Over the last decade the mobile infrastructure has expanded rapidly in Uganda, from 126,913 in 2000 to 16.356 million in 2012. In 1999, Uganda became the first country on the continent where the number of mobile subscriptions surpassed fixed lines (Renner and Sheehan 2003: 60). The expanding mobile infrastructure has provided access to telecommunications to huge numbers of people for the first time. Mobile telephony offers new ways to communicate, and these are often quicker, cheaper and more convenient than existing infrastructures. This rapid expansion has been much hyped in the popular press of the global North, where phones have become a symbol of connection to the modern and global, and increasingly to worlds of innovation. Writing for Time magazine Perry argues that Africa’s developers are finally “joining the global marketplace… the next killer piece of code is as likely to be written in Africa as anywhere else” (2011). This kind of commentary requires willful blindness to the chasms of unequal opportunity between centers of the “hyperdeveloped world” (Suchman) such as Silicon Valley and their counterparts in the global South.

The aim of my research was to take a different perspective - that of the maintainers. I wanted to explore the less visible work of caring for the mobile handsets that had come to Kampala to serve as the last mile of this emerging infrastructure. In a wide-ranging ethnographic study of technicians in downtown Kampala, I explore the circulations of information, knowledge, parts and tools that
constitute “infrastructures of repair”. I move outwards from the canonical site of repair - the workbench - to review collaborative and competitive local networks of repair practice, online communities of repair technicians, and the global circulation of mobile phones themselves.

In this short paper / presentation, I explore the complex ecologies of tool developers that produce the repair tools used by technicians in downtown Kampala. I focus on the story of the death of a software repair tool called the J.A.F. box, which poses intriguing questions about who innovates and who maintains. Firstly, I want to make visible this intriguing world of repair tool developers and secondly, show to how software repair tools are deeply unsettled. The following text and slides are taken directly from my presentation.

Software repair aims to address any corruptions or malfunctions in the firmware on board the phone that enables it to boot up and run. It usually involves a process called ‘flashing’ - where technicians gain access to the embedded systems on board the phone, erase and then write new, clean copies of firmware onto the flash memory of the device.
It’s not possible to gain access to the firmware through any customer-facing part of the phone’s system. You need a special set of tools, as shown in this photograph. The vast majority of technicians in downtown Kampala are working independently of manufacturers and informally - so today I am exploring “unauthorized” tools developed by third parties.
So, the phone is here, and it’s connected via modified cables to this hardware box. This ‘flasher box’ acts as an interface between the phone and the computer. On-screen you can see a proprietary program running. This allows technicians to load the software files for re-writing to the phone.
This tool is distributed beyond this visible assemblage though - as the developers of these tools also host large servers full of proprietary firmware files belonging to mobile manufacturers, accessible through the software’s interface.

These software and hardware assemblage also allows for different types of interventions beyond flashing - such as self-testing, and removing SIM locks from phones. As I’m sure you all know, SIM locks are cryptographic limitations programmed into the device at the point of production. They restrict the phone’s operation to certain networks, providers or geographic areas.

This matters in Kampala, because mobile phones travel there from other places for a second life. Devices are imported from “recycling” companies in the global North while other phones travel through networks of family and friends - as forms of kinship infrastructure-in-the-making. Technicians unlock these phones in order to enroll them in Kampalan networks, and allow them to become part of infrastructures in Uganda.
Independent technicians described how they relied on the “geeks and hackers” who made flasher boxes and their accompanying software in order to be able to perform software repair. They met these tool developers on online repair sites such as the virtual message board known as GSM Forum.
Sections of this site have been given over to multiple tool developers to provide support to technicians - they can create a help thread or posting, and receive a reply from the developer. It is a place where we can see this ecology of third party tools in microcosm.

The site is an important way that tool developers communicate updates to their customers. Tools tend to address a specific architecture of phones, usually from one particular manufacturer. Given the speedy timeframe within which new phones get released, software programs always need to be iterated and expanded to encompass new models in order to stay competitive in this market.
Today’s case is about an update gone wrong. This is the story of one particular flasher box and its accompanying software program, called “Just Another Flasher,” or J.A.F. This flashing system repairs and unlocks Nokia models from the early 2000s with DCT4 architecture. To produce these types of repair tools, developers need to find their ways around secured boot loaders - reverse engineer the cryptographic algorithms that protect the SIM locks, and embed these into flashing software and flasher boxes. In order to remove SIM locks, the J.A.F. system reads information held locally on each particular handset, and calculates unlocking codes that are then written back onto the phone.

In October 2010, just after I had first arrived in Kampala, a technician called Wilson was showing me his collection of flasher boxes. He lifted out two red boxes called J.A.F. and explained that these were ‘dead’ boxes, which had been killed by another tool called MXKey.

He opened up GSM Forums, and navigated to a sub-forum called The Market Feedback. He scrolled down the page to show me threads by other technicians complaining of box malfunctions, and commenting on the long running ‘war’ between these two developers.
Rivalries between the developers of software repair tools played out in two different ways. First of all, these conflicts were enacted materially, through the development and distribution of tools that ‘crack’ or ‘kill’ those belonging to opponents. In the case of J.A.F., Wilson explained that:

> the MXKey could write a script in your JAF and restrict it from reading any phones - that's what it used to do... it would write something in the chip so that it doesn't detect any phone. *Field notes 21 September, 2012*

A piece of malware was hidden within an MXKey update, so some users unwittingly downloaded and installed something that was harmful to J.A.F. - another tool in their assemblage.

Another technician, Peter, interpreted the agency of these tools by describing them as animals. MXKey - which at the time came in a green plastic housing - was a green snake.

> The people used to see it as a green snake in the house, you see? You're keeping a green snake in the house because like, any time it can do harm to you when you forget and connect it... “Oh f*** I connected my JAF box to MXKey”. *Field notes 20 September, 2012*

So tool developers may work hard to circumvent the security systems encoded
by mobile phone manufacturers. But this is not a utopian story about a heroic hacker who goes up against corporations for the good of the community at large. It’s a story about an ecology of developers producing circumventions in order to make money from the sales of boxes and services to technicians, and in the process hacking and sabotaging each others’ tools.

GSM Forums is an incredibly important location in these processes. Rivalries are performed here directly, through text ‘talk’ between developers. But technicians also come to be positioned in relation to controversy, through their engagement with the Forum.

Threads in The Market Feedback section discuss the rights and wrongs of each intervention, and accounts of what happened are collectively negotiated across threads by technicians. In the case of the dead J.A.F. boxes, technicians use the testimony and consensus on the forum to make sense of what has happened to their boxes, and to negotiate solutions.
Software repair tools like J.A.F. never reach a state of fixity - these are tools-as-networks. Third party developers are always reacting and responding to the upgrade cycle of mobile manufacture. The fact that these tools work against proprietary closures, means they are always one step behind manufacturers, which we might think of as a kind of perpetual and designed-in delay.

But, these tools are also subject to rivalries from other developer teams. Even seemingly stable tools could be subject to the bite of the green snake. Because of this, technicians consume these tools very differently. Tool maintenance doesn't involve simply downloading the latest update. It involves participating deeply in this community, and keeping informed of the relationships between different tool developers and how these might affect the longevity of a flasher tool.
Technicians’ capabilities to perform repairs are materially contoured by these personal and professional rivalries between developers, and there are many others beyond this example. Conflicts that result in killing or cracking, terminate the product development process. Technicians who had paid for a tool find it cracked, and therefore no longer supported or updated, because it can no longer make money. This is an interesting moment - where a tool often still usable, and reaches a much larger audience - because it is now open and freely available to use. But for many technicians it also represents the ‘death’ of the tool as it can no longer be updated. It becomes orphaned, and will gradually obsolesce as the market for mobiles moves relentlessly on.

So I hope that this very brief ethnographic snippet has demonstrated three aspects of software repair tools -

- that they never settle into fixity and stability, but are deeply entangled in the high-speed cycles of "innovation" in the mobile market, as new handsets and firmware files are released and incorporated

- that seemingly stable tools could be subject to the bite of the green snake, meaning that technicians must be aware of relations within the developer community
- and finally to point to the situated judgments on when and how technologies die: whether a repair tool’s life is over when (as in common sense definitions) it ceases to work, or when it becomes unsupported, and ceases to iterate

Thank you!

Notes and References

1 Figures from the International Telecommunications union, generated using their


RECLAIMING REPAIR

http://reclaimingrepair.com/