The results from the survey will be made available under a free licence.

As part of this survey, I also intend to probe participants about their individual adoption behaviour and their perceptions of specific diffusions. The resulting data are used during the fourth phase of the research (see section 4.2.4).

4.2.3 Phase 3: Delphi study

With a the set of factors from phase 1, as well as the input from the community gathered in phase 2, my next step is to embark on a Delphi study. Given an initial structuring of the factors, the output of this study is a refined framework, which can be used to assess diffusions of methods among DDs.

I am looking at assembling around 30 experts into the panel, which seems like a good compromise between diversity and manageability; I have been part of discussion with more than 30 people and would have a hard time arguing that each additional participant significantly increases the group's diversity, or the amount of original information. Cantrill et al. [1996, cited in Mullen [2003]] reported on studies with panel sizes varying from 4 to 3000 and recommend that size "should be governed by the purpose of the investigation." Holsapple and Joshi [2002], who used an expert panel to design ontologies (which is closely related to framework design), also consulted around 30 experts.

Core to the Delphi approach is the choice of communication medium. Given the study at hand, with experts spread all over the globe, an electronic medium is a necessity. One may find a plethora of supporting methods in the domain of Computer Supported Cooperative Work (CSCW) [Greif, 1988, Schwabe et al., 2001]. The simplest choice would be e-mail, using a star topology, in which the facilitator communicates with each of the experts individually. This approach allows for asynchronicity, which Turoff and Hiltz [1996] identify as the most important feature of the communication medium in a Delphi study. In addition, the degree of anonymity is flexible — Turoff and Hiltz [1996]

call anonymity a property "that should [not] be considered a hard and fast rule for all aspects of a Delphi [study]."

E-mail provides the facilitator with the greatest level of control and overview, since s/he moderates every response, and sends a summary to the other expert. However, e-mail also imposes the largest workload on the person chairing the study, who has to continuously organise and integrate the feedback s/he receives. Furthermore, during integration and in summarising, subtle influences of bias are easily overlooked.

The desired output of the Delphi study is structured information. It is thus worth to consider another means of collaboration. A number of Delphi-specific applications exist,[7] but these are either web-based or only available as proprietary software, and it might be inappropriate to ask FLOSS project participants to use those (*cf.* section 2.4). In addition, such tools strictly impose the structure of the Delphi study and thus may be limiting [Turoff and Hiltz, 1996].

If I can ensure that all participants understand the purpose and desired outcome of the study, then a loosely-coupled set of tools may be beneficial. In particular, I am thinking of the combination of a wiki, an anonymising mailing list, and an IRC chat channel, tools with which all candidates for the expert panel are highly familiar and accustomed. The interactive survey script from phase 2 (see 44) may also fit in.

I have already mentioned this Delphi study to a couple of colleagues, most of whom were interested in serving on such a panel. In addition to DDs and contributors, I would like to include developers from other distributions (such as Fedora and Gentoo) and other complex FLOSS projects (such as GNOME and KDE), as well as social science researchers familiar with the Debian Project. An issue to be considered is the compensation of the experts, who are best treated as consultants and honoured for their time [Linstone and Turoff, 2002, p. 6]. Possible means of compensation include a draw for a technological device, or monetary rewards.

[7] *E.g.* the Delphi Decision Aid is a free service which implements the Delphi method: `http://armstrong.wharton.upenn.edu/delphi2/` [7 Nov 2007]

4.2.4 Phase 4: Application and verification

In the final phase of my research, I seek to test the framework developed thus far. Even though a prescriptive application would be ideal, I leave that for further research as it is outside the scope of my current research. Instead, I intend to descriptively investigate the diffusion of two classes of methods in the Debian Project: package build helpers (`debhelper`, `cdbs`, and `yada`), and patch management and version control systems for package maintenance (`dpatch`, `quilt`, `SVN`, and `Git`).

Both classes render themselves nicely for this type of research, as data pertaining to the evolution of their usage frequency exist for most of the history of the Debian Project. A (source) package using a build helper or patch management system must state so in its meta data, and these are archived for every release since Debian 2.0 ("hamm"), almost a decade ago (see section A.3). Since spring 2005, the Debian archive is snapshotted daily,[8] allowing for granular analysis.

The repositories of most Debian packages maintained with a version control system are published on a single server maintained by the Debian Project, to which I have access, and whose administrators I know very well. I should thus have no problem to determine the usage data of the various version control systems hosted on that machine.

The testing procedure is as follows: I structure the results from the survey of phase 2 (see section 4.2.2) into an instance of the framework from phase 3 (see section 4.2.3), using the framework to average responses. If the framework is descriptive, the result should be indicative of the evolution of use of each of the specific methods under investigation.

[8] `http://snapshot.debian.net` [7 Nov 2007]

5 Final remarks

5.1 Research to date

The university's Code of Practice [*sic.*] for Transfer from the Masters to PhD Register requests the candidate to "Report on research work carried out to date by the candidate, which demonstrates: (i) feasibility of the proposed research work; (ii) the ability of the candidate to carry out the proposed research work."

I have not produced significant academic research output up to this point, but I have been working on the topic even before I enrolled for the university programme, starting with the writing of my book. In writing, I got very deeply immersed in the Debian Project and gained insight into many of the processes far beneath the surface. Throughout the process, I did not merely report on the *status quo* of the project and its operating system, but I implemented fixes instead of documenting workarounds, which provided even more in-depth experience.

I started to become interested in Debian workflow issues while assisting a lecture on CSCW at the University of Zurich in 2003. However, it was not until after I had finished my book that I saw a way to put two and two together: following the release of my book in the summer of 2005, I thus declared my intent to pursue doctoral research on the area and discontinued my assistantship in Zurich.

My research-relevant output so far has been almost exclusively within the project, in mailing list threads, discussions at conferences and on IRC, web log posts, and presentations. The following is a selection of the most important items:

- I gave an online tutorial about using distributed VCS for Debian packaging: `http://blog.madduck.net/debian/2005.08.11-arch-packaging-over`

- I conceptualised a new VCS-based upload process for Debian packages, which was later adopted and modified by the Ubuntu project:[1] `http://blog.madduck.net/debian/2005.08.11-rcs-uploads`
- I modelled the workflow of our testing security team in a (failed) attempt to streamline it: `http://lists.alioth.debian.org/pipermail/secure-testing-team/2006-February/000689.html`
- I gave a presentation about improved Debian workflows at FOSDEM 2006: `http://phd.martin-krafft.net/talks/phd-overview_fosdem_2006.02.26/slides.pdf`
- I gave a presentation (in German) on the Debian package life-cycle and associated workflow at the Chemnitzer Linux-Tage 2006: `http://chemnitzer.linux-tage.de/2006/vortraege/detail.html?idx=252` [6 Nov 2007]
- I gave a presentation on improving cooperation in volunteer projects at FrOSCon 2006: `http://phd.martin-krafft.net/talks/improving-cooperation_froscon_2006.06.24/slides.s5.html`
- I presented my research objective at Debconf7: `http://phd.martin-krafft.net/talks/phd-overview_debconf7_2007.06.21/slides.s5.html`
- I gave a joint presentation at Debconf7 about using distributed VCSs for Debian packaging: `https://penta.debconf.org/~joerg/events/53.en.html` [6 Nov 2007]
- I created the `vcs-pkg` mailing list as a discussion forum for using VCS for distribution packaging, and invited people from other distributions to join: `http://lists.madduck.net/listinfo/vcs-pkg`
- I later worked out a complete workflow for Debian packaging with Git: `http://blog.madduck.net/debian/2007.10.03_packaging-with-git`

In addition, I've been part of numerous discussions on efficiency and workflow issues in the Debian Project, most of which took place on IRC.

All these contributions have put me into the position of a lead user [*cf.* von Hippel, 1986] within the project. In addition, I am a well-known and respected member of the community, which is a great foundation for my research.

[1] Without the appropriate acknowledgement: `https://wiki.ubuntu.com/NoMoreSourcePackages` [8 Nov 2007]

5.2 Work plan

The following table displays the time frame of my research:

Date	Milestone
Sep 2007	Commence collection of factors
Dec–Apr 2008	Semi-structured interviews
Jan 2008	Present research at Linux Conf Australia
Jan 2008	Publish wiki with factors
Feb 2008	Present research at FOSDEM
Feb–Apr 2008	Conduct community survey
May–Sep 2008	Conduct Delphi study
Aug 2008	Present research at Debconf8
Aug 2008	Conduct more interviews at Debconf8
Oct–Nov 2008	Apply and verify framework
Dec–Feb 2009	Finish writeup

Bibliography

Dennis A. Adams, R. Ryan Nelson, and Peter A. Todd. Perceived usefulness, ease of use, and usage of information technology: a replication. *MIS Quarterly*, 16(2):227–47, June 1992.

Icek Ajzen and Martin Fishbein. *Understanding Attitudes and Predicting Social Behavior*. Prentice-Hall, Eaglewood Cliffs, NJ, USA, 1980.

Teresa M. Amabile, Regina Conti, Heather Coon, Jeffrey Lazenby, and Michael Herron. Assessing the work environment for creativity. *Academy of Management Journal*, 39 (5):1154–84, October 1996.

Juan-José Amor-Iglesias, Jesús M. González-Barahona, Gregorio Robles-Martínez, and Israel Herráiz-Tabernero. Measuring libre software using Debian 3.1 (sarge) as a case study: Preliminary results. *UPGRADE: the European Journal for the Informatics Professional*, 6(3):13–6, June 2005a.

Juan-José Amor-Iglesias, Gregorio Robles-Martínez, Jesús M. González-Barahona, and Israel Herráiz-Tabernero. From pigs to stripes: A travel through Debian. In *Proceedings of the 6th Debian Conference*, Helsinki, Finland, June 2005b.

Richard P. Bagozzi, Fred D. Davis, and Paul R. Warshaw. Development and test of a theory of technological learning and usage. *Human Relations*, 45(7):659–86, July 1992.

Akhilesh Bajaj. A study of senior information systems managers decision models in adopting new computing architectures. *Journal of the AIS*, 1(1es), March 2000.

George M. Beal, Everett M. Rogers, and Joe M. Bohlen. Validity of the concept of stages in the adoption process. *Rural Sociology*, 22:166–8, 1957.

Jesús Bermejo and Naci Dai. Open source strengths for defining software product line practices. Presented at the First International Workshop on Open Source Software and Product Lines, 10th International Software Product Line Conference (SPLC 2006), August 2006. URL `http://www.sei.cmu.edu/splc2006/OSS_paper.pdf`.

Barry W. Boehm. *Software Engineering Economics*. Advances in Computing Science & Technology. Prentice-Hall, October 1981.

J. A. Cantrill, B. Sibbald, and S. Buetow. The Delphi and nominal group techniques in health services research. *International Journal of Pharmacy Practic*e, 4(2):67–74, 1996.

Patrick Y. K. Chau and Kar Yan Tam. Factors affecting the adoption of open systems: an exploratory study. *MIS Quarterly*, 21(1):1–24, Mar 1997. URL `http://links.jstor.org/sici?sici=0276-7783%28199703%2921%3A1%3C1%3AFATAOO%3E2.0.CO%3B2-0`.

Patrick Y.K. Chau. An empirical assessment of a modified technology acceptance model. *Journal of Management Information Systems*, 13(2):185–204, Fall 1996.

Wesley M. Cohen and Daniel A. Levinthal. Absorptive capacity. *Administrative Science Quarterly*, 35(1):128–52, March 1990. URL `http://links.jstor.org/sici?sici=0001-8392%28199003%2935%3A1%3C128%3AACANPO%3E2.0.CO%3B2-5`.

E. Gabriella Coleman. *The Social Construction of Freedom in Free and Open Source Software: Hackers, Ethics, and the Liberal Tradition*. PhD thesis, University of Chicago, Chicago, IL, USA, August 2005a.

E. Gabriella Coleman. Three ethical moments in Debian: the making of an (ethical) hacker, part III. In *The Social Construction of Freedom in Free and Open Source Software: Hackers, Ethics, and the Liberal Tradition* Coleman [2005a], chapter 6.

Kevin Crowston and James Howison. The social structure of open source software development teams. In *Proceedings of the OASIS 2003 Workshop (IFIP 8.2 WG)*, 2003.

Kevin Crowston, James Howison, and Hala Annabi. Information systems success in free and open source software development: Theory and measures. *Software Process: Improvement and Practic*e, 11(2):123–48, March 2006. doi: 10.1002/spip.259.

Norman Dalkey and Olaf Helmer. An experimental application of the Delphi method to the use of experts. *Management Science*, 9(3):458–67, April 1963. URL `http://links.jstor.org/sici?sici=0025-1909%28196304%299%3A3%3C458%3AAEAOTD%3E2.0.CO%3B2-K`.

Frank E. X. Dance and Carl E. Larson. *The Functions of Human Communication: a Theoretical Approach*. Holt, Rinehart & Winston, New York, NY, USA, 1976.

Fred D Davis. *A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results*. PhD thesis, Massachusetts Institute of Technology, Sloan School of Management, 1986.

Fred D. Davis, Richard P. Bagozzi, and Paul R. Warshaw. User acceptance of computer technology: a comparison of two theoretical models. *Management Science*, 35(8): 982–1003, August 1989.

Debian Project. The Debian Free Software Guidelines, April 2004a. URL `http://www.debian.org/social_contract#guidelines`. Last accessed: 31 July 2006.

Debian Project. Constitution for the Debian Project. online, September 2006. URL `http://www.debian.org/devel/constitution`. Last accessed: 4 October 2007.

Debian Project. The Debian New Maintainer process. online, August 2007a. URL `http://www.debian.org/devel/join/newmaint`. Last accessed: 8 October 2007.

Debian Project. *The Debian Policy Manual*, June 2005. URL `http://www.debian.org/doc/debian-policy`. Last accessed: 31 July 2006.

Debian Project. Debian Social Contract, April 2004b. URL `http://www.debian.org/social_contract`. Last accessed: 31 July 2006.

Debian Project. *How to Report a Bug in Debian*, October 2007b. URL `http://www.debian.org/Bugs/Reporting`. Last accessed: 10 October 2007.

Paul J. Deutschmann and Orlando Fals Borda. La communicacíon de las ideas entre los campesinos Columbianos. Technical report, Universidad Nacional de Colombia, 1962.

Jan Ernst Fagerberg. Innovation: a guide to the literature. In Jan Ernst Fagerberg, David C. Mowery, and Richard R. Nelson, editors, *Oxford Handbook of Innovation*, chapter 1, pages 1–26. Oxford University Press, Oxford, UK, 2004.

Joseph Farrell and Garth Saloner. Standardization, compatibility, and innovation. *The RAND Journal of Economics*, 16(1):70–83, Spring 1985.

Joseph Farrell and Garth Saloner. Installed base and compatibility: Innovation, product preannouncements, and predation. *The American Economic Review*, 76(5):940–55, December 1986.

Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison-Wesley, London, England, 2002. URL `http://opensource.ucc.ie/uossd`.

Robert G. Fichman. Information technology diffusion: a review of empirical research. In DeGross, Becker, and Elam, editors, *Proceedings of the Thirteenth International Conference on Information Systems (ICIS)*, pages 195–206, Dallas, TX, USA, December 1992.

Robert G. Fichman. Going beyond the dominant paradigm for information technology innovation research: Emerging concepts and methods. *Journal of the Association for Information Systems*, 5(8):314–55, August 2004.

Martin Fishbein. Attitude and the prediction of behavior. In Martin Fishbein, editor, *Readings in Attitude Theory and Measurement*, pages 477–92. Wiley, 1967.

Martin Fishbein and Icek Ajzen. *Belief, Attitude and Behavior: an Introduction to Theory and Research*. Addison-Wesley, 1975. URL `http://www.people.umass.edu/aizen/f&a1975.html`.

Miguel A. Fortuna, , and Carlos J. Melián. Do scale-free regulatory networks allow more expression than random ones? *Journal of Theoretical Biology*, 247(2):331–6, July 2007. doi: 10.1016/j.jtbi.2007.03.017.

Ruud T. Frambach and Niels Schillewaert. Organizational innovation adoption: a multi-level framework of determinants and opportunities for future research. *Journal of Business Research*, 55(2):163–76, February 2002. doi: 10.1016/S0148-2963(00)00152-1.

Monica J. Garfield. Acceptance of ubiquitous computing. *Information Systems Management*, 22(4):24–31, 2005.

Giampaolo Garzarelli and Roberto Galoppini. Capability coordination in modular organization: Voluntary FS/OSS production and the case of Debian GNU/Linux. Working paper, University of the Witwatersrand - School of Economics and Business Sciences, November 2003.

Barney G. Glaser. *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. Sociology Press, 1978.

Barney G. Glaser. *Basics of Grounded Theory Analysis: Emergence vs. Forcing*. Sociology Press, 1992.

Barney G. Glaser and Anselm L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing Co., Chicago, IL, USA, 1967.

Peter F. Goldschmidt. Scientific inquiry or political critique? Remarks on Delphi assessment, expert opinion, forecasting, and group process by H. Sackman. *Technological Forecasting and Social Change*, 7:195–213, 1975.

Jesús M. González-Barahona and Gregorio Robles-Martínez. Unmounting the "code god" assumption. Technical report, GSyC, Universidad Rey Juan Carlos, 2003.

Jesús M. González-Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González, and Vicente Matellán Olivera. Counting potatoes: the size of Debian 2.2. *UPGRADE: the European Journal for the Informatics Professional*, 2(6): 60–6, December 2001.

Theodore J. Gordon and Olaf Helmer-Hirschberg. Report on a long-range forecasting study. Technical Report P-2982, RAND Corperation, 1964. URL `http://www.rand.org/pubs/papers/P2982/`.

Irene Greif. *Computer Support for Cooperative Work*. Morgan Kaufmann Publishers Inc., San Mateo, CA, USA, 1988.

Torsten Haegerstrand. *Innovation Diffusion as a Spatial Process*. University of Chicago Press, Chicago, IL, USA, 1967. translated from Swedish (1953) by Allan Pred.

Tim Halloran and William L. Scherlis. High quality and open source software practices. In *Proceedings of the 24th International Conference on Software Engineering*, May 2002. URL `http://www.fluid.cs.cmu.edu:8080/Fluid/fluid-publications/HalloranScherlis.pdf`.

Clyde W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Communications of the ACM*, 45(2):42–7, February 2002. doi: 10.1145/503124.503147.

Thomas P. Hughes. The evolution of large technological systems. In Wiebe E. Bijker, Thomas P. Hughes, and Trevor J. Pinch, editors, *The Social Construction of Technological Systems*. MIT Press, Cambridge, MA, USA, 1989.

Herbert H. Hyman. *Survey Design and Analysis*. Free Press, January 1955.

Chris Jensen and Walt Scacchi. Modeling recruitment and role migration processes in OSSD projects. In *In Proceedings of 6th International Workshop on Software Process Simulation and Modeling*, May 2005.

Michael L. Katz and Carl Shapiro. Network externalities, competition, and compatibility. *The American Economic Review*, 75(3):424–40, June 1985.

Michael L. Katz and Carl Shapiro. Technology adoption in the presence of network externalities. *Journal of Political Economy*, 94(4):822–41, August 1986.

Kelvin P. Kearns. Innovations in local governments: a sociocognitive network approach. *Knowledge and Policy*, 5(2):45–67, 1992.

Steven K. Kline and Nathan Rosenberg. An overview of innovation. In Ralph Landau and Nathan Rosenberg, editors, *The Positive Sum Strategy: Harnessing Technology for Economic Growth*, pages 275–304. National Academy Press, Washington, DC, USA, 1986. URL `http://books.nap.edu/openbook.php?record_id=612&page=275`.

Tero Kojo, Tomi Männistö, and Timo Soininen. *Towards Intelligent Support for Managing Evolution of Configurable Software Product Families*, volume 2649/2003 of *Lecture Notes in Computer Science*, chapter 7, pages 86–101. Springer Verlag, Berlin, Germany, August 2003.

Martin F. Krafft. *The Debian System — Concepts and Techniques*. Open Source Press, Munich, Germany, June 2005. URL `http://debiansystem.info`.

Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, IL, USA, 2nd edition, 1970.

Tae H. Kwon and Robert W. Zmud. Unifying the fragmented models of information systems implementation. In R. J. Boland and R. A. Hirschheim, editors, *Critical Issues in Information Systems Research*, chapter 10, pages 227–51. John Wiley & Sons Ltd., 1987.

Christoph Lameter. Debian GNU/Linux: The past, the present and the future. online, October 2002. URL `http://lameter.com/tokyo/`. Presented at the Free Software Symposium 2002 on October 22, 2002 15:30 at the Japan Education Center.

William Lazonick. The innovative firm. In Jan Ernst Fagerberg, David C. Mowery, and Richard R. Nelson, editors, *Oxford Handbook of Innovation*, chapter 2, pages 29–55. Oxford University Press, Oxford, UK, 2004.

Younghwa Lee, Kenneth A. Kozar, and Kai R.T. Larsen. The Technology Acceptance Model: Past, present, and future. *Communications of JAIS*, 12, December 2003.

Harvey Leibenstein. Bandwagon, snob, and veblen effects in the theory of consumers' demand. *The Quarterly Journal of Economics*, 64(2):183–207, May 1950.

Steven Levy. *Hackers: Heroes of the Computer Revolution*. Penguin Books, New York, NY, USA, 1984.

Stan J. Liebowitz and Stephen E. Margolis. Network externality: an uncommon tragedy. *The Journal of Economic Perspectives*, 8(2):133–50, Spring 1994.

Harold A. Linstone and Murray Turoff, editors. *The Delphi Method: Techniques and Applications*. Information Systems Department, New Jersey Institute of Technology, 2002. URL `http://www.is.njit.edu/pubs/delphibook/`.

Vijay Mahajan, Eitan Muller, and Frank M. Bass. New product diffusion models in marketing: a review and directions for research. *Journal of Marketing*, 54(1):1–26, January 1990.

Vijay Mahajan, Eitan Muller, and Yoram Wind. *New-Product Diffusion Models*. Springer, 2000.

M. Lynne Markus. Toward a "critical mass" theory of interactive media universal access, interdependence and diffusion. In Janet Fulk and Charles W. Steinfield, editors, *Organizations and Communication Technology*, volume 14, pages 491–511. Sage Publications, Newbury Park, CA, USA, 1990.

Martin Michlmayr. Managing volunteer activity in free software projects. In *Proceedings of the 2004 USENIX Annual Technical Conference, FREENIX Track*, pages 93–102, Boston, MA, USA, 2004.

Martin Michlmayr. Quality improvement in volunteer free software projects: Exploring the impact of release management. In Marco Scotto and Giancarlo Succi, editors, *Proceedings of the First International Conference on Open Source Systems*, pages 309–10, Genova, Italy, July 2005a.

Martin Michlmayr. Software process maturity and the success of free software projects. In Krzysztof Zieliński and Tomasz Szmuc, editors, *Proceedings of the 7th Conference on Software Engineering, KKIO 2005*, chapter 1, pages 3–14. IOS Press, Krakow, Poland, September 2005b. accepted.

Martin Michlmayr. *Quality Improvement in Volunteer Free and Open Source Software Projects: Exploring the Impact of Release Management.* PhD thesis, University of Cambridge, Cambridge, UK, March 2007.

Martin Michlmayr and Benjamin Mako Hill. Quality and the reliance on individuals in free software projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 105–9, Portland, OR, USA, 2003.

Martin Michlmayr and Anthony Senyard. A statistical analysis of defects in Debian and strategies for improving quality in free software projects. In Jürgen Bitzer and Philipp J. H. Schröder, editors, *The Economics of Open Source Software Development*, pages 131–48. Elsevier Science Publications B.V., 2006. doi: 10.1016/B978-044452769-1/50006-8.

Martin Michlmayr, Francis Hunt, and David Probert. Quality practice and problems in free software projects. In Marco Scotto and Giancarlo Succi, editors, *Proceedings of the First International Conference on Open Source Systems*, pages 24–28, Genova, IT, July 2005.

Martin Michlmayr, Francis Hunt, and David Probert. Release management in free software projects: Practices and problems. In Joseph Feller, Brian Fitzgerald, Wald Scacchi, and Alberto Silitti, editors, *Open Source Development, Adoption and Innovation: IFIP Working Group 2.13 on Open Source Software*, pages 295–300. Springer Verlag, New York, NY, USA, 2007a.

Martin Michlmayr, Gregorio Robles, and Jesus M. Gonzalez-Barahona. Volunteers in large libre software projects: a quantitative analysis over time. In Robles et al. [2007], chapter 1, pages 1–24. doi: 10.1007/978-0-387-72486-7.

Lars Mjøset. Can grounded theory solve the problems of its critics? *Sosiologisk*, 13: 379–408, 2005.

Lawrence B. Mohr. *Explaining Organizational Behavior*. Jossey-Bass, San Francisco, CA, USA, 1982.

Geoffrey A. Moore. *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers*. Harper Business Essentials, 1991.

Penelope M. Mullen. Delphi: Myths and reality. *Journal of Health Organisation and Management*, 17(1):37–52, 2003. doi: 10.1108/14777260310469319.

Siobhan O'Mahony and Fabrizio Ferraro. Managing the boundary of an 'open' project. Working Paper 03-60, Harvard University, March 2004. URL `http://ssrn.com/abstract=474782`.

Siobhan O'Mahony and Fabrizio Ferraro. The emergence of governance in an open source community. *The Academy of Management Journal*, 50(5), October 2007.

Urban B. Ozanne and Gilbert A. Churchill. Five dimensions of the industrial adoption process. *Journal of Marketing Research*, 8:322–8, 3 1971. URL `http://links.jstor.org/sici?sici=0022-2437%28197108%298%3A3%3C322%3AFDOTIA%3E2.0.CO%3B2-4`.

Jon L. Pierce and Andre L. Delbecq. Organization structure, individual attributes and innovation. *Academy of Management Review*, 2(1):27–37, January 1977. URL `http://links.jstor.org/sici?sici=0363-7425%28197701%292%3A1%3C27%3AOSIAAI%3E2.0.CO%3B2-Y`.

A. Guus M. Pijpers. Senior executives' use of information technology. *Information and Software Technology*, 43:959–71, 2001.

Karl Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge and Kegan Paul, London, England, 1972.

Sridhar A. Raghavan and Donald R. Chand. Diffusing software-engineering methods. *IEEE Software*, 6(4):81–90, 1989. ISSN 0740-7459. doi: 10.1109/52.31655.

S. A. Rahim. *The Diffusion and Adoption of Agricultural Practices: a Study in a Village in East Pakistan*. Pakistan Academy for Village Development, Comilla, East Pakistan, 1961.

Eric S. Raymond. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, October 1999.

Thomas S. Robertson and Hubert Gatignon. Competitive effects on technology diffusion. *Journal of Marketing*, 50(3):1–12, July 1986. URL `http://links.jstor.org/sici?sici=0022-2429%28198607%2950%3A3%3C1%3ACEOTD%3E2.0.CO%3B2-E`.

Gregorio Robles and Jesus M. Gonzalez-Barahona. Contributor turnover in libre software projects. In Ernesto Damiani, Brian Fitzgerald, Walt Scacchi, and Marco Scotto, editors, *Proceedings of the 2nd International Conference on Open Source Systems*, pages 273–286, Como, Italy, June 2006.

Gregorio Robles, Jesús M. González-Barahona, and Israel Herraiz. Challenges in software evolution: the libre software perspective. Paper presented at the Workshop "Challenges in Software Evolution", April 2005a. URL `http://herraiz.org/papers/paper_berna.pdf`.

Gregorio Robles, Jesús M. González-Barahona, and Martin Michlmayr. Evolution of volunteer participation in libre software projects: Evidence from Debian. In Marco Scotto and Giancarlo Succi, editors, *Proceedings of the First International Conference on Open Source Systems*, pages 100–7, Genova, Italy, July 2005b.

Gregorio Robles, Juan Julian Merelo, and Jesus M. Gonzalez-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling*, St. Louis, MO, USA, May 2005c. co-hosted at the ICSE2005.

Gregorio Robles, Jesús M. González Barahona, Martin Michlmayr, and Juan José Amor. Mining large software compilations over time: Another perspective of software evo-

lution. In *International Workshop on Mining Software Repositories*, Shanghai, China, May 2006a.

Gregorio Robles, Luis López, Jesús M. González-Barahona, and Israel Herraiz. Applying social network analysis techniques to community-driven libre software projects. *International Journal of Information Technology and Web Engineering*, 1, July-September 2006b.

Gregorio Robles, Santiago Dueñas, and Jesus M. Gonzalez-Barahona. Corporate involvement of libre software: Study of presence in Debian code over time. In *Open Source Development, Adoption and Innovation*, volume 234/2007 of *IFIP International Federation for Information Processing*, pages 121–32. Springer Verlag, Boston, MA, USA, August 2007. doi: 10.1007/978-0-387-72486-7.

Everett M. Rogers. *Diffusion of Innovations*. Free Press, London, UK, 1st edition, 1962.

Everett M. Rogers. New product adoption and diffusion. *The Journal of Consumer Research*, 2(4):290–301, March 1976.

Everett M. Rogers. *Diffusion of Innovations*. Free Press, London, UK, 5th edition, 2003.

Jeffrey Rohlfs. A theory of interdependent demand for a communications service. *The Bell Journal of Economics and Management Science*, 5(1):16–37, Spring 1974.

Harold Sackman. Delphi assessment: Expert opinion, forecasting, and group process. Technical Report R-1283-PR, RAND Corporation, Santa Monica, CA, USA, 1974.

Joseph Schumpeter. *The Theory of Economic Development: an Inquiry into Profits, Capital, Credit, Interest and the Business Cycle*. Harvard University Press, 1949 edition, 1934.

Joseph Schumpeter. *Business Cycles: a Theoretical, Historical, and Statistical Analysis of the Capitalist Process*, volume 2. McGraw-Hill, 1939.

Joseph Schumpeter. *Capitalism, Socialism and Democracy*. George Allen & Unwin, 1942.

Gerhard Schwabe, Norbert A. Streitz, and Rainer Unland. *CSCW-Kompendium*. Springer, Berlin, Germany, 2001.

Karsten M. Self, Peter Whysall, and Wade Bowmer. Why Debian rocks. online, July 2002. URL `http://twiki.iwethey.org/Main/WhyDebianRocks`. Last accessed: 11 October 2007 [OFFLINE].

Anthony Senyard and Martin Michlmayr. How to have a successful free software project. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04)*. IEEE Computer Society, 2004.

Leiser Silva. E-mail correspondence, 2 Nov. Message ID: 002d01c81d6b$7d882640$6500a8c0 @Bauer.uh.edu, November 2007a.

Leiser Silva. Post-positivist review of technology acceptance model. *Journal of the Association for Information Systems*, 8(4):256–266, April 2007b.

Sulayman Sowe, Ioannis Stamelos, and Lefteris Angelis. Identifying knowledge brokers that yield software engineering knowledge in OSS projects. *Information and Software Technology*, 48(11):1025–33, November 2006.

Sulayman K. Sowe, Ioannis Stamelos, and Lefteris Angelis. Understanding knowledge sharing activities in free/open source software projects: an empirical study. *Journal of Systems and Software*, in press, 2007. doi: doi:10.1016/j.jss.2007.03.086.

Andrew Stellman and Jennifer Greene. *Applied Software Project Management*. O'Reilly Media, 2005. URL `http://www.stellman-greene.com/aspm/content/view/13/1/`.

Troy J. Strader, Sridhar N. Ramaswami, and Philip A. Houle. Perceived network externalities and communication technology acceptance. *European Journal of Information Systems*, 16:54–65, 2007. doi: 10.1057/palgrave.ejis.3000657.

Anselm L. Strauss. *Qualitative Analysis for Social Scientists*. Cambridge University Press, 1987.

Anselm L. Strauss and Juliet Corbin. *Basics of Qualitative Research*. Sage Publications, 1990.

Anselm L. Strauss and Juliet Corbin. *Basics of Qualitative Research*. Sage Publications, 2nd edition, 1998.

Matthias Stuermer. Open source community building. Master's thesis, University of Berne, Berne, Switzerland, March 2005.

James Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Random House, New York, NY, USA, May 2004.

E. Burton Swanson. Information systems innovation among organizations. *Management Science*, 40(9):1069–92, 1994. URL `http://links.jstor.org/`

sici?sici=0025-1909%28199409%2940%3A9%3C1069%3AISIAO%3E2.0.CO%3B2-A.

Gabriel Tarde. *The Laws of Imitation*. University of Chicago Press, Chicago, IL, USA, 1903.

Gabriel Tarde. *The Laws of Imitation*. University of Chicago Press, Chicago, IL, USA, 1969. translated from French by Elsie C. Parsons.

Louis G. Tornatzky and Katherine J. Klein. Innovation characteristics and innovation adoption-implementation: A meta-analysis of findings. *IEEE Transactions on Engineering Management*, 29:28–45, 1982.

Sherry Turkle. *The Second Self: Computers and the Human Spirit*. Simon and Schuster, New York, NY, USA, 1984.

Murray Turoff and Starr Roxanne Hiltz. Computer-based Delphi processes. In *Gazing Into the Oracle: The Delphi Method*. Kingsley Publishers, 1996. URL http://web.njit.edu/~turoff/Papers/delphi3.html.

Christophe van den Bulte. New product diffusion acceleration: Measurement and analysis. *Marketing Science*, 19(4):366–380, Autumn 2000.

Viswanath Venkatesh and Fred D. Davis. A theoretical extension of the Technology Acceptance Model: four longitudinal field studies. *Management Science*, 46(2):186–204, February 2000.

Viswanath Venkatesh, Michael G. Morris, Gordon B. Davis, and Fred D. Davis. User acceptance of information technology: Toward a unified view. *MIS Quarterly*, 27(3): 425–78, 2003.

Eric von Hippel. Lead users: A source of novel product concepts. *Management Science*, 32(7):791–805, July 1986.

Hanna Wallach, Moray Allan, and Dafydd Harries. The Debian New Maintainer process: History and aims. In *Proceedings of the 6th Debian Conference*, Helsinki, Finland, July 2005. Debian.

Steven Weber. *The Success of Open Source*. Harvard University Press, Cambridge, MA, USA, April 2004.

Barbara Wejnert. Integrating models of diffusion of innovations: a conceptual framework. *Annual Review of Sociology*, 28:297–326, 2002. doi: 10.1146/annurev.soc.28.110601.141051.

Barbara Wejnert. Diffusion, development, and democracy, 1800–1999. *American Sociological Review*, 70(1):53–81, February 2005.

Barbara H. Wixom and Peter A. Todd. A theoretical integration of user satisfaction and technology acceptance. *Information Systems Research*, 16(1):85–102, March 2005.

Yuwen Ye, Kumiyo Nakakoji, Yashiro Yamamoto, and Kouichi Kishida. The co-evolution of systems and communities in free and open source software development. In *Free/Open Source Software Development*, pages 59–92. Idea Group Publishing, Hershey, PA, USA, 2004.

Muhammad Imran Yousuf. The Delphi technique. *Essays in Education*, 20:80–9, Spring 2007. URL `http://www.usca.edu/essays/vol202007/yousef.pdf`.

Shaker A. Zahra and Gerard George. Absorptive capacity: a review, reconceptualization, and extension. *Academy of Management Review*, 27(2):185–203, April 2002. URL `http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=6587995&loginpage=Login.asp&site=ehost-live`.

Robert W. Zmud. Diffusion of modern software practices: Influence of centralization and formalization. *Management Science*, 28(12):1421–1431, December 1982.

Robert W. Zmud. An examination of 'push-pull' theory applied to process innovation in knowledge work. *Management Science*, 30(6):727–38, June 1984.

A Background information on Debian

A.1 Evolution of membership in Debian and the role of contributors

A.1.1 Debian's organisational structure

The Debian project is organised according to the structure described in its constitution (see section A.1.4). which establishes the decision-making bodies and the processes for making decisions within the project. Figure A.1 is a rough approximation of the structure of the Debian project.

Software in the Public Interest (SPI) is the legal entity of the Debian Project, founded by Bruce Perens in 1997 to hold all trademarks for Debian, own all of its monetary and material assets, and represent the project in legal matters. SPI has no authority over decisions cast within the Debian project.

The following sections recount the history of Debian with respect to the evolution of membership in the project.

A.1.2 1993–1996: The beginnings

In its early years, the Debian Project was just a small interest group, and most users of Debian GNU/Linux were themselves developers of the system Sowe et al. [*cf.* 2007], who knew each other and cooperated closely. During that era, anyone who took the time to express an interest in the development of the system was given access: Steve McIntyre recalls in his 2007 Debian Project Leader (DPL) platform that in 1996, he "installed Debian, then two days later [...] mailed Bruce Perens [the DPL at the time] with a PGP key and asked him for a Debian login. [Bruce] responded almost immediately with account details."[1]

[1] `http://www.debian.org/vote/2007/platforms/93sam` [3 Oct 2007]

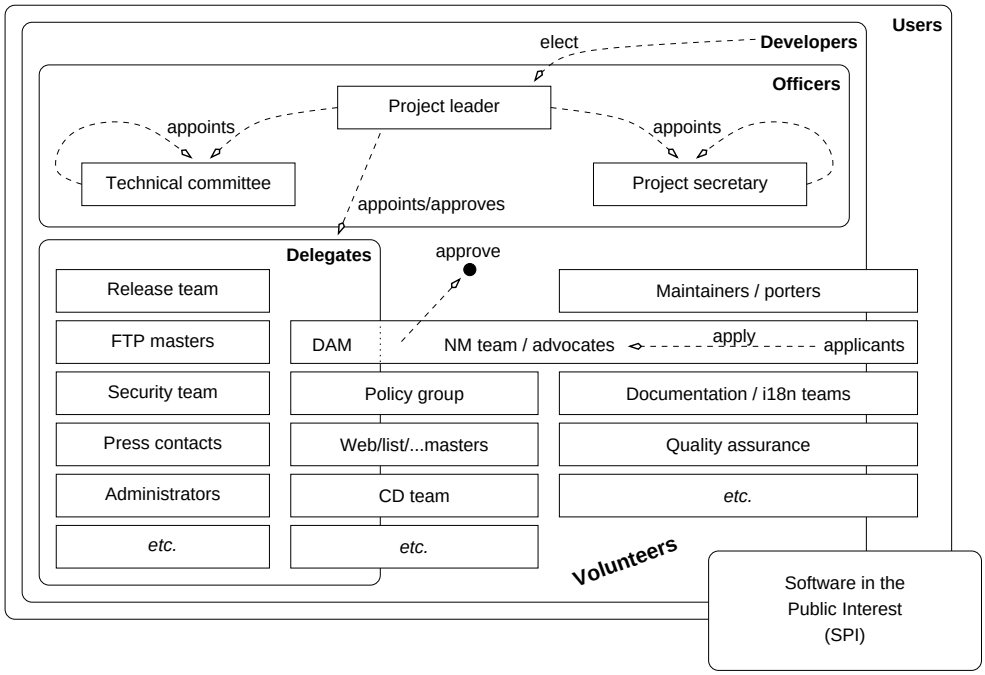

Figure A.1: A rough sketch of Debian's organisation

At that time, it was commonly accepted practice (but not policy) for members of the project to sign new versions of packages uploaded to the Debian archive with their respective PGP keys, which allowed for the verification of authenticity and integrity of new packages.

A.1.3 1996–1997: Exploding growth

With increasing media attention following the project's first public release, Debian 1.1 ("buzz"[2]), project members started to raise concerns over quality and security issues, as new "members were being admitted at rates faster than the project's ad hoc social systems could integrate them" [Coleman, 2005b]. At time of release of "buzz", the project had 90 members; at the time of release of the successive release, Debian 1.2 ("rex") six months later, this number had grown to 120.

The project started to require signed package uploads around the time of the release of Debian 1.3 ("bo"; June 1997; 200 members), and restricted the set of people allowed to make uploads to the official project members. But Debian's roster continued to

[2] Debian releases are named after characters of the computer-animation series *Toy Story*; see section A.3.

mushroom, minimising the effect of this new security measure. Long-winded, year-long discussions ensued on the project's mailing list, in which the topic of verification of new project members continuously percolated to the top. After Perens instituted Debian's infamous Social Contract (see section A.4) in 1997, "prospective members needed only to informally demonstrate their technical competence and to claim knowledge of and adherence to the project's Social Contract and policies before being admitted to the project" [Coleman, 2005b].

Over debates on membership verification and requirements, many started to voice concerns, such as Manoj Srivastava: "I'm not sure about competence or integrity requirements; some how [*sic.*] it goes against the grain for someone who is not issuing my paycheck."[3] Meanwhile, project leader Perens, who was solely in charge of accepting new members at the time, persistently made a case for the importance of identity verification to guard against rogue maintainers, and following the rejection of a number of applicants on the basis of trust issues, several members began questioning his authority and the role of the leader.

A.1.4 1998: The Debian Constitution

After taking over leadership in January 1998, the third DPL, Ian Jackson, initiated the public drafting of the Debian constitution in March to document the rights and obligations of project members and their leader.[4] The document was ratified by public vote in December 1998,[5] a few months after the release of Debian 2.0 ("hamm"); the number of project members had again doubled and grown beyond 400.

The Constitution [Debian Project, 2006] allows any member of the project to

- make any technical or nontechnical decision with regard to their own work;
- propose or sponsor draft general resolutions;
- propose themselves as a project leader candidate in elections;
- vote on general resolutions and in leadership elections.

[3] `http://lists.debian.org/msgid-search/87ybc8l3qo.fsf@tiamat.datasync.com` [6 Oct 2007]

[4] `http://lists.debian.org/msgid-search/E0yFXzq-00009z-00@anarres.greenend.org.uk` [6 Oct 2007]

[5] `http://www.debian.org/vote/1999/vote_0000` [6 Oct 2007]

The Constitution also manifests the volunteer nature of the project:

> Nothing in this constitution imposes an obligation on anyone to do work for the Project. A person who does not want to do a task which has been delegated or assigned to them does not need to do it. However, they must not actively work against these rules and decisions properly made under them.
>
> [...]
>
> A person may leave the Project or resign from a particular post they hold, at any time, by stating so publicly.

The document furthermore specifies the powers of the project leader and defines the procedure of leadership:

> The Project Leader should attempt to make decisions which are consistent with the consensus of the opinions of the Developers.
>
> Where practical the Project Leader should informally solicit the views of the Developers.
>
> The Project Leader should avoid overemphasizing their own point of view when making decisions in their capacity as Leader.

Finally, the Constitution places the member collective at the top, stating, among other rights, that "the Developers by way of General Resolution or election" may "appoint or recall the Project Leader, amend this constitution, [...] and make or override any decision authorised by the powers of the Project Leader or a Delegate. [...]"

The Constitution does *not* specify the following three privileges, which are generally considered part of the developer status, but which are granted by the project delegates specific in brackets:

- upload *any* package to the Debian archive management queue, where it will be processed and installed into the archive [FTP masters];
- access (or request access to) machines owned and administered by the Debian Project, so-called developer machines [system administrators];
- receive emails via the `debian-private` mailing list [list masters].

In the context of this thesis, I assume the definition of Debian's New Maintainer (NM) team: a Debian developer is "a Debian Project member, who has gone through the NM process and had their application accepted" [Debian Project, 2007a]. To make sense of this definition, we have to continue following the evolution of membership in the Debian Project.

A.1.5 1997–present: The New Maintainers process

The discussions on the verification of integrity and identity of new members led to the delegation of a new maintainer authority, consisting of two members, in April 1997.[6] After a wave of new applicants were accepted, first signs of stagnation showed in August of the same year,[7] until James Troup and Martin "Joey" Schulze "hatched a plan to basically hijack new-maintainer in order to rescue it from stagnation" [James Troup, cited in [Wallach et al., 2005]] in September. The two were in charge of processing applicants and made weekly announcements about new maintainers for the better part of 18 months. Shortly after Troup and Schulze took over, the project instituted the `debian-mentors` mailing list[8] as a forum where new maintainers could ask for help from the more experienced. Around that time, the concept of package sponsorship first surfaced. A sponsor is "a Debian Project member who acts as the mentor of an applicant: s/he checks packages provided by the applicant and helps to find problems, [...] to improve the packaging, [and to] upload [the package] on behalf of the applicant to the Debian archive. The applicant is recorded as the maintainer of such a package." [Debian Project, 2007a].

In July 1999, Schulze announced the freeze of the new maintainer process,[9] citing quality issues related to uncontrolled growth as the reason; instead of further growth, Debian should take a step back and return to quality as top priority. In addition, more rigorous checks and vetting of applicants was needed to maintain the integrity of the distribution, as well as to keep the workload for the NM team manageable. This freeze was meant to stay private but found its way to the public within a few weeks.[10]

[6] Message <m0wKAwZ-00IdecC@golem.pixar.com> to `debian-private` [15 Oct 2007]
[7] Message <19970818203920.51870@dungeon.inka.de> to `debian-private` [15 Oct 2007]
[8] `http://lists.debian.org/msgid-search/199711102134.QAA02106@hmm.nowhere` [7 Oct 2007]
[9] Message <19990702115419.D27941@finlandia.infodrom.north.de> to `debian-private` [15 Oct 2007]
[10] `http://lists.debian.org/msgid-search/19990805074250.a528@box` [6 Oct 2007]

The project was facing a crisis as project members and outsiders alike demanded the recommencement of new maintainer processing.[11]

Troup announced his intent to resign from NM in October. In response, the fourth DPL, Wichert Akkerman, who had stepped up to the throne in January, proposed the creation of a NM committee,[12] composed of "experienced Developers, with strong views in favour of free software, who could be trusted more than ordinary active Developers, and who understood the responsibility involved in accepting new Developers to Debian" [Wallach et al., 2005]. The proposal required applicants to identify themselves in person towards at least one project member, using an officially issued identification document; this identity verification was recorded by exchanging cryptographic signatures.

Under the lead of Dale Scheetz, the details of this proposal were worked out and NM officially relaunched towards the end of the year and ran smoothly for the next years, despite repeated complaints about the slowness of the process by applicants who were growing impatient.

In 2001, Martin Michlmayr, the seventh DPL, added the requirement for NM applicants to present a body of contributions with their application, to be backed by a project member ("advocate"), who would confirm the contributions and vouch for the new maintainer. In general, a package in the archive, maintained by the applicant, was sufficient. The concept of sponsorship was formalised.

A.1.6 2007: The Debian Maintainers role

In later years, the requirement for applicants to maintain a package was subject of many criticisms and statements of frustration as Debian's body of contributors grew to encompass translators, documentation writers, graphic designers, testers, people fixing bugs and submitting patches, and mailing list participants. Such contributions were generally appreciated: "It cannot be stated enough that it takes more than just package maintainers to make Debian work as a distribution."[13]. To the contributors, however, not being given access to project resources or the right to vote was not very motivational. Suggestions on instituting an official Debian contributor status

[11] Message <oaiu4nr4by.fsf_-_@burrito.onshore.com> to debian-private [15 Oct 2007]
[12] http://lists.debian.org/msgid-search/19991017121536.A6299@mors.net [6 Oct 2007]
[13] http://lists.debian.org/msgid-search/20020129214344.GA31327@netexpress.net [15 Oct 2007]

were made every couple of weeks,[14] but it took until July 2007 for these efforts to achieve the goal: in an official vote, the status of Debian Maintainer (DM) was created.[15]

A.2 Additional terminology

A.2.1 System

In the Debian world, one will often encounter the word "system." The Oxford English Dictionary[16] does not offer a concise definition, but Wikipedia defines it as "a set of entities, real or abstract, where each entity interacts with, or is related to, at least one other entity.[17] When applied in the Debian context, the word can take any of three meanings:

1. from the perspective of an individual user, a Debian system is a computer device controlled by one of the operating systems produced by the Debian Project, such as Debian GNU/Linux.
2. from the perspective of a system administrator, who manages multiple Debian systems, the Debian System comprises Debian's administration concepts and techniques.[18].
3. from the perspective of the Debian Project, its developers and contributors, the Debian System is the common vision of an operating system, observing Debian's foundation documents (see section A.4 and section A.2.5).

A.2.2 Packages

The fundamental unit of development in the Debian Project is the package (see [Krafft, 2005, ch. 5] for an exhaustive exploration of Debian packages). A package is a container for software to be installed on (or removed from) a target system. We need to distinguish between two types of Debian packages:

[14] *e.g.* `http://lists.debian.org/msgid-search/200308081807.46433@fortytwo.ch` [15 Oct 2007]
[15] `http://www.debian.org/vote/2007/vote_003` [15 Oct 2007]
[16] `http://www.askoxford.com/dictionaries/compact_oed` [4 Nov 2007]
[17] `http://en.wikipedia.org/wiki/System` [24 Oct 2007]
[18] This is the meaning to which I allude with the title of my book: *The Debian System — Concepts and Techniques*.

Binary packages — are what users install onto their system to augment its functionality. Packages are usually handled by programmes known as package managers, which keep track of the changes introduced by a package during installation, and hence allow for later manipulation of installed packages as a unit, *e.g.* enabling their traceless removal.

Binary packages consists of two components, which are concatenated into a single DEB file (extension `.deb`):

Payload/data — executable programmes, libraries, scripts, data, and configuration files installed onto the target system, which are necessary to run a programme, or which implement a set of related commands or features. Every Debian package's payload must furthermore include copyright information and a change log. Documentation and examples may be included.

Control information — meta data, which include: descriptions targeted at the human operator, dependency information relating the package to other packages, scripts to interact with the installing user, and scripts to fine-tune the integration of installed files with the rest of the target system, and to coordinate their later upgrade or removal.

Even though DEB files are best handled by the family of Debian package managers,[19] they can also be manipulated with standard Unix tools.

Source packages — provide the sources needed to generate ("build") binary packages. In this context, sources may include code to be compiled, data transformed to binary formats during the build process, or files transferred verbatim onto the target system. A source package consist of two or three files, which can be manipulated with standard Unix tools.

Most software in the Debian archive is developed by another FLOSS project (called "upstream"), but has been made available as a non-native Debian package. On the other hand, native Debian packages contain software developed specifically for the Debian System.[20]

[19] `dpkg` is the package handler (which prevents actions violating the Debian Policy; see section A.2.5), while APT!-based tools are higher-level package managers (which can use the packages' meta-data to pre-emptively resolve problems).

[20] In the case of a native Debian package, the source package consists of two files: the source archive ("tarball") and the `.dsc` file with control data; non-native packages are made up of three files: the original source "tarball", a patch with Debian-specific changes ("diff"), and the `.dsc` file with control data.

When users talk about "packages" without further specification, they likely refer to binary packages. On the other hand, developers usually refer to source packages.

One of the central activities of DDs is (source) package maintenance, which involves:

- importing new upstream versions.
- analyse bug reports, correspond with the submitter, forward information upstream, and/or fix the problem.
- look after the package's meta data and make sure that relationships to other packages are correct and current.
- follow and implement Debian Policy changes.
- develop new features, or better means of integration with the rest of the Debian System.

Section 2.3 illustrates the workflow of Debian package maintenance.

A.2.3 Bugs and bug reports

The Debian Project project makes extensive use of its publicly-accessible[21] BTS[22], publishes instructions on how to submit bug reports [Debian Project, 2007b], and maintains a number of interactive bug reporting tools to facilitate the process.

A bug report is made up of the original report and any number of replies sent to the bug report's address. All interaction takes place via e-mail. Reports and replies may include attachments. Every bug report in the Debian BTS belongs to one (or more) packages and has a unique ID, which is usually prefixed by a pound sign when mentioned in writing: *e.g.* #123456. Ideally, a bug report describes a single fault in the corresponding package.[23] For each report, the BTS tracks submitter, responsible person ("owner"), severity, tags, and the software versions to which this bug applies, all of which may be manipulated by anyone, but a log of all actions is kept.[24] The available severities are:

[21] The project promises in its Social Contract (see section A.4) that it "will keep [the] entire bug report database open for public view at all times."
[22] `http://bugs.debian.org`
[23] Debian bug reports can be split ("cloned") and merged to achieve this one-to-one correspondence.
[24] Manipulation happens via e-mail: `http://www.debian.org/Bugs/server-control` [11 Oct 2007]

critical, grave, serious, important, normal, minor, wishlist. The first three form the class of release-critical (RC) bugs.

In Debian parlance, "bug report" is often abbreviated to "bug", *e.g.* "have you replied to my bug yet?" In this thesis, an attempt will be made to use the separate terms accordingly. A bug is thus a fault in the software contained in the package, or its packaging, and bug report refers to the discussion thread following and including the initial report, encompassing the meta data tracked by the BTS.

A.2.4 Archives

When a project member uploads a package,[25] it is processed by the archive management scripts. These perform a number of tests on the package, checking *e.g.* for integrity and authenticity. If the package passes these, it is installed into the archive's package pool.[26] Package indices refer users (or rather: package manager software invoked by users) to the correct location within this pool.

The Debian archive comprises five different collections of package indices: unstable, testing, stable, oldstable, and experimental. These are themselves often referred to as archives. In this text, the use of the word "archive" in the context of Debian will be qualified: Debian archive refers to the entire Debian package pool and all indices, while *e.g.* the unstable archive identifies the collection of indices referencing packages considered to be unstable, together with these packages, and analogously for the other collections: testing, stable, oldstable, and experimental. The collection names by themselves are synonymous to the respective archives: "this package is not yet in testing." All archives except for the experimental archive can be used to install computer systems; a user tracking the unstable archive is said to run Debian unstable, or just "unstable".

Every package starts its life-cycle in the unstable archive, replacing any previous versions. It remains there for a given period of time to allow for potential problems to surface. If during this period, no new RC bugs are reported against the package, and a number of other criteria are met, the package "migrates to testing," which means that it propagates to the testing archive, again replacing any previous versions.

[25] *i.e.* the files that constitute a (signed source) package, together with a file identifying the upload (the `.changes` file), which also has to be cryptographically signed for the upload to be considered.
[26] `http://ftp.debian.org/debian/pool/`

Packages in testing remain in this archive until a new stable version is released. Leading up to such a release, the testing archive is frozen, and only packages fixing RC bugs are allowed to propagate from unstable. During the freeze period, developers are encouraged to concentrate on fixing bugs.

At the end of the freeze, packages with remaining RC bugs are reassessed and generally removed from the testing archive (they continue to exist in unstable). Finally, a new stable release is made: the current stable replaces oldstable, and testing replaces stable. A new stable release is thus made up of packages without any (known) RC bugs.

The experimental archive holds packages which are not yet polished enough to enter the regular package life-cycle. Project members may make use of it at their own discretion, and no automatic migration takes place.

Figure 2.1 (see page 12) depicts a package's life cycle.

A.2.5 The Debian Policy

The Debian Policy Manual is a fundamantal Debian document, which defines the "technical requirements that each package must satisfy to be included in the distribution" [Debian Project, 2005]. The manual helps to lessen the number of decisions a package maintainer has to make, and ensures that packages, which have been built in accordance with its rules, will be compatible with the Debian System and the tools used for its management. A compliant package can furthermore coexist with thousands of other packages without conflicts. Familiarity with the policy is required of each Debian developer, and policy compliance is a core requirement in package maintenance.

The Debian Policy has been described as

> the crux, the narthex, the throbbing heart of Debian and what makes it so utterly superior to all other operating systems. [...] What Policy defines are the bounds of Debian, not your own actions on the system. [...] In essence, Policy introduces a new class of bugs, policy bugs. Policy bugs are release-critical — **a package which violates policy will not be included in the official stable Debian release**. [...] That is the whole secret [Self et al., 2002, their emphasis].

A.3 Debian releases

Date	Version	Codename	# Developers	# Packages (S:B)	# Architectures	Notes
Aug 1995	0.96r3	n/a	60	250	1 (i386)	*a,b*
Jun 1996	1.1	buzz	90	474	1	*a,b*
Dec 1996	1.2	rex	120	848	1	*a,b*
Jun 1997	1.3	bo	200	1 197	1	*a,b*
Jul 1998	2.0	hamm	400	1 115: 1 958	2 (+m68k)	*a,c*
Jun 1999	2.1	slink	413	1 580: 2 269	4 (+alpha,sparc)	*c,d*
Aug 2000	2.2	potato	448	2 647: 3 889	6 (+powerpc,arm)	*c,d*
Jul 2002	3.0	woody	892	5 218: 8 273	11 (+mips/el,hppa,ia64,s390)	*c,d*
Jun 2005	3.1	sarge	1000	8 728: 15 196	11	*e,f*
Apr 2007	4.0	etch	1036	10 222: 18 110	11 (+amd64, -m64k)	*e,g*

Notes (a) (approximated) number of developers from [Lameter, 2002]; (b) (approximated) number of packages from [Lameter, 2002]; (c) number of packages from indices available at `http://archive.debian.org` [7 Nov 2007]; (d) number of developers from [Amor-Iglesias et al., 2005b]; (e) number of packages from indices available at `http://ftp.debian.org/debian/dists` [7 Nov 2007]; (f) number of developers from `http://www.debian.org/vote/2006/vote_003` [7 Nov 2007] for lack of a better number (see `http://bugs.debian.org/295527` [7 Nov 2007]); (g) number of developers from `http://www.debian.org/vote/2007/vote_001` [7 Nov 2007] for lack of a better number (see `http://bugs.debian.org/295527` [7 Nov 2007]);

A.4 Debian's Social Contract

The Social Contract is a foundation document of the Debian project [Debian Project, 2004b] and, to my knowledge, distinguishes the Debian project from most other FLOSS projects, which do not obey a manifest of this kind.

Coleman [2005b] recounts the genesis of the Social Contract as follows:

> Ean Schuessler came up with the idea for the Contract after a conversation at a conference with Bob Young, the co-founder of a then-emergent commercial Linux distribution, Red Hat. Ean suggested that Red Hat might want to guarantee in writing that as they grew larger they would always provide GPLed software. Young replied that "would be the kiss of death," implying that such a guarantee made to the users of free software could prove disastrous to his business. Ean (who was himself a business owner)

> was both amused and disturbed by Young's answer, and with other developers at the conference he decided that it would behoove Debian to provide such a guarantee in writing.

The Social Contract was drafted in 1997 by Bruce Perens, after Ean Schussler suggested the concept of a Linux distribution stating its "social contract with the free software community" to him. In a month-long e-mail conference, several other developers joined to refine the text, before Perens, project leader at the time, announced it as publicly stated policy of the Debian project.

The project ratified a number of changes via an official vote in the summer of 2004[27] but postponed their institution until after the release of Debian 3.1, codenamed "sarge" in June 2005.

The following is thus the updated version of the document, which applies to Debian's most recent release, Debian 4.0 ("etch").

Debian Will Remain 100% Free We provide the guidelines that we use to determine if a work is "free" in the document entitled "The Debian Free Software Guidelines" (see section A.5). We promise that the Debian system and all its components will be free according to these guidelines. We will support people who create or use both free and non-free works on Debian. We will never make the system require the use of a non-free component.

We Will Give Back to the Free Software Community When we write new components of the Debian system, we will license them in a manner consistent with the Debian Free Software Guidelines. We will make the best system we can, so that free works will be widely distributed and used. We will communicate things such as bug fixes, improvements and user requests to the "upstream" authors of works included in our system.

We Won't Hide Problems We will keep our entire bug report database open for public view at all times. Reports that people file online will promptly become visible to others.

[27] `http://www.debian.org/vote/2004/social_contract_reform.3` [26 Sep 2007]

Our Priorities are Our Users and Free Software We will be guided by the needs of our users and the Free Software community. We will place their interests first in our priorities. We will support the needs of our users for operation in many different kinds of computing environments. We will not object to non-free works that are intended to be used on Debian systems, or attempt to charge a fee to people who create or use such works. We will allow others to create distributions containing both the Debian system and other works, without any fee from us. In furtherance of these goals, we will provide an integrated system of high-quality materials with no legal restrictions that would prevent such uses of the system.

Programs That Don't Meet Our Free-Software Standards We acknowledge that some of our users require the use of works that do not conform to the Debian Free Software Guidelines. We have created "contrib" and "non-free" areas in our archive for these works. The packages in these areas are not part of the Debian system, although they have been configured for use with Debian. We encourage CD manufacturers to read the licenses of the packages in these areas and determine if they can distribute the packages on their CDs. Thus, although non-free works are not a part of Debian, we support their use and provide infrastructure for non-free packages (such as our bug tracking system and mailing lists).

A.5 The Debian Free Software Guidelines

The DFSG [Debian Project, 2004a] regulates the availability of software in Debian's archive according to its licence. Software, whose licence is in accordance with *all* terms of these guidelines, may be included in the official Debian archive. Software licenced incompatibly with these guidelines might also be distributed, but in a secluded section of the archive, called "non-free."

Free Redistribution The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

Source Code The program must include source code, and must allow distribution in source code as well as compiled form.

Derived Works The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

Integrity of The Author's Source Code The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (This is a compromise. The Debian group encourages all authors not to restrict any files, source or binary, from being modified.)

No Discrimination Against Persons or Groups The license must not discriminate against any person or group of persons.

No Discrimination Against Fields of Endeavor The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

Distribution of License The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

License Must Not Be Specific to Debian The rights attached to the program must not depend on the program's being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.

License Must Not Contaminate Other Software The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.

Example Licenses The "GPL", "BSD", and "Artistic" licenses are examples of licenses that we consider "free".

B Background information on diffusions

B.1 Rogers elements of diffusion

Rogers [2003] identifies four "elements" of diffusion:

- the innovation itself,
- the process of communicating the innovation (the diffusion),
- the adoption process (the time element),
- and the social system in which the diffusion occurs.

These elements of diffusion are often referred to as Rogers' framework, which has been used to assess and engineer diffusions (descriptive and prescriptive use). His tome is possibly the most cited work in diffusion research, even though it is not without problems (see section 3.2.2). It is thus in order that I offer a brief description of each.

Innovation The first set of characteristics in Rogers' theory deal with the individual adopter's perception of the innovation itself [Rogers, 2003, ch. 6]. These attributes are highly subjective, and singular in the sense that they pertain to each individual in isolation from the rest of the subjects targeted by a diffusion.

Rogers makes no claim that these are the only attributes of an innovation that affect the rate of adoption, just that they are the ones that "explained most of [an] innovation's rate of adoption" [*ibid.*, p. 226]. In a study by Kearns [1992, cited in [Rogers, 2003]], the five attributes of an innovation quoted by Rogers could account for 26% of the 27% of variance that could be explained by a total of 25 perceived attributes; the other 20 attributes hardly had any relevance in the eight diffusions analysed as part of the research.

Relative advantage — a subject is more likely to adopt an innovation that s/he perceives as having an advantage (economic, social, personal, ...) over the *status quo*.

Compatibility — an innovation compatible with the *status quo* — thus fitting in with social norms and requiring little effort — is likely to be more popular.

Complexity — innovations that are difficult to grasp, learn, or use, are likely to be adopted at a slower rate (accessibility).

Trialability — innovations that can be tried without much effort are likely to be adopted quicker, mainly because of the ability to reduce uncertainty about the innovation by using it (learning by doing).

Observability — innovations whose results are observable will likely be favoured among adopters (return on investment, gratification).

Communication channels Rogers defines communication as "the process by which participants create and share information [...] to reach a mutual understanding" [Rogers, 2003, p. 18]. Rogers isolates two aspects of the channels of communication:

Channel/medium — information may be exchanged face-to-face (interpersonal media), or pushed to a broad audience (mass media). While in the former case, communication is often two-way (discussions, question & answer sessions, presentations), mass media communication is usually a one-way means to reach a larger number of people (radio, newspapers, announcement lists). Innovations which are primarily communicated through interpersonal channels are likely to reach a higher adoption rate, because they are "more effective in persuading an individual," and since potential adopters "depend mainly upon a subjective evaluation [...] from other individuals like themselves who have already adopted the innovation" [*ibid.*, p. 18f].

Homophily — two peers who are "similar in certain attributes, such as beliefs, education, socioeconomic status, and the like" [*ibid.*, p. 19] are called homophilous.[1] With greater degree of homophily, communication between peers becomes more

[1]The peers must not be homophilous with respect to their technical grasp of an innovation, or no diffusion can take place.

effective, but the spread of ideas also becomes more horizontal (as it spans areas of competence of peers at the same status level), which can dampen the adoption rate.

Time The third element of diffusion describes the phases leading up to an adoption and including its verification, which Rogers calls the "innovation-decision process." The process begins with initial knowledge of an innovation and ends with the confirmation of the decision to adopt or reject this innovation by in individual (or other decision-making unit). The author presents a model that breaks the process into five stages [Rogers, 2003, ch. 5]:

Knowledge — at the first stage, the individual learns of an innovation and how it works. The knowledge may be gained following a need, or the need may be developed as a consequence of the knowledge. At this stage, knowledge is mostly cognitive (emotional, non-verbal), but the individual already starts to investigate how the innovation functions.

Persuasion — after being exposed to an innovation, an individual forms an interest or disinterest therein. In case of interest s/he actively seeks to reduce the uncertainty about an innovation's expected consequences. The knowledge of the innovation becomes affective (factual).

Decision — the individual then decides to adopt or reject the innovation, possibly after trying it or witnessing a demonstration. A favourable decision means the individual will move to incorporate the innovation into ongoing practice. A rejection the innovation may be either active or passive: deciding against the innovation *vs.* never considering it.

Implementation — in the forth phase, the individual incorporates the innovation into ongoing practice. At this stage, re-invention[2] can occur. The implementation phase ends when the innovation has become a regular part of current operations.

Confirmation — finally, the individual may confirm the decision of adoption (to reduce dissonance), or reverse it (discontinuance). This stage is not exhibited by all adopters, but it can also take place long after adoption has occurred, when the individual ceases to use the innovation, gradually discontinues, or replaces the innovation with a better solution.

[2] Rogers defines re-invention as the "degree to which an innovation is changed or modified by a user in the process of its adoption and implementation" [Rogers, 2003, p. 180].

While developing the time element, Rogers also identifies the source of information as a trait of the communication process that influences the success of a diffusion [*ibid.*, p. 207]: cosmopolite channels transport information into a system, whereas localite channels are contained within the system. This characteristic overlaps functionally with homophily, a feature of the second element of diffusion, which may be the reason why Rogers did not include cosmopoliteness in his framework.

Social system The fourth element of diffusion deals with the social system in which a diffusion is taking place and is divided into the following aspects [Rogers, 2003, p. 23ff]:

Structure and communication — applies to both, the social structure within an organisation (pecking order), as well as the communication structure (interaction networks). A communication structure usually (and frequently) grows between homophilous people to satisfy their social needs; it mostly relies on interpersonal exchange. The hierarchical organisation of a system, which is often definitive of the system's social structure, allows for the prediction of behaviour to a certain degree, given that individuals at lower ranks are expected to carry out orders issued by those higher up. This flow of command and the separation of responsibilities between levels serve to decrease uncertainty surrounding the process, which can benefit diffusions; communication down the hierarchy can be either interpersonal or public, whereas it will be almost exclusively interpersonal upwards [*ibid.*, p. 337ff].

Norms — refer to the established standards among members of a social system and relates to the compatibility attribute of an innovation (see section B.1): unless an innovation squares with the norms of the targetted social system, its diffusion will be less successful [Rogers, 2003, p. 26].

Opinion leadership — is the "degree to which an individual is able to influence other individuals' attitudes or overt behaviour informally in a desired way with relative frequency" [*ibid.*, p. 27], a status which is attributed by the respecting followers rather than determined by structure, and does not have to be explicit. Opinion leaders are thus at the centre of interpersonal communication networks and maintain this position through technical competence, social accessibility, and conformity with the system's norms. They are generally more innovative and cosmopolite, and enjoy a higher social status than the average member of the system. Nevertheless, they are not the most innovative members of the system,

who are often perceived as deviants and attributed low credibility [*ibid.*, p. 26]. Obviously, opinion leaders do not have to agree with each other: an opinion leader could also reject an innovation.

Change agents — are individuals who influence clients' innovation-decisions in a direction deemed desirable by a change agency, whether the goal is adoption or rejection [*ibid.*, p. 366]. They are usually professional and heterophilous, which might limit the effectiveness of their communication. For this reason, change agents sometimes employ aides, who are less professional and more homophilous, to facilitate communication with the client.

Types of innovation-decisions — Depending on the nature of the social system, three motivations can drive adoptions. In general, these are to be seen as a continuum, and diffusions can lie anywhere between these motivations:

optional/voluntary — the decision to adopt is made by the individual at his/her own discretion and independent from the other members of the system.

collective — the adoption decision is made by consensus among the members of the system.

authoritarian — the decision is made by a few at high levels of the social structure and the innovation must be adopted by all members of the system.

Decisions made by authority are usually adopted the fastest, collective decisions the slowed [*ibid.*, p. 29]. Given that in those two cases, the individual still has a decision to make (to go with the flow *vs.* rebel against it), a forth type of decision is inherent: contingent decisions, made by the individuals only after a collective or authoritarian decision has been made.

Consequences of innovations — Rogers attributes the study of consequences an ever increasing importance in diffusion research [*ibid.*, p. 440] and differentiates between three dimensions of consequences of diffusions: desirable/undesirable, direct/indirect, and anticipated/unanticipated, and these usually go together in that anticipated consequences are usually direct and desirable, while unanticipated ones are undesirable and indirect. These dimensions are per consequence, not per diffusion; any given diffusion will have a number of consequences, some of which may be desirable, others less so [*ibid.*, p. 442ff].

Another distinction of consequences is between form, function, and meaning; while the first two are more readily assessable by diffusers and adopters alike,

a grasp of the latter is only really available to the adopters themselves [*ibid.*, p. 451].

B.2 Wejnert's integrated model of innovation diffusion

The following is a slightly more thorough description of the variables of the framework proposed by Wejnert [2002], which was introduced in section 3.2.2.

Characteristics of innovations

public versus private consequences — This variable deals with the spatial and temporal contingency between source and adopter, including aspects of communication, coercion, and peer/social pressure.

benefits vs.costs — Costs can be monetary or non-monetary, direct or indirect. For instance, learning time is a direct, non-monetary cost, while possible infrastructural changes and delays would be considered indirect costs.

Benefits are not explicitly mentioned in the model.

Characteristics of innovators

societal entity — The size of target group for a diffusion, as well as the strength of social ties affect the nature of the diffusion process.

familiarity with the innovation — People are naturally risk averse and cautious towards the new. Social networks, opinion leaders, and the media come into play here to increase familiarity, which drives adoption. Sources closer to an individual (and thus more subjective) are usually more effective.

status characteristics — An individual's status determines the likelihood of adoption. Lead users [*cf.* von Hippel, 1986] adopt early and drive others to adoption, while those with lesser social status and less prestige to lose might adopt controversial innovations.

socioeconomic characteristics — This category includes variables of the individual, such as education level and cosmopolitanism, as well as variables relating to the collective, such as technological advancement. Labor market practices, and the

consideration whether economical decisions are centralised or market-oriented are part of the latter.

position in social networks — Network interconnectedness (size, distance, frequency, level of privacy), helps spread adoptions, as well as pressure to conform to peers in tight networks. Structural equivalence of members modulates homogeneity of behaviours, and may also activate competition [*cf.* Robertson and Gatignon, 1986].

personal characteristics — Self-confidence, as well as independence, readiness to take risks, and exposure of individual success drive adoption rates; collective interaction and collaboration also have a positive effect.

Environmental context

geographical setting — Geographic proximity affects the frequency and level of communication and facilitates spreading. Population density is closely related.[3]

societal culture — Local cultural values and norms can account for inertia effects, slowing adoptions, and increase perceived costs of adopting an innovation. On the other hand, traditions provide homogeneity.

political conditions — The political situation can have dampening effects on adoption rates (promotion of local technology, patents, censorship). The author does not identify positive effects.

global uniformity — Institutionalisation, world-wide spread facilitated by internationally active corporations, and global communication channels and media constitute this variable.

[3]The literature generally speaks of geographic proximity, but that term does not make much sense in an Internet setting. It is arguable, whether the flow of information and thus the rate of adoption is actually *higher* in Internet-connected groups. However, geographical proximity also suggests cultural proximity and thus homogeneity. While a group such as the Debian developers is rather homogeneous, the impact of cultural difference to the spread of ideas might be significant.

C Acronyms & abbreviations

AC	absorptive capacity
APT	Advanced Package Tool
BSP	bug squashing party
BTS	Bug Tracking System (for Debian: `http://bugs.debian.org`)
CSCW	Computer Supported Cooperative Work
DD	Debian Developer
DEB	The Debian package file format
DFSG	Debian Free Software Guidelines
DM	Debian Maintainer
DPL	Debian Project Leader
FLOSS	Free/Libre/Open-Source Software
FSF	Free Software Foundation (`http://www.fsf.org`)
GNU	GNU is not Unix (a recursive acronym)
GPG	GNU Privacy Guard (`http://en.wikipedia.org/wiki/Gpg`)
IRC	Internet relay chat
IS	information systems
IT	information technology
NM	New Maintainer
NMU	non-maintainer upload
OSI	Open Source Initiative (`http://www.opensource.org`)
OSS	Open-Source Software
PGP	Pretty Good Privacy (`http://en.wikipedia.org/wiki/Pretty_Good_Privacy`)

RC	release-critical
SPI	Software in the Public Interest (`http://spi-inc.org`)
TAM	Technology Acceptance Model
TRA	Theory of Reasoned Action
TPB	Theory of Planned Behavior
UTAUT	Unified Theory of Acceptance and Use of Technology
VCS	version control system