Automating Software Recovery

Catherine Garabedian

Embedded Software Engineer

Automating Software Recovery
Presentation Overview

- About Kubos
- Description of Issues
- Current Solution
- Risks and Challenges
- Customer Experience
- Future Development

Automating Software Recovery
About Me and the Company

- Embedded SW Engineer with Kubos for 2.5 yrs
- KubOS - Full-stack flight software
  - Achieved flight heritage in 2018
- Headquarters - Denton, TX

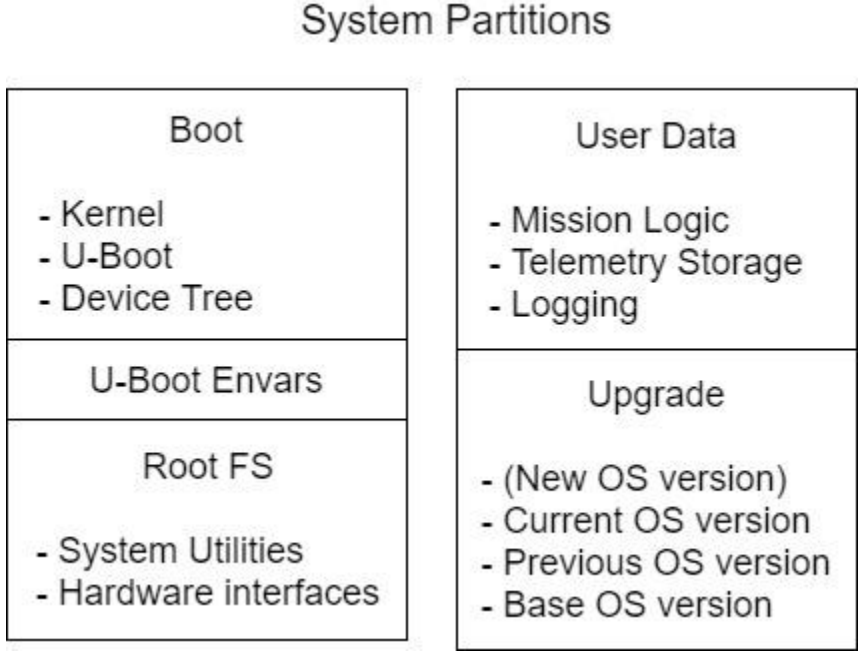
Automating Software Recovery
Problem Overview

While a satellite is in orbit, it might encounter problems with its flight software which cannot be handled from the ground, or which need to be handled in a time sensitive manner.

- Kernel corruption
- File system corruption
- Problematic system upgrade

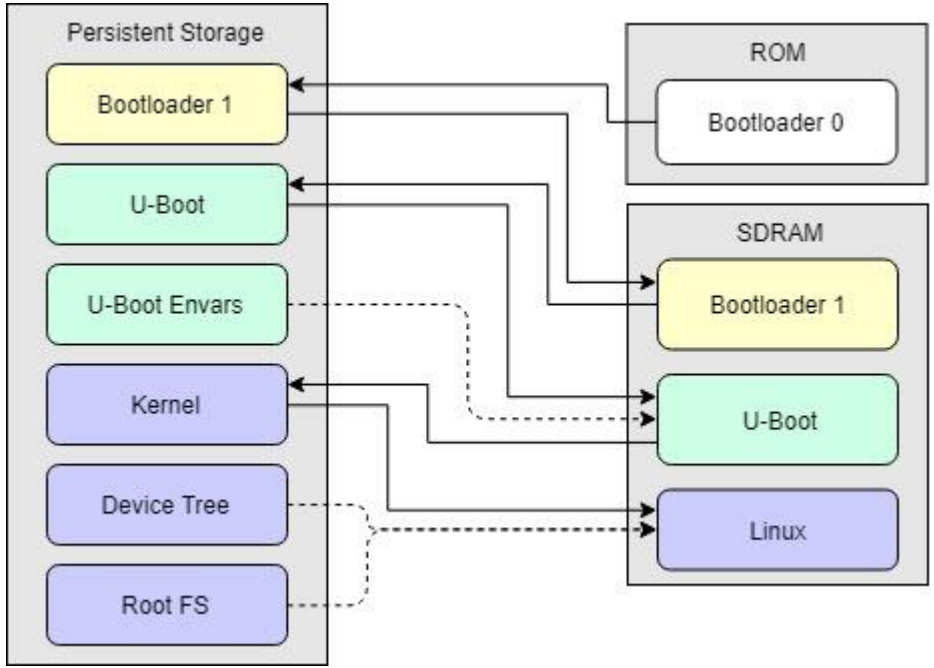
Automating Software Recovery
Our Current Solution

- Run `fsck` on OS start against relevant partitions
- Use Monit to monitor and restart long-running processes
- U-Boot OS recovery



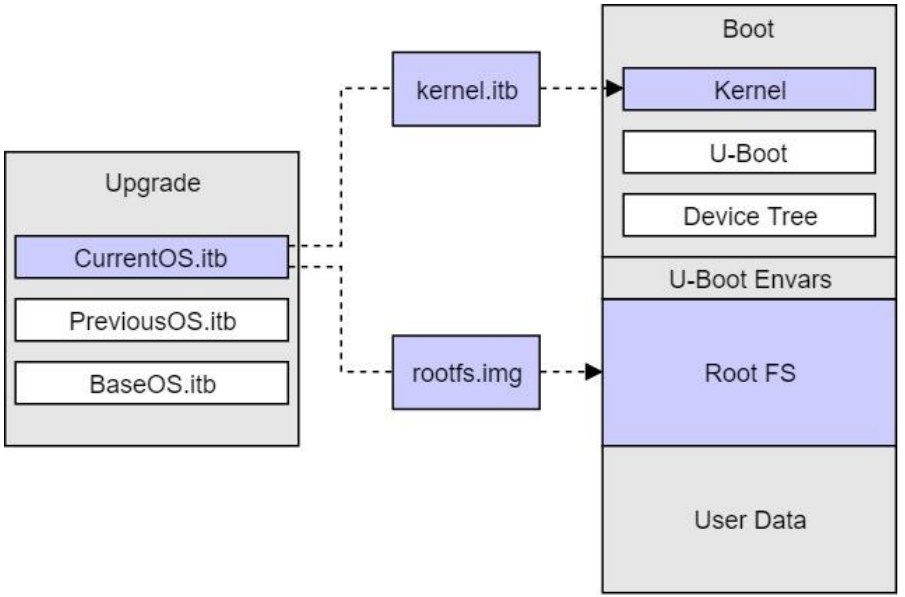
Automating Software Recovery
More details about U-Boot

- “Das U-Boot is an open-source, primary boot loader used in embedded devices to package the instructions to boot the device's operating system kernel.”
- Responsible for verifying and loading the OS and OS-related files into SDRAM



Automating Software Recovery
U-Boot Recovery Process

- When triggered, the recovery logic will attempt to reload a known-good version of the OS
- On subsequent failures, older versions are tried
- If all stored versions fail, the system can go into an alternate boot process



Automating Software Recovery
How Does This Process Get Triggered?

- Kernel checksum verification failure
- Failed boot count limit exceeded
  - Boot counter is normally set to zero after OS successfully boots
- Manual trigger

Automating Software Recovery
Risks and Challenges

- Logic won't be triggered if:
  - The system doesn't reboot
  - The system makes it through the OS boot successfully, but encounters errors later on
  - The partition containing the U-Boot envars (i.e. the boot counter) is put into read-only mode
- Devs must pre-define which areas of storage can be recovered by U-Boot
- U-Boot, itself, can also be recovered this way, but this is very risky
- Can upload new OS files while in orbit, but full OS files are ~60MB
  - Can update through diffs instead to reduce link load

Automating Software Recovery
Customer Experiences and Notes

- Overall, enjoyed working with the logic
- Liked: The staged fallback system
- Changed:
  - Tweaked the logic to use binary diffs for OS upgrades, which reduces the bandwidth needed to uplink a new OS file
  - Moved logic to reset boot counter until after FSW initialization completed
- Disliked: Testing the recovery process

Automating Software Recovery
Future Plans

- Automate our existing process of upgrading and rolling back mission logic
- Explore binary diffs for OS files
- Back-up bootloader logic

Automating Software Recovery
Conclusion

- Adding recovery automation helps enhance system flexibility, stability, and resiliency
- Strategies used are relatively simple
  - Reduces risk
  - Requires minimal development time
- Not a front-line defense, but an important auxiliary support feature

KUBOS

Main Website: kubos.com
Documentation: docs.kubos.com
Source Code: github.com/kubos