SONY

Sony's Open Devices Project - What We Got Right and Wrong

Tim Bird
Senior Software Engineer
Sony Mobile Communications

SONY

Agenda

- Sony's Open Devices Project
  - Goals
  - Achievements
- What went right?
- What went wrong?
- Lessons learned

Open Devices Project

- Ambitious project to support open software on Sony Mobile's phone platforms
- 2 main areas:
  - Android Open Source Project
  - Mainline Linux kernel

Goals

Open Devices - Goals

- Allow people to run as much open source software as possible on our platforms
  - Common misconception is that the main target was standard end-user customers
    - Lots of people skeptical of the value of this
    - But here's a quote:
      - "if SONY gives us a fully Open Source Smartphone i will completely change my View about this Company ;) And i will definitely buy that Thing." Jose Ramirez, G+, July 29 2015
- Targets:
  - Least important: normal customers
  - Some importance: advanced, external developers
  - Most important: ourselves!!

Open Devices – Goals (cont.)

- Allow people to run recent kernel versions on our platforms
- Targets:
  - 3rd party developers
  - Primarily our team and our own engineers

Specific objectives

From least to most important:

5. Use open access to our hardware as a differentiator
4. Make it possible for users and developers to contribute to our platform
3. Allow our developers to submit patches upstream
2. Reduce our time-to-market, by lowering our own patch count
1. Reduce our time-to-market by fixing problems with kernel and hardware support before our product engineers encountered them

Problems (Justifications for Project)

- Time to Market:
  - Market entry delays, particularly for US market was costing us dearly
    - US phones trailed Japan and Europe by 4 to 6 months
  - We had recurring problems in our bringup and QA
- Engineering costs
  - Too many of our own patches
- Obstacles:
  - Kernel version gap – 3 years behind mainline kernel.org
  - Huge patch mountain from vendor (Qualcomm)
    - Over 20,000 commits and 1.6 million lines of code

Open access as a differentiator

- Allow people to use AOSP on our phones, and build a mod' community
- Allow people to use their hardware as they want
- Differentiate from other locked-down vendors

SONY

Facilitate 3rd party contributions

- Allow external developers to fix bugs
- Allow external developers to add value to platform
  - For example: xda-developers and cyanogenmod

SONY

Push our patches upstream

- In order to *X* we had to *Y*
  - Reduce our time to market
  - Reduce our patch burden
  - **Submit our patches upstream**
  - Give product engineers the capability to participate in open source community
  - Give our engineers access to latest kernel.org version on our hardware
    - Also known as: Closing the version gap
  - Submit base Qualcomm processor support to mainline
- Lots of steps, and that last one looks very hard!

Reduce TTM with lower patch count

- Reduce our technical debt
- Sony had 1800 patches spread across 800 git trees
  - Different combinations cherry-picked into product trees
  - Quality control was difficult
  - Increased our bringup cost for a new product

Commiter e-mail	Commits	Authors
Google/Android commits	963	61
Other	2677	828
Qualcomm	20395	635
Sony Mobile	1799	203
Between our tree and mainline base (3.4)	25843	1757

Reduce TTM by pre-fixing problems

- Had previously identified technical problems that cost us a lot of time each product cycle
  - Ex: display/touchscreen integration problems
- Sony Mobile had no way to fix upstream problems
- We couldn't push to kernel.org
  - Because of 3-year version gap
- Code dumps from Qualcomm were one-way drops
  - Only way to send something to Qualcomm is via public mailing list
  - They won't take direct e-mails or git pulls

Strategic (Secret) Goal

- Reduce cost for our vendor, and increase quality of received software
- Qualcomm has ~600 kernel engineers to support their SoCs
  - We estimated this could be reduced by half
  - Using about 15 engineers (5 of ours and 10 of theirs) over the course of 5 years
  - Cost of 75 man years, to save 300 man years/per year for our SoC vendor
- We estimated that the trickle-down benefit to Sony would outweigh our investment cost (25 man years)

SONY

And so we did the unthinkable…

- We started pushing Qualcomm code into mainline
  - Remember – this was not Sony's code, nor was it Sony's hardware (per se)

- We also started "Device Mainlining" initiative in the Linux Foundation CE Workgroup
  - To identify and remediate obstacles to mainlining, for "industry" developers

Open Devices - tactics

- Small team (6 people + a manager)
  - With some support from other developers
  - Very experienced
- Isolate our group from product schedules
- Wrote tools to analyze Sony Mobile's commits
  - Find duplicated work and code with relatively little dependencies (candidates for mainline)
- Focus on making a "contribution platform"
  - Close the version gap enough so that Sony developers could mainline their code

Achievements

Open Devices - Achievements

- AOSP on Xperia
  - Were available very soon on Xperia phone hardware
    - (one time within 24 hours of Google release)
  - Not all platform functionality available
    - E.g. camera functionality required binary blobs with legal issues for redistribution
- Mainline kernel on phone hardware
  - QualComm SoC support
- Device mainlining project

AOSP on Xperia - details

- Solved basic issues running vanilla AOSP on Xperia devices
- Gave us enough support to use as contribution platform
  - Contributed many Android patches to Google
- Good enough for daily use by engineers
  - Found and fixed bugs in Android before Google and Sony engineers
- Reworked internal deployment flow for daily builds
- Worked with Sony and Qualcomm legal to allow public release of some binary blobs

AOSP on Xperia – Results

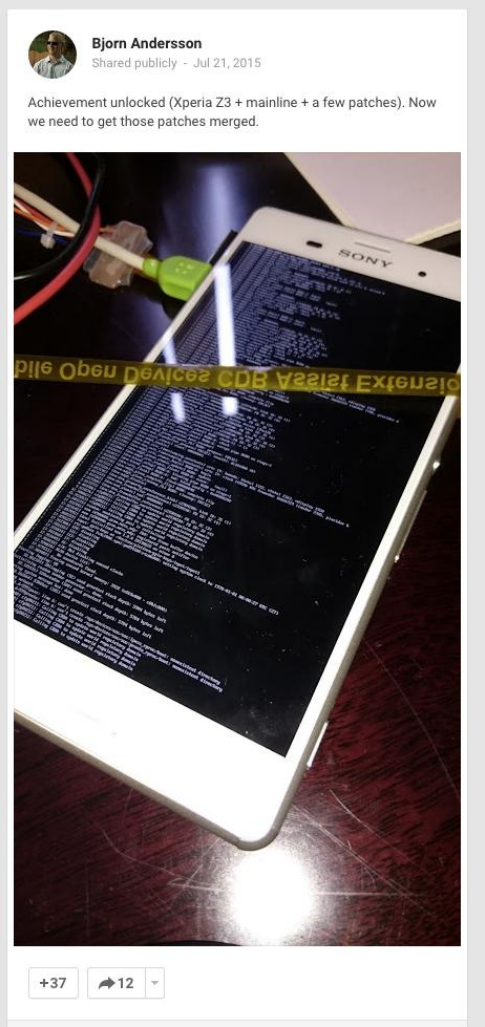
- Full support for AOSP on (many) Xperia devices
- Documentation for 3rd parties to load platform
- Github account for kernel access and contribution
- 3rd party contributions
  - 17 authors contributed 69 commits (3.4 kernel project)
    - From Feb 17 to March 17 (one month)
    - https://github.com/sonyxperidev/kernel/pulse/monthly
- “hero open source developers” program
  - David Viteri (not a Sony employee) contributed 121 accepted commits in a 2-month period to AOSP project

Mainlining of QualComm SOC support

- Device drivers for core Qualcomm infrastructure
  - Jointly developed with Qualcomm, Linaro, Redhat and the Linux community
- Framework changes needed to implement these
- Devicetree definitions for multiple products
- Anyone can use latest Linux kernel.org kernel and get clocks, regulators, gpios, uart, i2c etc. on a Honami
- Documentation

Kernel contribution platform – milestone 1

- July 2015 – mainline kernel running on Xperia phone
  - Phone shipped commercially with 3.4
  - Kernel at time was 4.1
- Immediately saw use by external party to develop GPU driver for Qualcomm chip (Rob Clark)
  - By November, we had:
    - Display running with acceleration
    - WiFi, USB, touchscreen



Device Mainlining project

- Survey of obstacles for “industry” engineers
- White paper
- “Overcoming Obstacles/Best Practices” talk given at 6 events in 2014/2015
- Identify major out-of-mainline areas in kernel
  - Discussed some issues at Kernel Summit 2015
- Work with Linaro and community to address specific needs (technical projects)
  - Ex. USB charger framework

SONY

Objectives scorecard

5. Open access as a differentiator

4. Facilitate user contributions

3. Submit patches upstream

2. Lower our patch count

1. Fix problems before our engineers

0. Reduce time-to-market

SONY

Objectives scorecard

5. Open access as a differentiator — Some visibility
4. Facilitate user contributions — 3rd party – good; Sony engineers - fail
3. Submit patches upstream — Qualcomm patches - good; Sony patches - fail
2. Lower our patch count — Fail
1. Fix problems before our engineers — AOSP - good; Linux kernel - partial
0. Reduce time-to-market — Not yet

What went right?

What went right?

- We made significant progress on milestone deliveries and some key low-level features
- Our efforts were visible outside the company, and caught the attention of customers and 3rd party developers
- Our team was successfully insulated from product issues
- External developers contributed to, and built on our work
- We actually achieved our technical milestones!!!

What went wrong?

What went wrong (1)?

- It took us too long
  - Some patches took longer to get upstream than we expected
    - Some community maintainers were unresponsive or stubborn
      - Needed a faster mechanism to fail forward (more persistence in pushing patches)
  - Some patches were impossible to get upstream, but it took months to figure this out (ex: NFC)
  - We kept trying small patches that ended up requiring substantial changes upstream
  - At first, we didn't work on the dependencies in order
- Some key management was unconvinced of the eventual savings

What went wrong (2)?

- We couldn't apply pressure to Qualcomm through our purchasing channel
- We were missing key information about the SOC
  - Undocumented registers cost us at least 3 months on one driver
  - Is a significant problem "proxying" someone else's driver
- Some of our team members didn't get proficient with upstreaming
  - Reduced our capacity

What went wrong (3)?

- We ran out of runway
  - Due to version gap, our mainline work was not going to show up in Sony products for 4 years
  - Business climate for our division put our group on the chopping block after 2.5 years
    - We had bad luck with our location in the overall business (US division)
  - We didn't make it to phase 2
- We fought with our own infrastructure for the first year
  - Campus experimented with IPV6, but it failed
  - Took us too long to set up contribution infrastructure
- Our team also did other activities
  - Not product stuff, but I did CEWG stuff, others did other internal projects – we were not concentrated enough

What went wrong (3)?

- We ran out of runway
  - Due to version gap, our mainline work was not going to show up in Sony products for 4 years
  - Business climate for our division put our group on the chopping block after 2.5 years
    - We had bad luck with our location in the overall business (US division)
  - We didn't make it to phase 2
- We fought with our own infrastructure for the first year
  - Campus experimented with IPV6, but it failed
  - Took us too long to set up contribution infrastructure
- Our team also did other activities
  - Not product stuff, but I did CEWG stuff, others did other internal projects – we were not concentrated enough

What went wrong (4)?

- We were too slow to call for “management” help from the community
  - Sometimes we would wait 2 months before escalating a maintainer issue to a higher authority
  - We didn’t push maintainers hard enough (it’s really hard – we understand they’re busy, but we let things go too long)
- Not enough experience with the technology
  - E.g. USB – we weren’t the original authors of the code
  - Took me 6 months just to learn technology
  - Was a “proxy” problem
- Kernel code was changing underneath us
  - We started on 3.10, moved to 3.14, and started major contributions on 4.1, then incremental updates after that
  - Each kernel version change brought changes to some of the sub-systems we were working on (kernel frameworks were changing)

What went wrong (5)?

- We didn't get support from internal technical teams
  - Could have helped with "proxy" problem
  - Remote office was a factor (main developers in Japan and Sweden – we were in US)
- Underestimated the difficulty of changing the industry dynamics
  - Version inertia much higher than expected
    - When we started, version gap was 3 years.
    - We hoped to get it to 1 day for our contribution platform, and 1 year for products.
    - It got bigger during our project!
  - Weird dependencies between Google, SoC Vendors

Lessons

Lessons learned

- You need enough time to reap the benefits
- Should consider mainlining like you would R&D
  - With long-term payoff
- Very hard to quantify ROI
  - If we had succeeded, Sony Mobile engineers would have "magically" had less work to do during their product development
    - (because we solved their problems in advance)
  - Would have "magically" met our time-to-market goals, by reducing issues during bring-up and QA
  - But no one would have thanked us

Lessons Learned

- Get required expertise
- Be more persistent with community
  - Build relationships early
- Build out “contribution infrastructure” immediately
- Keep management informed and interested

Thanks!