Rapport d'avancement et planification de fin de thèse

Comité de suivi de 3ème année de thèse de Edlira NANO[1], sous la direction d'Aurélien Tabard[1].

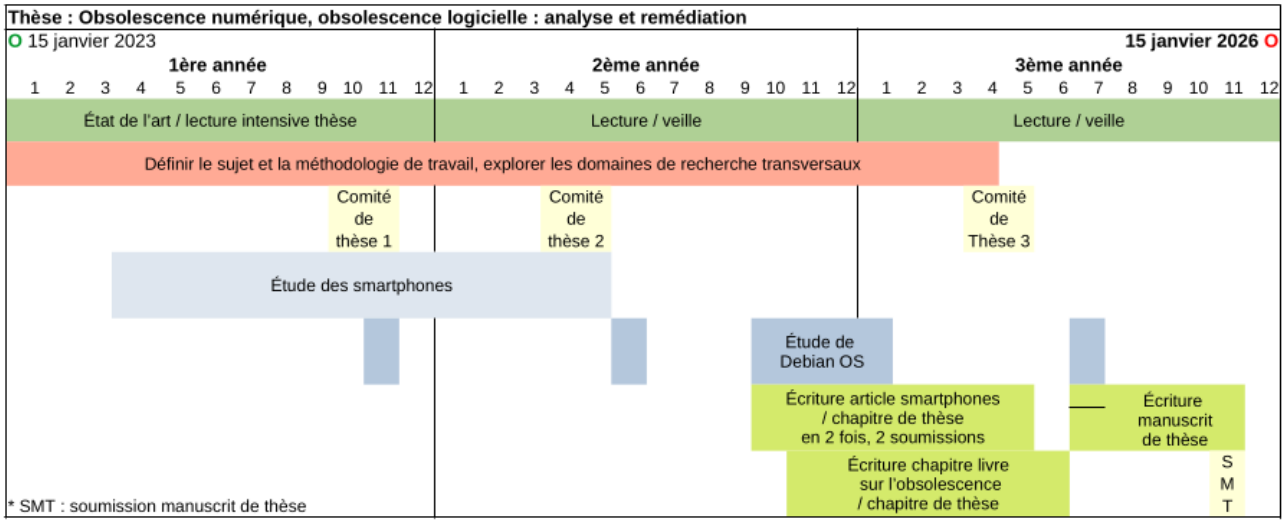
Membres du comité : Stéphane Crozat (extérieur) et Haytham El-Ghazal (interne)

Thesis title: Digital obsolescence, software obsolescence: analysis and remediation strategies

Abstract: Digital systems have a growing ecological impact and environmental footprint. Digital obsolescence is one of the factors driving this rapid increase. This PhD thesis aims to analyze software obsolescence, its role in digital device obsolescence and existing remediation strategies that fight, avoid or bypass it. The study will first focus on smartphones, whose rapid pace of development and replacement could allow the characterization of digital obsolescence. The second object of our study is maintenance and long term support within Debian, a widespread operating system based on the Linux kernel, maintained by a community organized as a non-profit organization, following the principles of open-source code, collaboration, free distribution and sharing for everyone.

1. Research context and work planning

This PhD thesis began the 15th of January 2023 at Université Lyon 1 in France, LIRIS laboratory, department of Computer Science, under the supervision of Aurélien Tabard, within the SICAL team, and the Limites numériques project, a research project focusing on the environmental footprint of digital technology. It will end the 15th of January 2026, after three years.



I have now completed two years and 4 months of the PhD thesis. I wrote and submitted my first article about my Phd work. I am also finalizing an academic book chapter on Digital Obsolescence and Software Obsolescence, also work undertaken during my PhD. I will integrate this work as chapters of my final manuscript. During the remaining 7,5 months of my thesis, I will focus on the writing of the remaining chapters for the PhD manuscript, the submission and the revision process.

2. Work carried out so far

1 Univ Lyon, UCBL, CNRS, INSA Lyon, Centrale Lyon, Univ Lyon 2, LIRIS, UMR5205, F-69622, Villeurbanne, France.

During the first two years :

- I explored my research domain and defined precisely my thesis subject and work plan.
- I performed an in-depth study and analysis of two ecosystems: the smartphone Android OS and Debian OS ecosystems.
- I wrote a scientific article on the Android OS ecosystem and submitted it to ACM CSCW 2026 (Conference on Computer-Supported Cooperative Work and Social Computing) :
- I am finalizing an academic book chapter on digital obsolescence in the context of the seminar Capitalisme numérique, directed by Olivier Alexandre at Centre Internet et Société, CNRS.

This second year of my PhD was mostly taken by analysing the results of data taken during my interviews and field studies, and writing articles to present and discuss them. A small part was also taken to gather some more data and interviews for the Debian OS and alternative OS study.

2.1 Analysis of the lack of maintenance and updates in the Android OS ecosystem

Research hypothesis

Software development and maintenance plays an important role in smartphone obsolescence. By studying the software involved in Android OS smartphones, their development and update process, we will be able to better understand why the Android OS market of smartphones presents a low rate of software maintenance yielding to a high rate of device renewal. We will also search and observe possible successful strategies to overcome these problems.

Research questions

1. How is Android structured? Who are the actors involved, and what is the process for building Android?
2. What inhibits Android updates? Where does obsolescence manifest itself?
3. What are the strategies of the actors in the Android or non Android smartphone ecosystems in tackling maintenance issues?

Research methodology

Selected interviews: I performed 10 interviews with key actors in the Android OS ecosystem: a developer at Google, Fairphone's leader of software longevity, the main developer of the alternative Android OS Lineage OS for microG, a developer at Debian working on porting the Linux kernel and Debian in a highly integrated System on a Chip (SoC) architecture, similar to the SoCs found in smartphones, developers of two alternative smartphone OSes : Replicant and Mobian.

Conference ethnography and field study: I participated in person or followed numerous online conferences on the technical aspects of Android OS related development and update process, and alternative OS systems for smartphones. These were mainly: the annual Linux plumber conference, the annual Linux kernel recipes conference, the Open Firmware conference, two Capitole du Libre events, the Free Silicon conference of free and open source design and manufacturing of chips, the Debian OS annual conference and a European Debian OS community gathering where the Mobian OS community was also present.

Technical exploration of documentation and specialized media: In particular I explored official technical Android developer documentation by Google, technical documentation on alternative Android OS based systems such as LineageOS, LineageOS for microG, e/OS, divestOS, grapheneOS and on alternative non-Android mobile OSes based on Linux, such as postmarketOS and Mobian. Developer community forums and mailing lists were also of great help (xda-developers website and forums for example). I also followed specialized news and analysis media such as lwn.net (a Linux news and information website), 9to5google.com news from Google website, and selected articles from Arstechnica, TheVerge and Wired magazines.

Workshops and user experience

During the PhD I organized two workshops with users on smartphones and took profit of experience from the user workshops on smartphones organized by other members of my team.

Results

First, we have been able to develop an analysis of the different software layers and actors involved in the development of Android OS as shown in Fig. 1.

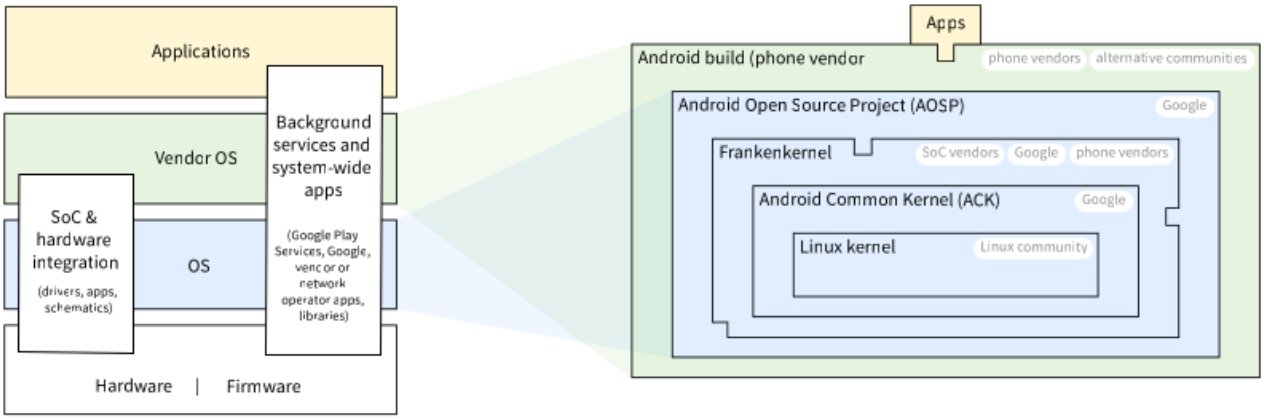


Figure 1: Android software layers (left) and Zoom into the Android OS composition (right)

These software layers and actors involved in developing each of them, allow us to better understand the development process of Android.

In this development process, we first understood that the Android OS is not one, but many : every phone has its own specific Android OS build. **The Android OS is made of Android specific builds for each phone device.**

Understanding this was the key point of our in depth analysis of what is known to be the **fragmentation problem of Android : many devices with many OS systems that do net get updated.** We explain how, at the kernel level, the Android fragmentation problem is a problem of creating **frankenkernels, altered Linux kernels holding code for Android and for the specific System on Chip component used in the phone.** These SoC frankenkernels are not being maintained and updated by the SoC manufacturers. They are not following development process that could ease their maintenance either: code is added in very large unreadable batches, it is not documented, schematics or documentation for SoC behavior are not publicly available and are not standard, and the code is not following the upstream and mainline coding practices of the Linux kernel community, which could help with maintaining from the open-source community.

The ways in which code is put in common, kept private, or shared in large batches plays an important role in enabling or inhibiting updates. Figure 2 shows the development flow of an Android build: the red crosses indicate a lack of contribution to the original code-base.

At the vendor level, where each vendor adds its drivers, system wide libraries and applications to Android, updates are not taking place for longer than one or two years, before the maintenance of the phone is stopped, without notice, while Google continues to publish new versions of the Android OS yearly. By not performing these updates, vendors abandon their older devices, creating problems of aging devices for customers, or security issues that yield to device renewal.

At the Google level, updating is a highly strategic procedure: while Google develops and maintains Android, AOSP, the Android Open Source Project at the core of Android, is not an OS that can be installed on phones, as we said before. Android is built in such way that every phone needs a specific Android build in order to work. Google has created a safe space for vendors to do such builds, and has isolated these in Android where vendors can insert hardware specific

frankenkernels, drivers and not open or not transparent code into it, making the updating of builds rely mostly on the vendor's good will to update.

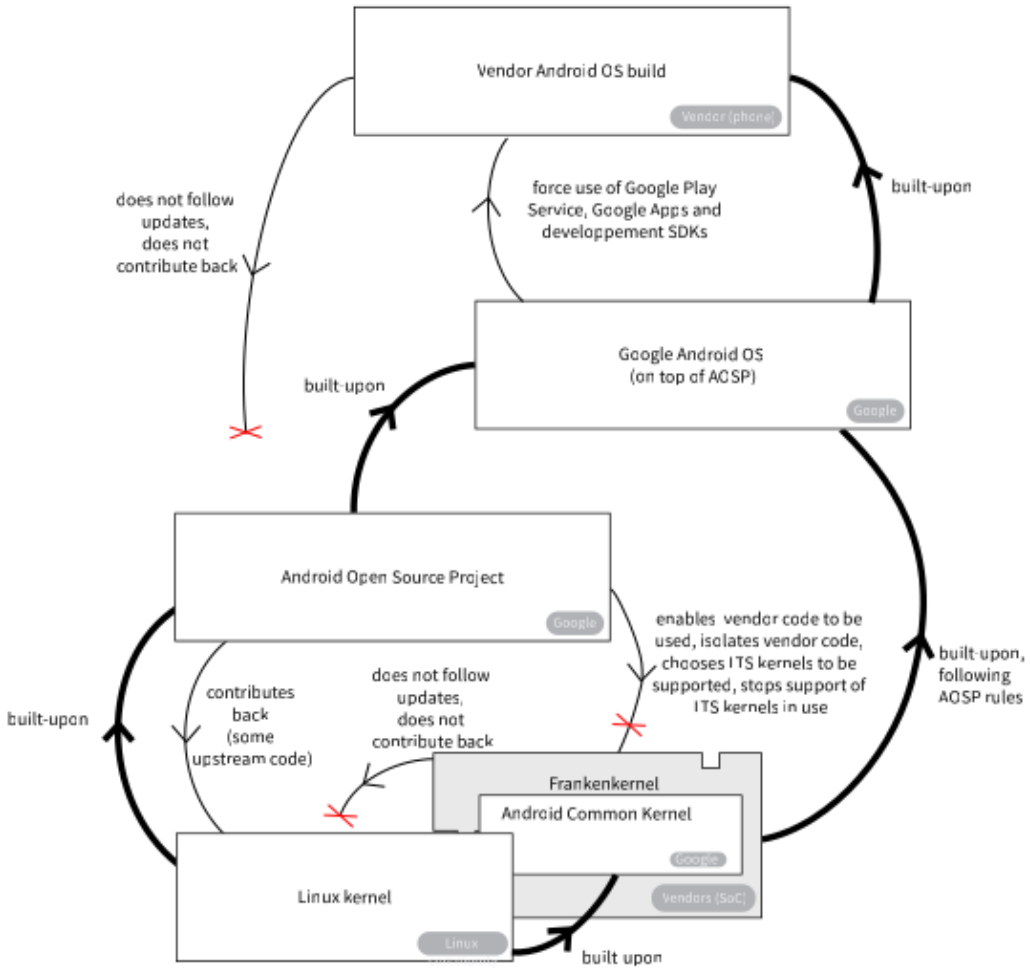


Figure 2: The development flow of an Android build: the red crosses indicate a lack of contribution to the original code-base.

This isolation helps Google to update the rest of the system. Google develops and updates core parts of AOSP, but also puts a particular effort in updating and inserting new features in it's own proprietary services and apps, such as Google Play Services, Google Search, Chrome, Play store etc. These Google services and apps have become more important for the system during the years, removing them would make the system miss key features. Furthermore, Google has set up a system of contracts with vendors in order for them to always include Google services and apps in their Android phones, enabling Google to maintain its business model. Several countries, the European Commission, the United States, UK, India and Japan to cite only some of them, have filed complaints and ruled against Google for this dominant position and power in the Android ecosystem. By analyzing some of the investigations and reports in this cases, we have also understood how the ecosystem of vendors and Google forges relationships and leaves little to no place for alternative solutions.

Alternative phone OSes build over AOSP to try to maintain Android OS for longer times, from 5 to sometimes up to ten years after the phone release in the case of Lineage OS. They do so by reverse-engineering and hacking the manufacturer OS code, and in particular the proprietary code of the drivers and the firmware. Depending on devices, the darkness of the proprietary code, or the success of reverse-engineering, this is a highly difficult task, and we can see that these alternative OSes only exist for some devices, not all of them, and not the majority of them. When they succeed the resulting Android OS still carries most of the frankenkernel that the SoC manufacturer

offered, but with some updates that they succeeded to do, some concerning security issues at the Linux kernel level, and other are updates coming from the AOSP level, in order for the new AOSP to work on the frankenkernel sometimes small updates are performed on the latter. At some point the Linux kernel that this frankenkernel holds will stop being supported by Google AOSP, which means that newer versions of AOSP will not anymore be available at all for the device. This happened for examples in January 2011 for Samsung S5 devices (released in 2014). This means that development of Android OS for this device is completely halted, including the development of alternative Android OSes such as LineageOs and LineageOs for microG that had been maintaining this device for 10 years. The only possibility for this device to stay updated is then to install an alternative non-Android OS such as postmarketOS (thus the name “post the market” OS) or Mobian. These non Android OSes follow the development of the mainline Linux kernel, they are also called mainline Linux OSes. While they have the advantage of always being up-to-date at the kernel level, these OSes have trouble with the driver and vendor specific proprietary code, whose development is based on reverse-engineering and hacking, and thus face greater difficulties in making the phone components work well. As a result, even fewer devices are fully functional for these OSes but the community continues efforts in implementing more functionalities and devices.

2.2 Debian OS: maintenance at the core of the system

Research hypothesis

Debian is a 30 years old OS, the second oldest Linux-based OS, used on a large number of servers, embedded devices, super calculators, on the largest number of system architectures compared to any other OS, and that is also at the core of many other derived Linux distributions (Ubuntu, Kali, Tails, Raspbian...). Maintenance and durability efforts in the development of Debian OS are the main focus of Debian developers who are called “maintainers”. At the same time, stability is an important issue, as many systems rely on Debian. We hypothesize that Debian has built through time solid knowledge and practice in maintenance of software to deal with updating issues.

Research questions

1. How is development and maintenance work in Debian structured?
2. How is a Debian package developed and maintained ? What coding and social practices are involved in their maintenance and longevity? How is the end-of-life of a package dealt with ?
3. How is a Debian release developed and maintained? What is the Long Term Support (LTS) release and how does it tackle maintenance issues?

Research methodology

The study of Debian OS consisted in attending several week long gatherings of the community during these two past years. These gatherings offered an opportunity to immerse myself into the community, to attend team workshops, internal meetings, public conferences and to be able to conduct several interviews with selected key players of the Debian community.

Results

Part of the data gathered in the Debian community was analyzed and used in my research around Android. Indeed, interviews and conferences with the Mobian team, a smartphone OS developed inside Debian and the Linux kernel team in Debian, helped to understand problems and remediation strategies to maintenance issues already in Android.

Another part of the preliminary analysis on data gathered in the Debian community, was used to understand the Debian ecosystem, how the community is organized, how development work is organized and how collaboration process and decisions are made. This also helped me define my focus on some aspects of maintenance in Debian, and precisely frame our research questions as they are shown above. Our analysis focuses on two subjects: a Debian package, its development and maintenance process, and the Long Term Support release in Debian, what it is, and how work in it it offers longer maintenance periods for packages, but also of the whole Debian stable release.

My first analysis of these two subjects shows that maintaining a package is not just about code, it is about building a social interaction around maintenance issues on the long term between developers: the persons in Debian that maintain the Debian package, and the person outside Debian most of the time, that builds and maintain the original upstream package. These two actors / communities need to build a relationship of trust and collaboration in order for each of them to notify changes to the other whenever their part of the code changes, and decide together or separately in their communities how updates will be done in practice. These social coding relations between developers, echo the problems in Android where we noticed that code maintenance is a bi-lateral relationship between upstream code producers and derived code producers. Without this relationship, updates can be easily missing, code and devices abandoned, and difficult to be implemented by external actors.
My analysis of LTS in Debian needs to be refined and the results incorporated into the overall work. But a first glance at the data indicates that by developing an alternative economical model, with the aim of longevity, little effort is needed to offer long term maintenance support. This was true in the Android ecosystem, with the example of Fairphone building over LineageOs and PostmarketOs to offer long term support for its phones. This also seems to happen in Debian, if we look at the original alternative economical model built over LTS and how it has, with little effort and not a big team, a positive effect on the longevity of packages and of the Debian OS stable release support.

3. Future work: writing the thesis dissertation

The writing work of the thesis dissertation should begin in June and finish in November. In November I plan to submit and devote the next two months to feedback and rectifications of the manuscript, and preparation of the thesis defense. I plan on reusing my previous written articles as two or three chapters of the thesis as it is usual in our field. To complete, I need to write the introduction, a new chapter on the analysis of Debian OS, another one on a compared analysis and discussion of these two studies (Android and Debian) and finally add a conclusion and perspectives chapter. Here is a more detailed writing plan:

1. Introduction (partly written, completion needed)
   Motivations to this thesis. Research questions. Methodology of work. (to be written, parts of the research questions and methodology are written in the work for the article).
2. General thoughts on digital obsolescence and remediation strategies: state of the art, areas of interest (mostly written in the book chapter Digital Obsolescence, to be adapted)
3. Producing software obsolescence: the case of Android OS (written article)
4. Obsolescence remediation strategies in alternative smartphone OSes (to be completed, partly written in the article)
5. Debian OS: strategies of maintenance and longevity (to be written)
6. Discussion (completion needed, partly written in the book chapter and article)
7. Conclusion and perspectives (to be written)

4. Publications and pre-publications

1. Léa Mosesso, Nolwenn Maudet, Edlira Nano, Thomas Thibault, Aurélien Tabard. Obsolescence Paths: living with aging devices. *ICT4S 2023 - International Conference on Information and Communications Technology for Sustainability*, June 2023, Rennes, France. ⟨10.1109/ICT4S58814.2023.00011⟩. ⟨hal-04097867⟩
2. Edlira Nano. Digital obsolescence. *Doctoral Symposium of ICT4S 2023, The International Conference on Information and Communications Technology for Sustainability*, Jun 2023, Rennes, France. pp.36-47. ⟨hal-04107104⟩
3. Edlira Nano, Aurélien Tabard, Nolwenn Maudet, Léa Mosesso. Producing software obsolescence: the case of Android OS. **Submitted in May 2025** to ACM, CSCW 2026, conference on Computer-Supported Cooperative Work and social computing.

4. Edlira Nano, Jeanne Guien. Digital obsolescence. Chapter book for Capitalisme Numérique, under the editorial supervision of Olivier Alexandre, Centre Internet et Société, CNRS. **In preparation**. The book is to appear in October 2025.

5. Seminar and conference talks

1. **Producing software obsolescence: the case of Android OS**, talk given at Séminaire politiques environnementales du numérique, Centre Internet et Société, online, April 2024
2. **Producing software obsolescence: the case of Android OS**, talk at Séminaire UTC.lowtech, June 2024.
3. **Studying longevity in Debian OS**. NetGouv 2024. Journée annuelle du GdT Gouvernance et régulation d'Internet du Centre Internet et Sociéte, CNRS. May, 2024.
4. **Designing obsolescence: the case of smartphone SoCs**, talk at the 2024 Free Silicon Conference (FSiC) sustainability session, Paris, June 2024
5. **Producing software obsolescence: the case of Android OS**, internal seminar of the Phenix team in the Citi Lab at INSA, Lyon. March 2025.
6. **Impact territoriaux, sociaux et environnementaux des infrastructures numériques : l'exemple de Marseille**, Journée "L'empreinte environnementale du numérique en débat" Master communication numérique et conduite de projets Cemti, Paris 8, April 2025.

6. Technology transfer activities

I am a member of La Quadrature du Net, a french national non-profit organization that defends digital rights. Working there for 32 days a year in 2024 and 2025, as my doctoral contract allows, I am organizing and participating in the newly created working group studying the environmental aspects of digital technology. In this working group, I was able to conduct a study on the environmental and social impacts of ICT infrastructure in Marseille, published as a report entitled: Enquête : à Marseille comme ailleurs, l'accaparement du territoire par les infrastructures du numérique, in December 2024.

7. Teaching experience

- **Ethics in ICT and digital technologies.** Licence MIASHS, Aix-Marseille Université, Licence 3, 3 hours of main course, May 2025.
- **Digital Obsolescence.** Ecole Centrale Méditérranée, master 2 DO-IT, 8 hours of main course, October 2024.
- **Ethics in digital technologies.** Ecole Centrale Méditérranée, master 1, Semester 8, 36 hours of main and TP course. I have been teaching and co-organising this course yearly since 2020. In 2024 for the first time, I was the main coordinator. April – May 2024.

8. Doctoral training

I completed 107 hours of the doctoral formation at Ecole Doctorale InfoMaths de Lyon, out of the required 100 required: 61 hours on FSC (scientific formation) and 46 on FIP (professional insertion ones). I also followed many other seminars and talks outside this training offer. Joined is an automatically generated list of my training activities at the Ecole Doctorale InfoMaths de Lyon.