2018-01-01

Algorithmic Sovereignty

Denis Roio

5 Devuan: the anatomy of a fork

Table 6: Information about the Devuan project, fork of Debian

Project name	Devuan
Home online	https://devuan.org
Started in	2014
Nature	Software
Advancement	Participatory
Media footage	https://www.youtube.com/watch?v=wMvyOGawNwo
Source code	https://git.devuan.org
Mail forum	https://lists.dyne.org/lurker/list/dng.en.html
Web forum	https://dev1galaxy.org

This project is to further evolve my research question on the assumptions laid down so far. Let us ask now: what are the traits for a (huge conglomerate) of algorithms to acquire the dimension of a sovereignty controlled by its participants? How does a socialised truth looks like, beyond the impossible assumption of universal neutrality? Beyond its mere execution, what in the design of an operating system generates or erodes the trust people put into it?

In this chapter I will describe at length the technical and socio-political conditions for the birth of a new GNU/Linux operating system. The widespread resistance to the introduction inside Debian of a new central framework for system management, called "systemd", is an extremely interesting opportunity to explore such dynamics in details. I've engaged this project in first person and this chapter is mostly written in the form of participatory research, reporting various accounts contributed by people who involved themselves into the creation of a new system, a sort of independent exodus for half of the active Debian user population.

The analysis of this episode is important to provide an advanced answer to my research question. As demonstrated, the design and adoption of software algorithms has a close relation to the design and adoption of systems of governance. In the experience of Bitcoin, it is evident how the fundamental trait of such a governance is the concept of neutrality, which has much to share with the definition of truth.

In the experience of Devuan it becomes clear that, even having a "democratic" system that relies on majority voting for the main choices affecting the design of its algorithms, as its the case for Debian, is not a sufficient condition to define the participants to this system as exercising sovereignty on the algorithms. In this particular case not only the needs of a 49% minority were disregarded, but the backward compatibility of the system was broken (Ellison and Fudenberg, 2000). In

case of Debian this became a sufficient condition to motivate the enormous effort of a fork.

After taking part as a leading participant into this initiative, for my own need to avoid systemd in production-level and mission critical projects, it became evident how this experience opened a new useful direction to explore in practice what I want to be a more complete and detailed answer to my research question.

This chapter will briefly outline the Debian project, which is worldwide renown and has been covered by valuable research quoted here. Then will proceed documenting the decision that in 2014 has torn the Debian community (and other GNU/Linux communities at large) to the point of making people switch (especially sysadmins) to platforms like BSD. Such episode has more or less marginally started a process of erosion for the trust in the Debian distribution. The tearing decision that was taken regarded the adoption of systemd, a new daemon managing the whole OS in place of the historical SysV scripts. The resentment and frustration for the faction against that adoption justifies in the eyes of many the creation of the Devuan fork.

The topic is still extremely controversial and as usual the voices of those who have "lost the battle" are the less heard in the future of the Debian project, sometimes even derided. This chapter provides an historical archive of some of the wisest voices in the debate, with a selected series of witnesses providing headspace for the comprehension and interpretation of this literature in terms of sovereignty, subjectivity and self-determination in technology.

5.1 The Debian project

Debian is one of the most popular GNU/Linux distribution and a uniquely big volunteer driven effort to build a UNIX-like operating system that can run on most computers or electronic devices and its mostly made of free and open source software. Debian calls itself a "universal operating system" and covers an impressive quantity of target architectures and has been adopted as a base for many more distributions. It was first created in 1993 by its now defunct author Ian Murdock and later supported by the Free Software Foundation and the GNU project (Murdock, 1994). It gathered a fast-growing community of passionate people who work on it on a voluntary basis, with the main incentive of reputation. As of today, Debian has access to online repositories that contain over 50,000 software packages making it one of the largest software compilations in the world.

Debian and the fork of it into Devuan is an extremely relevant subject for this thesis because, as other researchers pointed out:

> Debian is one of the largest F/OSS projects, although it is also one that has received very little academic analysis to date. Debian is a non-commercial

> version of the GNU/Linux operating system maintained by over 900 volunteer developers who package a wide variety of software applications. With the possible exception of the Free Software Foundation's (FSF) GNU project, Debian demonstrates the most overt commitment to the principles of free software as originally expressed by the FSF and the GNU General Public License (GPL) of any large F/OSS project. Debian developers have modified and extended these principles in their social contract and Debian Free Software Guidelines (DFSG).3 While in some respects, Debian is unique in its explicit ethical codes, its uniqueness should not obscure its wider relevance. The type of moral cultivation that forms an important facet of the Debian social sphere is a component of many other F/OSS projects in less accentuated forms. Debian is ideal to analyze because it brings into clear relief the subtle yet significant processes of ethical socialization that occur in most F/OSS projects. Though we focus on the domain of F/OSS, this case study also serves as an example of the unique ways in which an ethical social order is built and sustained in the "immaterial" setting of the Internet. (Coleman and Hill, n.d.)

The strong commitment of Debian to software freedom has gathered through the years what we can consider a large population of ethically attuned users and developers. While there are other GNU/Linux and BSD software distributions that have refined their ethical commitment to freedom,[12] Debian is most notable because of its size and outreach, combined to the fact it is an ethically-oriented volunteer-based project.

> F/OSS developers' attitudes toward freedom, set in broad terms in the larger F/OSS community and reinforced through the broad hacker public sphere, are particularized and reinforced in the context of individual F/OSS projects. [...] Our data shows that over the course of participating in the Debian project, developers move toward a more vigorous and overt ethical stance toward the uniqueness of their project and the importance of free software than when first joining. (Coleman and Hill, n.d.)

At the base of Debian there is its "Social Contract" with guarantees which include:

- Ensuring that the operating system remains open and free.
- Giving improvements back to the community which made the operating system possible.
- Not hiding problems with the software or organization.
- Staying focused on the users and the software that started the phenomena.
- Making it possible for the software to be used with non-free software.
- A set of guidelines for distributing free software.

The Debian project ratified its social contract version 1.0 on July 5, 1997. This document served to establish a clear differentiation when compared to other big GNU/Linux distributions, most importantly Red Hat, which have a corporate gov-

[12]The Free Software Foundation maintains a list of 100% free GNU/Linux and BSD distributions at the webpage address https://www.gnu.org/distros/free-distros.en.html

ernance and are characterised as business ventures, arguably with the merit of demonstrating F/OSS can be commercially sustainable and socially rewarding.

Debian changed a lot since its inception and the episode we describe here is undoubtedly the biggest crisis this initiative has ever suffered in terms of technical choices and community reaction to them. It was a practical schism between supporters and detractors of the switch to systemd, that split the community in half.

[13] Debian is not considered a 100% free operating system because it still allows the installation of non-free software for certain tasks, according to the GNU project.

5.2 The legacy init system: sysvinit

For a long time, the services handler called System V Init (also known sysvinit) has been responsible for the initialization of services after a reboot. Sysvinit is considered a "style" of system startup configuration, it was derived from AT&T's UNIX System III and later adopted into its System V from which it takes the name. Sysvinit is based on runlevels: at any moment the running operating system is considered to be in a specific state, numbered from 0 (shutdown) to 6 (reboot). Switching from one runlevel to another causes a per-runlevel set of scripts to be run, which typically mount filesystems, start or stop daemons, start or stop the graphical desktop, etc.

With each new release of GNU/Linux and BSD distributions, this init system, comprised of many scripts that have been written by different people and has been stratified on top of each other through decades to solve various problems, started to look more like an antique to many people.

One of the fundamental issues mentioned as an argument against the continued use of sysvinit is that it runs in a linear fashion, which is hardly efficient when parallelisation is possible.

Until approximately 2010 sysvinit was the init system of choice for most distributions and it was supposed to cover only the init role in operating systems, packed as a distribution of readable scripts written in procedural programming style and contributed to the project by various people around the world. Such scripts would often take values from configuration files, or carry the configuration values within themselves.

The sysvinit package weights approximately 560 Kilobytes and is comprised of 75 files and 15.000 lines of code.

[13] The Free Software Foundation maintains a list of 100% free GNU/Linux and BSD distributions at the webpage address https://www.gnu.org/distros/free-distros.en.html

5.3 The controversial innovation: systemd

Systemd was born with the main goal of unifying basic GNU/Linux configurations and service behaviors across all distributions. It has the ambition of verticalizing the integration of calls and access rules in a single monolitic system that spans from the kernel layer (ISO/OSI layers 3, 4 and 5) to the presentation and application layer (ISO/OSI layers 6 and 7). Quoting authors on the intentions behind system development:

> We try to get rid of many of the more pointless differences of the various distributions in various areas of the core OS. As part of that we sometimes adopt schemes that were previously used by only one of the distributions and push it to a level where it's the default of systemd, trying to gently push everybody towards the same set of basic configuration.

While systemd is most known for substituting the sysvinit and other init systems across distributions, this is not what it simply is. It is comprised of approximately 69 binaries and covers several roles as a central point that offers command and control operations to other software running in application space. Its main goal is to unify the interaction interface, standardize the code that handles the tasks covered using C language (compiled, hence not directly readable in production environments) and ultimately handle several base and mission-critical tasks as network configuration, logging of operations, virtualization setup, user login authentication.

Systemd also interfaces new specific Linux Kernel features as cgroups and is de-facto the only alternative that is actively developed to cover such new features, mostly due to lack of investment from other companies in the labour needed to achieve such results.

What is also notable is that systemd records initialization instructions for each daemon (background running software) in a configuration file (referred to as a "unit file") that uses a declarative language, replacing the traditionally used startup shell scripts written in a procedural language that is not compiled and therefore can be easily read an modified directly even on a running system.

The systemd software is also bound to a US patent 20150040216-A1 "Systems and Methods for Restricting Application Binary Interfaces" filed by Paul Moore, Dan Walsh and Lennart Poettering on behalf of Red Hat inc.

Arguably, referring to the patent filed by its main author, systemd may adopt this approach as the underpinning of its security model: lock down the functioning of the system in a binary architecture that, despite systemd being opensource, cannot be changed (nor verified) in production environments.

This approach is regarded as naive by many and presents several flaws which may appear on the long term, at least this is the opinion I share with most of the

community of professionals that is trying to resist the pervasive adoption of systemd in most GNU/Linux distributions. Yet regardless of us being right or wrong about this assessment, what is really important in the eyes of many people is the freedom of choice.

In 2014 and after a much heated debate, the chosen replacement by Debian was systemd. Debian followed as one of the last distributions to make the switch, as the controversial move was already made by other distros back in 2012 and in every single episode this caused considerable grief between supporters and detractors.

At the time of writing, systemd depends from other two big software components, dbus and glib, and is made out of 900 files and 224.000 lines of code, for a total weight of 125Kb source-code written in C language.

5.4 Summary of arguments for sysvinit

The systemd software presents several arguments for its adoption, mentioning advantages based on its ease of use, integration with application space and uniform API. There are interesting arguments also by those trying to defent the old system, sysvinit, as they do not touch the rethoric of innovation and efficiency, but try to defend the diversity and integration of different systems.

Referring to the debate that has taken place within the Debian users and developers community[14], the perceived virtues of sysvinit can be summarised as follows:

- It is integrated in the Debian ecosystem and this integration is well tested.
- It works on all supported architectures and kernels
- Has a long history (System V R4, 1988?), and so is widely used, well documented and understood by users of UNIX-like systems; /etc/rc.d initscripts are still used today by FreeBSD (since at least 1999), NetBSD since v1.5, OpenBSD uses something similar since v4.9
- Is the smallest and simplest initsystem available
- Init scripts can be improved using abstraction and addtional features can be implemented as suplemental software that could be used from shell scripts.
- Many people using sysvinit know how to fix an unbootable system: boot with init=/bin/sh and you can edit or invoke the initscripts by hand.
- Init scripts are a standard interface for many other purposes. Things like heartbeat, pacemaker, ... can all call initscripts directly.
- Init scripts are most easy to understand and debug. All important logic is contained in them, code is very readable, intuitive and simple and debug outputs can be inserted everywhere.

The most prominent arguments against its change are summarized by the following

[14]The summary reported here is based on information published on the Debian wiki and only slightly edited for the purpose to give a general overview on the issue.

points:

- Any change at all involves work, learning something new, and therefore should be a choice; this decision feels somewhat forced upon us by some who insist it is better for us.
- If GNOME[15] requires logind which requires systemd, it is unreasonable to impose such a change on the whole distribution, and ought to find an alternative which doesn't.

The consequences of a change from sysvinit to systemd were assessed as follows:

- Estimates are that up to 1200 Debian packages may be required to adapt to, and continue to support, such a change.
- If packages remove their current initscripts, ports like kFreeBSD and Hurd may be unable to use software that was working before this. (Unless the new initsystem can be ported; this is most likely for OpenRC but completely ruled out for systemd).
- Much third-party software outside of the Debian archive, or in-house projects within business, may no longer work without taking effort to port them to a new initsystem.
- Some maintainers object to maintaining separate initscripts to support alternate systems.
- Declarative systems are nice to look at and might cause fewer problems, but if there is a problem, they are almost impossible to debug. Even more so if you only have a non-working system (as the problem would be in the init system).
- At least judging from their documentation they all support significant logging.
- Even given the disadvantages of editing init scripts (hard to merge with upgrades, ...) and the resulting reluctance of admins to edit them directly, almost every admin has used this ultima ratio at some time, which indicates the missing flexibility to do so might be a problem with other init systems.

Further, here a summary of arguments that were put forward against systemd:

- Not portable, by design; uses Linux-specific functionality not available on other platforms.
- Binds Debian to being a Linux distribution; even some Linux ports may have to be dropped if systemd stops supporting them.
- Violates traditional UNIX principles of simplicity and the separation of kernel/initsystem/userland duties such that components are interchangeable.
- Begins a precedent for disruptive change, which may happen again if systemd changes its APIs or deprecates some interfaces (has been described as resembling 'vendor lock-in'; really meaning we must continue to keep up with changes mandated by upstream).

[15] A graphical and widely integrated desktop framework comprising many user-facing functionalities for GNU/Linux desktops.

- Not all functionality can be controlled by configuration files and is instead contained in compiled code and if you need to change something it is difficult.
- Principle of all-mighty solutions is in its nature authoritarian. All-in-one "solutions" are a characteristic of big companies that want to seduce their users into a vicious circle of using their integrated software, while sacrificing other tools that would give them more benefit, thereby realizing abusive psychological advantage over the user.
- Init system supported by a big company may have better support but that support is influenced by that company's interests and may not align with the community values.
- They depend on DBus, which means that if DBus is not working system initialization cannot begin.
- Key feature of free software is forking, which saves the software from falling into hands of corrupted people. When a project is corrupted, good developers can fork it and make a new project. But if a software is too complex, then forked project is either also complex, and that could demotivate developers to work on such project.

5.5 The General Resolution vote

When the pressure built up towards the decision of renewing the init system in Debian, the debates became absolutely heated and in many instances the community gave the worst show of itself, from both factions. This was so because the decision is an extremely important one which will arguably change the course of life of Debian as well that of many other distributions.

To tame down the debate to reason and call for a vote among the responsible developers, on the 16th October 2014 a Debian developer, Ian Jackson, called for a "General Resolution" vote with the following proposal. His proposal did not aim to affect the choice of the new init, but mostly the way the change should be handled, respecting the freedom of users and not imposing the change on everyone:

```
0. Rationale

  Debian has decided (via the technical committee) to
     change its
  default init system for the next release. The
     technical committee
  decided not to decide about the question of "coupling"
     i.e. whether
  other packages in Debian may depend on a particular
     init system.
```

```
  This GR seeks to preserve the freedom of our users now
     to select an
  init system of their choice, and the project's freedom
     to select a
  different init system in the future. It will avoid
     Debian becoming
  accidentally locked in to a particular init system
     (for example,
  because so much unrelated software has ended up
     depending on a
  particular init system that the burden of effort
     required to change
  init system becomes too great). A number of init
     systems exist, and
  it is clear that there is not yet broad consensus as
     to what the
  best init system might look like.

  This GR does not make any comment on the relative
     merits of
  different init systems; the technical committee has
     decided upon the
  default init system for Linux for jessie.

1. Exercise of the TC's power to set policy

  For jessie and later releases, the TC's power to set
     technical
  policy (Constitution 6.1.1) is exercised as follows:

2. Loose coupling of init systems

  In general, software may not require a specific init
     system to be
  pid 1.  The exceptions to this are as follows:

   * alternative init system implementations
   * special-use packages such as managers for init
      systems
```

```
   * cooperating groups of packages intended for use
     with specific init
     systems

  provided that these are not themselves required by
     other software
  whose main purpose is not the operation of a specific
     init system.

  Degraded operation with some init systems is
     tolerable, so long as
  the degradation is no worse than what the Debian
     project would
  consider a tolerable (non-RC) bug even if it were
     affecting all
  users.  So the lack of support for a particular init
     system does not
  excuse a bug nor reduce its severity; but conversely,
     nor is a bug
  more serious simply because it is an incompatibility
     of some software
  with some init system(s).

  Maintainers are encouraged to accept technically sound
     patches
  to enable improved interoperation with various init
     systems.

3. Notes and rubric

  This resolution is a Position Statement about Issues
     of the Day
  (Constitution 4.1.5), triggering the General
     Resolution override
  clause in the TC's resolution of the 11th of February.

  The TC's decision on the default init system for Linux
     in jessie
  stands undisturbed.
```

```
  However, the TC resolution is altered to add the
     additional text
  in sections (1) and (2) above.
```

Now this General Resolution proposal is a portrait of what we define an issue of "Algorithmic Sovereignty", its importance is confirmed by the facts that followed it.

The vote involved all "Debian Developers" took place on in November 2014 and offerent 5 different options to vote upon. Its outcome was calculated using the Schwartz criterion (Schwartz, 1972): assuming each voter must furnish an ordered preference list on candidates, the winning choice is the one preferred by a majority over every other choice in pairwise comparisons.

```
Starting results calculation at Wed Nov 19 00:02:03 2014

Option 1 "Packages may not (in general) require a
   specific init system"
Option 2 "Support for other init systems is recommended,
   but not mandatory"
Option 3 "Packages may require specific init systems if
   maintainers decide"
Option 4 "General Resolution is not required"
Option 5 "Further Discussion"

In the following table, tally[row x][col y] represents
   the votes that
option x received over option y.

                  Option
              1     2     3     4     5
            ===   ===   ===   ===   ===
Option 1          133   183   147   241
Option 2    313         251   180   366
Option 3    263   172         135   286
Option 4    323   280   308         358
Option 5    212    99   166    95



Looking at row 2, column 1, Support for other init
   systems is
recommended, but not mandatory received 313 votes over
```

```
   Packages may
not (in general) require a specific init system

Looking at row 1, column 2, Packages may not (in
   general) require a
specific init system received 133 votes over Support for
   other init
systems is recommended, but not mandatory.

Option 1 Reached quorum: 241 > 47.5762545814611
Option 2 Reached quorum: 366 > 47.5762545814611
Option 3 Reached quorum: 286 > 47.5762545814611
Option 4 Reached quorum: 358 > 47.5762545814611


Option 1 passes Majority.               1.137 (241/212)
   >= 1
Option 2 passes Majority.               3.697 (366/99)
   >= 1
Option 3 passes Majority.               1.723 (286/166)
   >= 1
Option 4 passes Majority.               3.768 (358/95)
   >= 1


  Option 2 defeats Option 1 by ( 313 -  133) =  180
     votes.
  Option 3 defeats Option 1 by ( 263 -  183) =   80
     votes.
  Option 4 defeats Option 1 by ( 323 -  147) =  176
     votes.
  Option 1 defeats Option 5 by ( 241 -  212) =   29
     votes.
  Option 2 defeats Option 3 by ( 251 -  172) =   79
     votes.
  Option 4 defeats Option 2 by ( 280 -  180) =  100
     votes.
  Option 2 defeats Option 5 by ( 366 -   99) =  267
     votes.
  Option 4 defeats Option 3 by ( 308 -  135) =  173
```

```
    votes.
  Option 3 defeats Option 5 by ( 286 -  166) =  120
    votes.
  Option 4 defeats Option 5 by ( 358 -   95) =  263
    votes.


The Schwartz Set contains:
     Option 4 "General Resolution is not required"



-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

The winners are:
     Option 4 "General Resolution is not required"

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

To this result has followed the resignation of Ian Jackson, proponent of the vote and reference figure for all those who wanted to limit the impact of the systemd init switch:

5.6 Response to the GR vote

The outcome of the GR vote has been harshly criticized: by a small percentage of vote the winning option has been that of not needing a general resolution on the issue. The effect it sorted was not just of ignoring the matter put forward by more than a few concerned developers, but practically showing the fact that a large group of Debian volunteers was not willing to acknowledge any concerns on the consequences caused by the change of init.

An active and well respected moderator of Debian's forum going with the moniker of "dasein" has also resigned to his position and ran an independent analysis on the vote which shows that, beyond the curtain of the 4th option to not consider the issue there was a majority of votes which have selected that option together with a negative view on what systemd would be.

Here the issue of power in using algorithms becomes recursive, as in the very count of this vote there is another prevarication denounced by the community, referring to the way the options were presented and the votes counted.

> The results of the GR vote were diluted and obfuscated by two non-resolution outcomes. Of the three technologically-relevant resolutions to the GR, one was unequivocally pro-systemd, the other two were contra-systemd, differing primarily in phrasing (essentially the difference between "must not" and "should not").

Dasein denounced that, if we ignore for a moment the 4th option which is formulated as "no GR required" and captivates the frustration of most people involved after more than a year of incendiary debates, then the winning option is the first: "Packages may not (in general) require a specific init system". This option basically meant that freedom is maintained with regards to init system dependency and that packages requiring the OS to run a particular init system shouldn't be deemed as valid and included until they can be independent from it.

This simple statement not only aligns with Debian's mandate and commitment for user freedom, but creates a condition of separation of dependencies between the different layers of the operating system, de-facto insuring that one single program does not propagates its dependencies everywhere as systemd is doing. Here below is reported dasein's analysis.

> The GR vote was structured as a rank-ordering of five choices. These are ordinal data and need to be treated as such. To be clear: I'm saying that any attempt to demote these data to nominal, or to promote them to interval, is an act of intellectual dishonesty, deliberate or otherwise.
>
> The five choices can be broken down into two discrete and non-overlapping subsets: three choices that take a technical position on systemd/gratuitous init coupling (S/GIC), represented as:
>
> 1) MUST not
> 2) SHOULD not
> 3) MAY
>
> And two choices that do not take a technical position, summarized as: 4) "I don't want to talk about this." 5) "Let's talk about this some more."
>
> Again, each vote was a ranking of all five options, which means that each DD's vote can be represented as a vector in 5-dimensional space.
>
> In case it isn't obvious, both #4 and #5 are identical in that they are talking about the individual's perceived need for dialog, be it further or any. But they do not affect that person's relative preferences for #1-#3. At all.

Thus #4 and #5 are said to be orthogonal to #1-#3. To build an analytical dataset, we separate out the individual's opinions about the need for dialog from his/her opinions about S/GIC (and thanks to their orthogonality, without altering the relative rankings of either subset in any way).

Now, the resulting data is going to be harder to read if we don't take a moment to re-rank the votes. Re-ranking also does not affect the data in anyway. Watch. Take two example vectors:

```
12345
14532
```

When we separate out the editorial opinions about dialog, we are left with:

```
123
145
```

In case it isn't obvious at first glance, these two votes express identical rankings of the three technical choices (that is: MUST not/SHOULD/not/-MAY). Because these data are ordinal, not interval, re-ranking the data doesn't alter the results of any appropriate analytical procedure. So our vectors now read:

```
123
123
```

A total of 483 GR votes were cast, but not all of those votes are analytically usable.

Examples of unusable votes include:

- Votes that "ranked" multiple technical choices identically. (A "vote" for everything is the same as a "vote" for nothing.)

- Votes that failed to assign an explicit rank to each of the three technical choices (rendering ordinal analysis impossible).

NOTE: Although some might argue that "no vote" is basically the same as "low vote," replicable analysis is not and cannot be based on individual claims of clairvoyance. If the person actually casting the vote can't be bothered to express a simple, clear preference among three distinct choices, then there's really no point in trying to guess what s/he "really" meant, much less in trying to compare, for instance, a 42413 vote to a -2213 vote.

In all, some 125 of the GR votes (26% of the total) were unusable under these criteria.

The easiest way to think about the re-ranked data is as a series of three-digit codes that tells us the individual voter's preference among the three technical options.

Each digit of the 3-digit code represents the individual's preference among the three alternatives. The first digit position is the relative ranking of "MUST not," the second digit is the relative ranking of "SHOULD not," and the third digit is the ranking for "MAY." So a code of 123 would correspond to an extreme contra-S/GIC position (MUST not/SHOULD not/MAY), and a code of 321 would represent the opposite pole (MAY/SHOULD not/MUST not).

Aside: In case it's not obvious, we've just seamlessly deobfuscated the data by reducing what was once a huge 5-D matrix down to a handful of discrete points. Dimensionality reduction is a fascinating topic for anyone who's actually enjoying reading this.

Okay, so there are six possible "states" for each vote, but some of those "states" seem nonsensical or contradictory on their face. For example, a vote of either 132 (MUST not/MAY/SHOULD not) or 231 (MAY/MUST not/SHOULD not) would seem to be best explained by a typo. Happily, these contradictions occur in only 8 votes (6 for 132 and 2 for 231); because they are so few in number, the 8 affected data points may be included in or excluded from the analysis without affecting the ultimate results. (For simplicity's sake, this analysis excludes these 8 points.)

Ultimately, the 350 votes in the analytical dataset can be represented using the remaining four points. Let's have a look:

Code	Vote	Count	Total	Percent
123	(MUST not/SHOULD not/MAY)	94		
213	(SHOULD not/MUST not/MAY)	42	136	39%
321	(MAY/SHOULD not/MUST not)	144	144	41%
312	(SHOULD not/MAY/MUST not)	70	70	20%
		Grand Total	350	

Figure 20: Summary of Debian's GR resolution votes dataset by dasein

As I've grown tired of repeating, the two "extremes" (the ??3s and the ??1s) are easily visible in the data, and they are nearly identical in size. They serve mostly just to cancel each other out. As is often the case in political

elections, it's the folks in the middle who end up making the decision.

In this instance, those "moderates" are the 312s: folks who think that S/GIC is a Genuinely Bad Idea, but who simultaneously think that a(ny) blanket prohibition is also a Genuinely Bad Idea. (In my current mood, I do not concede their point, not even grudgingly.)

But here's the thing and there's just no getting around it: the 312s chose SHOULD NOT as their first choice. They pointedly and explicitly say that S/GIC is a bad idea, their only quibble is over Just How Bad.

To recap the quote from above: "...a requirement for a non-default init system will mean the software will be unusable for most Debian users and should normally be avoided."

"This a bad idea and should be avoided in all but the most exceptional circumstances" is in no way a pro-S/GIC stance, nor is it even slightly equivocal on the question. And any attempt to suggest otherwise is nothing less than an egregious attempt to distort/misrepresent the historical facts. And that's exactly what this thread exists to combat.

On technical merits, S/GIC lost, 206-to-144. Really. Which is precisely why "we don't even need to talk about this" was pure cowardice. QE-friggin-D

Thus endeth the lesson.[16]

This fascinating and critical analysis of a GR vote whose importance stands hardly disputed in Debian's history is not only making justice to what ultimately appears being a majority of developers in Debian, but also renders a fascinating account on how algorithms can be interpreted far beyond the "neutrality" aura results are often painted in.

5.7 The aftermath of systemd in Debian

As a follow up to all these events, Debian has witnessed a dramatic loss of interest from some crucial developers.

```
To: debian-ctte@lists.debian.org
Subject: Resignation
From: Ian Jackson
Date: Wed, 19 Nov 2014 13:08:29 +0000
```

[16] last forum post by author dasein, retreived from http://forums.debian.net/viewtopic.php?f=20&t=120652&p=576562 on 10 July 2016

```
Message-id:
   <21612.38477.245453.248600@chiark.greenend.org.uk>
I am resigning from the Technical Committee with
   immediate effect.
```

> While it is important that the views of the 30-40% of the project who agree with me should continue to be represented on the TC, I myself am clearly too controversial a figure at this point to do so. I should step aside to try to reduce the extent to which conversations about the project's governance are personalised.
>
> And, speaking personally, I am exhausted.
>
> The majority of the project have voted to say that it was wrong of me to bring this GR at this time. Despite everything that's happened, I respectfully disagree. I hope that the next time a controversial issue arises, someone will step forward to advance what might be a minority view.
>
> Thanks to everyone who has served with me on the TC. I wish those who remain on the TC the best for the future and I hope that you'll find colleagues who are as good to work with as you have been to me.
>
> I now hope to spend more of my free software time doing programming. dgit is at the top of my Debian queue, but some of my GNU and SGO projects could do with attention too.
>
> Thanks, Ian.

Not just Ian Jackson, third Debian Project Leader in 1998, filed his resignation, but also Joey Hess, after 18 years of active core participation in the project:

```
To: debian-devel@lists.debian.org
Subject: so long and thanks for all the fish
From: Joey Hess <joeyh@debian.org>
Date: Fri, 7 Nov 2014 17:04:10 -0400
```

> It's become abundantly clear that this is no longer the project I originally joined in 1996. We've made some good things, and I wish everyone well, but I'm out.
>
> Note that this also constitutes an orphaning as upstream of debhelper, alien, dpkg-repack, and debmirror.

> I will be making final orphaning uploads of other packages that are not team maintained, over the next couple of days, as bandwidth allows.

> If I have one regret from my 18 years in Debian, it's that when the Debian constitution was originally proposed, despite seeing it as dubious, I neglected to speak out against it. It's clear to me now that it's a toxic document, that has slowly but surely led Debian in very unhealthy directions.

Hess has always been a mature and useful voice in the project and also in this occasion could assess well the problems he felt beyond the main controversy of a change of the init system. Of his resignation is worth nothing what he mentions in a blog-post follow up:

> Debian used to be a lot better at that than it is now. This seems to have less to do with the size of the project, and more to do with the project having aged, ossified, and become comfortable with increasing layers of complexity around how it makes decisions. To the point that I no longer feel I can understand the decision-making process at all ...

Hess kept far from polarizing the discussion around systemd, in fact resigning any anxiety about it as he and some others believe it is possible to opt-out of it later on. What he clearly suggests is that maintaining a set of socialised procedures, a cluster of algorithms of growing complexity, has a significant taxing effect on the possibility to innovate the system. The trade-off is clear, yet here the question we should be posing ourselves is not how to make algorithms more efficient, but how to make them appropriated by the vast majority of a community, so that they can be shared and redistributed further. On these regards Hess may be proven wrong: there will be hardly a way out of the systemd init change as inner processes get more entangled and less transparent for the Debian community.

Another important developer resignation is reported here below: Roger Leigh, whose efforts to maintain sysvinit through the years and this last message have provided great motivation for the fork to happen and for Devuan to be born:

```
Date: Thu, 27 Nov 2014 13:40:10 +0000
From: Roger Leigh <rleigh@codelibre.net>
To: VUA@debianfork.org
Subject: Debian fork and systemd
Received: from nagini.codelibre.net
   (nagini.codelibre.net [80.68.93.164])
Message-ID: <20141127134010.GF1657@codelibre.net>
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
User-Agent: Mutt/1.5.21 (2010-09-15)
Content-Length: 2760
```

```
Lines: 54
```

Hi,

I'm a Debian developer, currently quite disillusioned with what's been going on with Debian over the last two years. I'd certainly be interested in getting involved with a fork.

If systemd had just been an interchangeable init system it wouldn't be so problematic. It's the scope creep and mess of poorly-defined interdependencies that are truly shocking. Take logind, for example. When looking at how to implement XDG_RUNTIME_DIR for non-systemd inits, I couldn't find any actual specification for how to do this. That's because there isn't one, just some loosely-worded descriptions; it only exists in the systemd implementation. And the semantics of it are very poor indeed; it hasn't been developed with safety, security or flexibility in mind. We'll come to regret adopting this since the poor design decisions are likely to become entrenched.

And more recently, there have been several reports of unbootable systems. That's unconscionable, and a serious break with Debian's traditionally solid support for backward compatibility. Here, existing supported systems have had that support dropped on the floor. With sysvinit great effort was taken never to break existing configurations, and that appears to have been lost. Introducing dependency-based boot took over two stable cycles; optional in one, default in the next, mandatory after that. That could have been reduced certainly, but the point is that time was taken to ensure its correctness and robustness (and in the beginning, it did need work, so the wait was worthwhile). This has not occurred with systemd, which has been made the default yet is still not ready for production use.

Debian is developed by hundreds of active developers and used by many times more people. People rely on Debian for their jobs and businesses, their research and their hobbies. It's not a playground for such radical experimentation. systemd support was forced in rapidly and didn't just cause breakage, it caused breakage with our own past, breaking the reliable upgrades which Debian has been renowned for. Personally, I'd like to see a much higher regard for stability and backward compatibility, rather than just ripping out the old in place of the new without any regard for its true value. It might not be bleeding edge, but we already have Fedora for people who value this over a solid and dependable system. It's possible to be up-to-date without being a Fedora; Debian unstable historically made a good job of this.

Kind regards, Roger

```
--
  .'''.  Roger Leigh
 : :' :  Debian GNU/Linux
   http://people.debian.org/~rleigh/
 `. `'   schroot and sbuild
   http://alioth.debian.org/projects/buildd-tools
   `-    GPG Public Key      F33D 281D 470A B443 6756
     147C 07B3 C8BC 4083 E800
```

5.8 The Debianfork declaration

The "Debianfork" declaration was published during the period of the GR vote in November 2014 by an anonymous group calling itself "Veteran Unix Admins", referring to a seminal article by Paul Venezia defining the traits of such a professional figure[17]. In this declaration, reported below, there was a call to fork Debian over the diatribe, constituting an attractor for all those whose concerns were liquidated by the GR Vote.

As some people found out, I wrote this declaration myself after having followed closely most debates, strong of the support and critical review of a larger private group of sysadmin and programmers (more than a thousand subscribers) which is indeed called VUA. Here below the text:

5.8.0.1 Who are you?!

We are **Veteran Unix Admins** and we are concerned about what is happening to Debian GNU/Linux to the point we decided to fork the project.

5.8.0.2 And why would you do that?

Some of us are upstream developers, some professional sysadmins: we are all concerned peers interacting with Debian and derivatives on a daily basis.

We don't want to be forced to use systemd in substitution to the traditional UNIX sysvinit init, because systemd betrays the UNIX philosophy.

We contemplate adopting more recent alternatives to sysvinit, but not those undermining the basic design principles of "do one thing and do it well" with a complex collection of dozens of tightly coupled binaries and opaque logs.

5.8.0.3 Are there better solutions than forking?

[17] "Nine traits of the veteran Unix admin" published on 14 Feb. 2011 on Infoworld.com http://www.infoworld.com/article/2623488/unix/nine-traits-of-the-veteran-unix-admin.html

At the moment no, unfortunately.

The default Init system in the next Debian v.8 “Jessie” release will be systemd, bringing along a deep web of dependencies.

We need to individuate those dependencies, clean them from all packages affected and provide an alternative repository where to get them. The stability of our fork is the main priority in this phase.

5.8.0.4 Why is this happening in your opinion?

The current leadership of the project is heavily influenced by GNOME developers and too much inclined to consider desktop needs as crucial to the project, despite the fact that the majority of Debian users are tech-savvy system administrators.

Moreover Debian today is haunted by the tendency to betray its own mandate, a base principle of the Free Software movement: put the user’s rights first. What is happening now instead is that through a so called “do-ocracy” developers and package maintainers are imposing their choices on users.

5.8.0.5 Can you articulate your critique to systemd?

To paraphrase Eric S. Raymond on the issue, we see systemd being very prone to mission creep and bloat and likely to turn into a nasty hairball over the longer term.

We like controlling the startup of the system with shell scripts that are readable, because readability grants a certain level of power and consciousness for those among us who are literate, and we believe that centralizing control services, sockets, devices, mounts, etc., all within one daemon is a slap in the face of the UNIX philosophy.

A timely reply by some people willing to use systemd is visible at forkfedora.org. This page is useful to highlight a fundamental difference: systemd may simplify the task of configuring init, but it does so by enforcing an increasingly opaque approach to the init procedure. In systemd sure it appears easier: one can tweak a few variables and have all the rest handled by a big binary system that is way bigger than sysvinit.

```
  ls -lH /sbin/init
  sysvinit: -rwxr-xr-x 1 root root 36992 Jul 14  2013
     /sbin/init
  systemd: -rwxr-xr-x 1 root root 1317632 Sep  1 14:41
     /sbin/init
# You may not be veteran enough, but you’re already
   pretty fat.
```

It can be said that the security model of systemd relies much more on developers and package maintainers and much less on system administrators. As Debian users we are simply asking not to be forced into this model and, taking into account the CTTE vote on the issue was almost draw, we believe its execution should be way

more attentive in listening to what many users are asking: Init Freedom! and not being entangled by systemd and its omnipresent accessories.

5.8.0.6 How long are your beards?

This is not a beard contest, rest assured the furry ones among us are not sheep.

5.8.0.7 To sum it up?

We are forking the Debian project to create a new base distribution that promotes Init Freedom.

This will take time and we will proceed step by step.

First of all and concurrently with Debian 8 "Jessie" release, we aim at having a complete solution to which current Debian users can dist-upgrade smoothly.

If you need this too, then please help: with a donation or getting involved.

5.8.0.8 We need to talk.

Sure, write an email to VUA@debianfork.org.

Following our call some people gathered on **IRC Freenode channel #debian-fork**. Be welcome.

The mailinglist is open to subscriptions. Break the ice if you feel like, take care to do it for a reason.

5.8.0.9 Are you guys alone in this?

Not at all, there are more protests against the imposition of systemd on users.

This article is a good introduction to the issue at hand: Systemd: Harbinger of the Linux apocalypse.

There is the boycott systemd website providing several references.

Then there is the "systemd fork" called uselessd with some good points and lots of lulz.

An exit strategy is being elaborated at The World After Systemd

The wikipedia page lists also some critiques in its systemd reception section.

5.8.0.10 Thanks for doing this. How can I help?

A small core group of Veteran Unix Admins is actively producing a *proof of concept* for the fork and setting up some infrastructure that can be used for its development.

The response to this declaration, published on the website debianfork.org, was overwhelming. The IRC channel, ran by volunteers we have never met, grew to an approximated amount of 500 regular participants in a few days from the vote. Headlines

on Slashdot, The Register UK, Heise.de, YCombinator, Linux Journal, Opennet.RU and ZDNet.FR ran stories with hundreds of comments attached. To summarise, an archive of the press coverage is maintained at the address: devuan.org/os/press/in-the-news

As the project opened for donations to be accepted to support the declared plan, without any marketing campaign and only during the first year it received approximately 10.000 EUR. To many of us who were involved into the Debianfork declaration it became clear: we needed to follow up with our plans. I was extremely lucky to not be left alone on the lead of such a project and immediately aided by Franco "Nextime" Lanza who took on him more than half of the work needed in the early stages of development.

Through work done in 2015 a new independent distribution was forked from Debian. After a quick consultation with the growing community we gathered, we decided to call this new distribution "Devuan", pronounced "Dev One" in english, since it recalled the name of Debian as well the VUA acronym in it.

5.9 General response to Devuan

Devuan coalesced the participation of a vast portion of people concerned in the course of 2 years to build a new GNU+Linux base operating system project called Devuan. The Devuan GNU+Linux project is not only of enormous interest for my thesis, but ended up leveraging my professional activity as a developer and the future of our research organization Dyne.org.

The overwhelming quantity of reactions that this event have generated is far beyond the possibility to gather a full account of it in this thesis. Nevertheless in appendix to this thesis is provided a qualitative selection of messages received in support of the effort to fork. These messages are particularly interesting since they take the time to explain, in some cases from a user perspective and in some other cases from a system administrator perspective, why the adoption of systemd was problematic, revealing a diversity of aspects that were easily missed or dismissed by systemd supporters.

Beyond most theoretical formulations, it is reasonable to think that the Devuan project gives a poignant picture of what can be perceived as "sovereignty" by software practitioners and more specifically what "UNIX principles" and "UNIX philosophy" really mean to many people as they are often used in ICT and security contexts.

5.10 Devuan is born

We decided the first focus with Devuan was not that of releasing an operating system for general use, not even a base system to build derivatives. It was that of building

a “continuous integration infrastructure” (CI) that could streamline the builds of packages with our modifications and overlap them to the existing unmodified packages in Debian.

This architecture has later revealed to be the best possible foundation for Devuan, leading to a modernisation of Debian’s infrastructure and to the creation of a few new software patches and applications for the building and distribution of software package. This approach also helped to establish a trustworthy process of deployement of the algorithms into the actual platform distributed and accessed by participants, while completely appropriating (and in some cases rewriting) the production means necessary to produce the distribution.

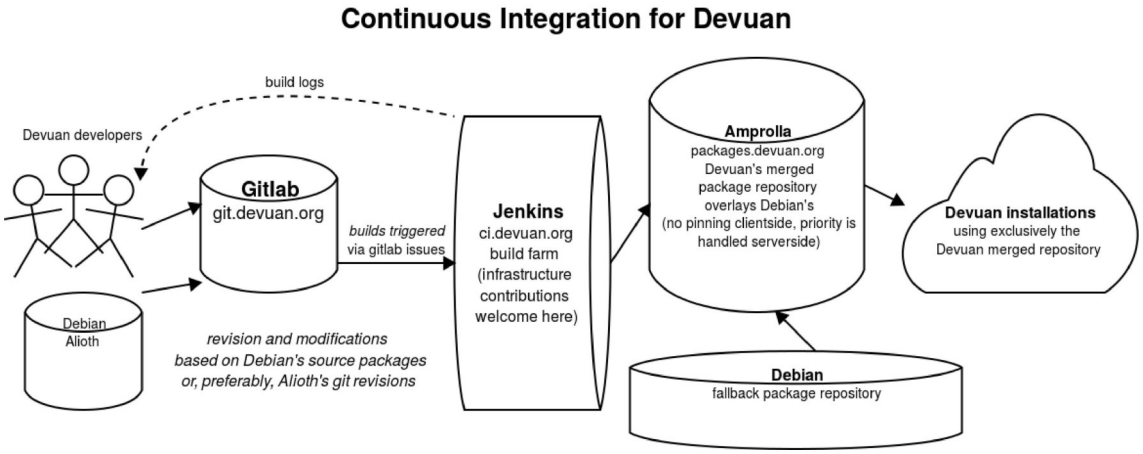

Figure 21: A graphical representation of the Continous Integration infrastructure planned for Devuan

As for the rest of the distribution, the rationale for such an approach has been that of minimalism. Assuming Debian as a trustworthy base at least until its version 7 release cycle, we needed to carefully document every single change we made to it. Our work then demonstrated how is well possible to do that with modern tools as Git and Jenkins and with an ad-hoc system for packaging which we called Amprolla.

The development continued touching the milestones of some alpha and beta releases until a stable release in mid 2017, only after the community behind Devuan considered the current beta to be stable enough to even use it in production. It is important to consider that the early focus on a solid CI for Devuan has also meant it took the role of a base distribution open to customization (appropriation) and that overall adapted to the sort of circular economy described in the first chapter of this thesis. With that in mind, it is easy to imagine how Devuan became a base distribution for other creations already in its early stage with up to 12 derivative operating systems at the time of writing, so called downstreams developments. The current development goes in the direction of curating the necessary changes in existing packages, welcoming a growing number of developers and establishing good practices for decision making, consolidating the on-line package availability under

heavy infrastructural load and developing from scratch a “Simple Development Kit” which allows people to build ad-hoc versions of their own operating system based on a very minimal base of Devuan. One of the first results of this possibility was the adoption of the system by co-developer Enzo “Katolaz” Nicosia (Queen Mary Univ. UK) to build a “Devuan Minimal Live” system which became extremely useful for the blind GNU/Linux user community for its simplicity and adherence to text interfaces.

In 2017 the Devuan GNU+Linux community is thriving and keeps growing, the distribution is listen in the top30 worldwide by Distrowatch, donations continue flowing in and development moves steadily towards a second stable release, while experimentations for future improvements (also to the init system) proceed at a slow pace and in good contact with the community. The chat channels have an approximate amount of 500 participants (#devuan and #debianfork on freenode.net) and the mailinglists count 1128 subscribers at the time of writing, with more than 500 daily unique visits on the websites and worldwide mirrors offered by multi-national ICT companies like Leaseweb, but also educational institutions as University Warwick and the Mathematics dept. in Princeton and historical cultural associations as NLUUG.

5.11 Conclusion

When I started studying the systemd phenomenon in 2013 I suspected it would be relevant for my Ph.D thesis, but I had not imagined it would have resolved to become an entirely new distribution. This project has an enormous positive impact not only on my part-time Ph.D research position, but also on my professional life, as it grew to consitute a base for many more projects and made Dyne.org double its size in terms of participation and donation budget. Arguably, the base for this success is not technical, but related to the very subject of algorithmic sovereignty.

What we came to call the systemd controversy revealed to be an extremely relevant issue in the software freedom movement, still unfolding its narrative. Software freedom is defined by the “four freedoms” devised by Richard Stallman and the Free Software Foundation. Two individual freedoms relate to *appropriation* (to use and to study how a program works, by reading its source code) and are completed by two collective freedoms related to *sharing*, *creation* an *distribution* (to share and modify the program’s source code) that enable programmers in a community to create software on the behalf of others. This is a fundamental stance enabling social emancipation and appropriation of the technology.

The Debian Free Software Development Guidelines (Debian FSDG) was a foundational ethical document that grounded Debian as a promoter of software freedom, and inspired the Open Source Definition in 1997 (adapted by its primary author, and then Debian Project Leader: Bruce Perens). Although the four freedoms and

the GNU legacy are not mentioned in either of these documents, they form a solid base for people to understand the meaning of free software: “free software grants you freedom”.

Arguably, since systemd itself is free software, the four freedoms are a necessary but not sufficient condition to protect the interest of free software users. The four freedoms pose the question of who can participate, since programming requires special skills: if you don’t have these skills yourself, and your community doesn’t have these skills, you’re automatically excluded from controlling your own computing. With more software, lines of code and complexity, even programmers are not always able to interpret, understand, or modify software. The “avalanche” of systemd is a situation in which a new complex system is developed top-down and within a short amount of time and at the risk of security implications not being throughly verified. Such a complex system, characteristically based on CRUD[18] interaction, is immediately deployed to substitute a shared literature of algorithms in place since decades, a literature that through all those years has been *appropriated* and *shared* by communities, *distributed* also in professional and mission critical environments and on which several *creations* were based.

The systemd controversy illustrates how expectations can deceive the mind and lead to diverging, even antagonistic perspectives given the same facts. If younger developers who grew up with proprietary operating systems are mostly satisfied with a consistent set of interfaces to ease their work, or even abide to open source principles without conceiving the importance of democratic governance and diversity of needs and visions, people with experience of more UNIX-inspired systems and various deployments in largely different settings (from small companies with heterogeneous systems to large professional data centers) are more focused on portability and long-term focus on maintenance, continuity, and backwards compatibility that are definitely broken with a switch to systemd. As demonstrated by years of diatribes it is almost impossible to reconcile the two sides without a large effort to bridge the gap, that translates into a quantity of work that nobody wants to take up: on the one hand, systemd supporters want their way exactly to avoid having to rely on artesanal contextualization of their work, while systemd detractors prefer theirs to avoid having to rely on a large set of programs that require a completely alien mindset to cope with and a different governance on algorithms that completely substitutes the one in place (or arguably not in place) for decades.

Comparatively, the git program to manage source code, which shares some characteristics with systemd (a large set of very different binaries covering a large and extensible set of loosely related functionality, new interfaces from what was existing

[18]CRUD stands for “create, read, update and delete” and it indicates an interaction pattern in computer systems which allows such actions to be operated over variables in a dataset, implying the internal processes are hidden and in any case their design is not part of the interaction.

at the time of its introduction, and complex internals) didn't meet much resistance to become the most-used program in its field. This suggests that free software hackers and users (programmers, system administrators, advanced users) are willing to take up new systems as long as they choose to do so on their own terms. This is confirmed by the GR resolution vote where the option to create a hard dependency on a specific init system was marginal.

Well beyond the project and the community themselves, the Devuan experience provides new solid grounds for theoretical assumption of what algorithmic sovereignty means and what can be its potential. While this experience relates mostly to a sector of specialised professionals, is well imaginable that in a close future of pervasive computing scenarios analog situations will repeat.

After two years of working on Devuan, most decisions were taken by a handful of loosely-coordinated people who presented their results to the community. When attempting to take decisions affecting directly the community and without consultation (e.g., for the choice of a forum infrastructure that heavily differs from Debian's) there was a huge drawback on these choices, forcing resignation from roles and change of plans which were rather quickly taken up by new volunteers.

Radical decisions that affect the future of the project can be made as long as participants are given sufficient information and time to ponder it, already have positive experience of previous decisions by identified leaders, have a way to opt-in the change, or if the change is not perceived as disruptive of one's workflow. An initial effort can be made to adapt if the long-term effect is perceived as favorable.

On the level of algorithmic design, the lesson to be learned here is the difference between closed declarative designs and open scriptable ones. The difference is clear when considering an episode which was omitted so far. At the time of the debian-fork.org declaration there was a response given by the opposite camp, presumably anonymous developers of the systemd software or sympathizers, called forkfedora.org. On their website they have shown two different source code examples to start the popular UNIX mail daemon sendmail, one using sysvinit and the other using systemd, side by side. The main argument used against sysvinit was the length and simplicity of the source for the systemd example which indeed was improved. See the section "Sysvinit scripts vs. systemd service files" in Appendix for comparison.

More considerations on this project are left in the conclusion of this thesis, in particular about the role "forks" have in the context of algorithmic sovereignty.