Templated Android App Architecture

Jeremy Oosterbaan
Advisor (Professional): Isaac Rose

My Employer

I've completed this capstone project for my current co-op & part-time employer: **Performance Electronics, Ltd**. (PE). The resources & support they've provided has enabled me to take on such a project. The project's success is largely due to the freedom & responsibility which PE awarded me.

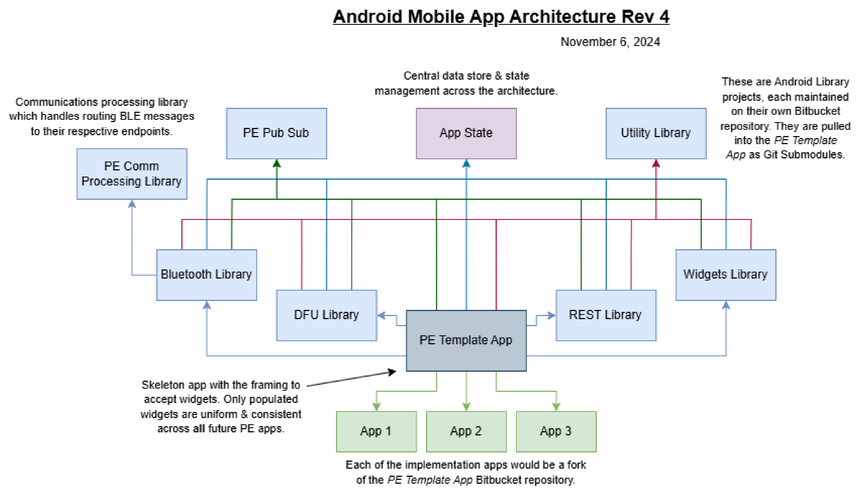
The Problem

PE is an embedded systems company, developing both hardware & software. We build physical modules which communicate to mobile apps developed for Android/iOS. Previously, the process of creating a new mobile app was costly and inefficient. Each mobile app had to be started essentially from scratch. Moreover, there is little to no reusability in the previous mobile app structure, which means a large portion of time is spent merely copying or reimplementing already previously developed components. This outdated approach to mobile development is tedious & not easily scalable.

My Proposed Solution

I proposed a templated structure with one consistent, generic framework app. Additionally, we'd implement a collection of UI widgets that can be populated throughout the app(s). Each project or app would be a fork of the template app repository, with a different configuration of widgets & themes. Moreover, this should be a less monolithic application, following closer to the popular microservices architecture pattern. In this way, the various components of the app would be distinct, independent libraries/modules that communicate via a publisher/subscriber communication protocol.

Architecture Diagram



Key Components

A lot of foundational items needed to be developed in tangent to enable this project. A few of these items are briefly outline below:

- **PE UI Specification Document** – A 100+ UI specification document outlining available widgets, theming, template app screens, etc. See physical handout or GitHub repo for details.
- **PE Communication Protocol** – An entirely new communication protocol to be implemented across any medium (BLE/CAN/etc.).
- **PE Pub/Sub Library** – A generic publisher/subscriber library for enabling cross-module communication.
- **PE Communication Processing Library** – A routing library to actually implement the designed communication protocol. Lots of unit tests in this library, ensuring the comms are implemented properly.
- **Device Firmware Update (DFU)** – A custom DFU mechanism for updating the firmware of our boards over-the-air (OTA).

Constraints

- **Developers & Resources** – The #1 constraint for this project was the lack of resources & developers available. This was a large, complicated project for a single (junior) developer to take on alone.
- **Hardware** – PE products are physical devices, and thus software development was reliant on hardware's progress. If hardware wasn't available, I was not able to continue until I could get updated hardware.

Future Improvements

- **More Extensive Testing** – Due to the resourcing constraints, testing is not as thorough as I personally would've liked. A future effort to make even more extensive unit & integration tests would be beneficial.
- **Improved Documentation** – Similarly, documentation on some of the components (UI Spec or Comms Spec) is great. But other aspects like the DFU/BLE, or Pub/Sub libraries are lacking thorough documentation.

Technologies

Java
Kotlin
espresso
UI TESTING FOR ANDROID
ATLASSIAN Bitbucket
Bluetooth Low Energy

Contact Info: Jeremy Oosterbaan | **Computer Science, 2025** | **oosterjm@mail.uc.edu**