Multicore Avionics Operational Analysis

Lightning Talk 6: Design Check-In

Team: sddec24-09

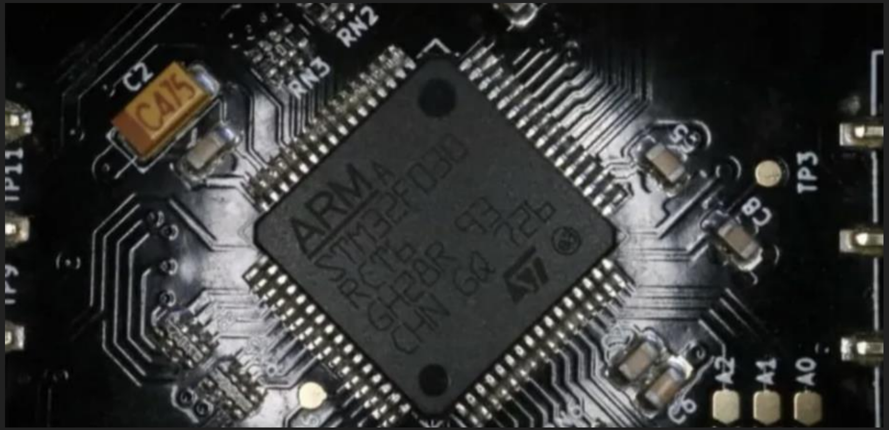
Team members: Alex Bashara, Joe Dicklin, Hankel Haldin, Anthony Manschula

Faculty advisors: Dr. Zambreno & Dr. Jones

Client: Boeing

Project Overview

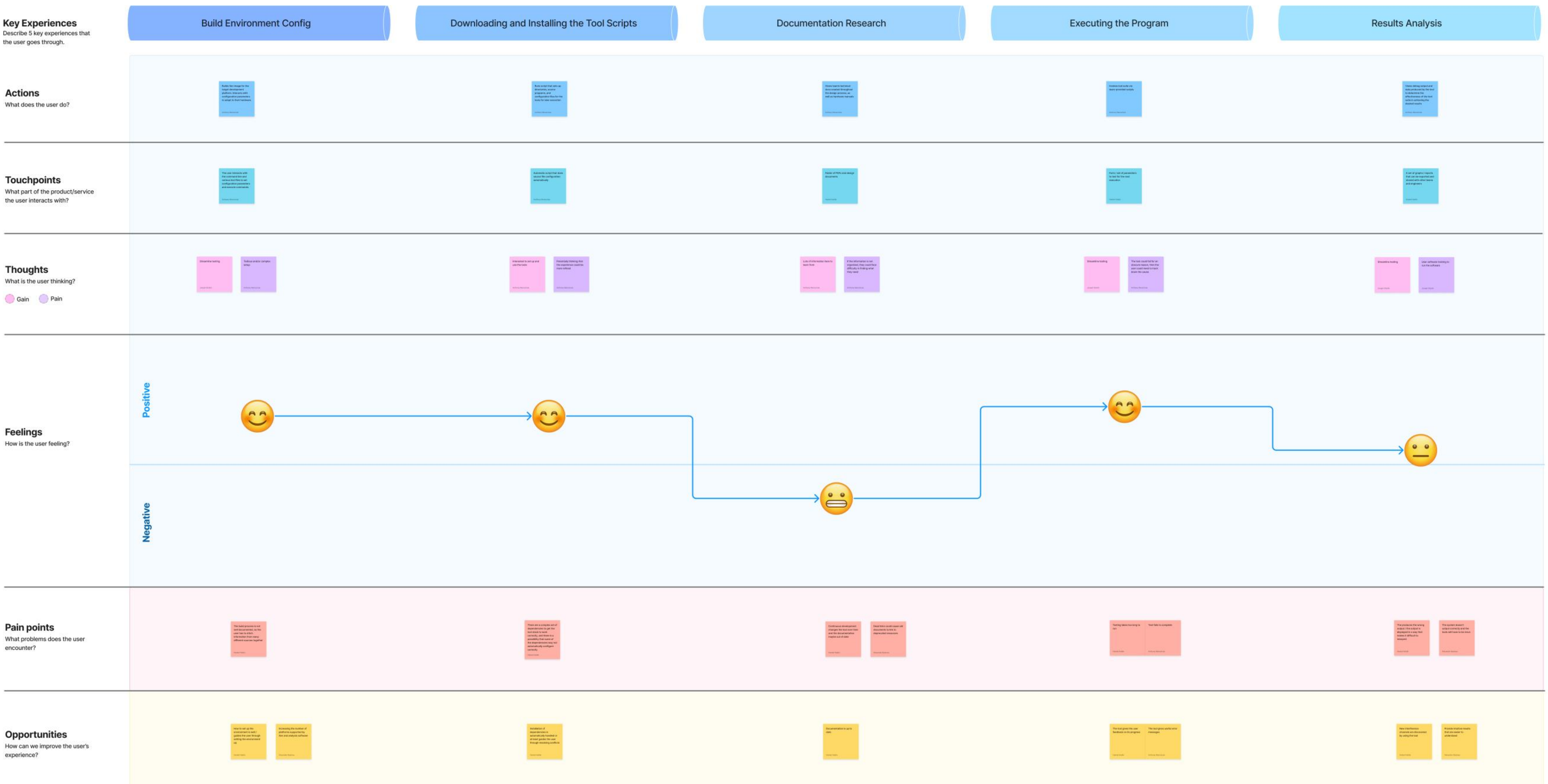
- Multicore avionics systems
  - Meet the increasing compute demand of modern avionics software with concurrent execution of programs
  - Concurrent programs competing for shared resources
    - Introduce interference & negatively affect execution timing behavior
    - Ability to examine and verify the effects of interference is critical for FAA certification
- Hardware: ARM-based SBC and bare-metal hypervisor
  - Hypervisor allows more granular control of resource allocation to programs
  - Run control applications with the system under extreme load
    - Collect and analyze data on worst-case execution time (WCET)



Problem Statement

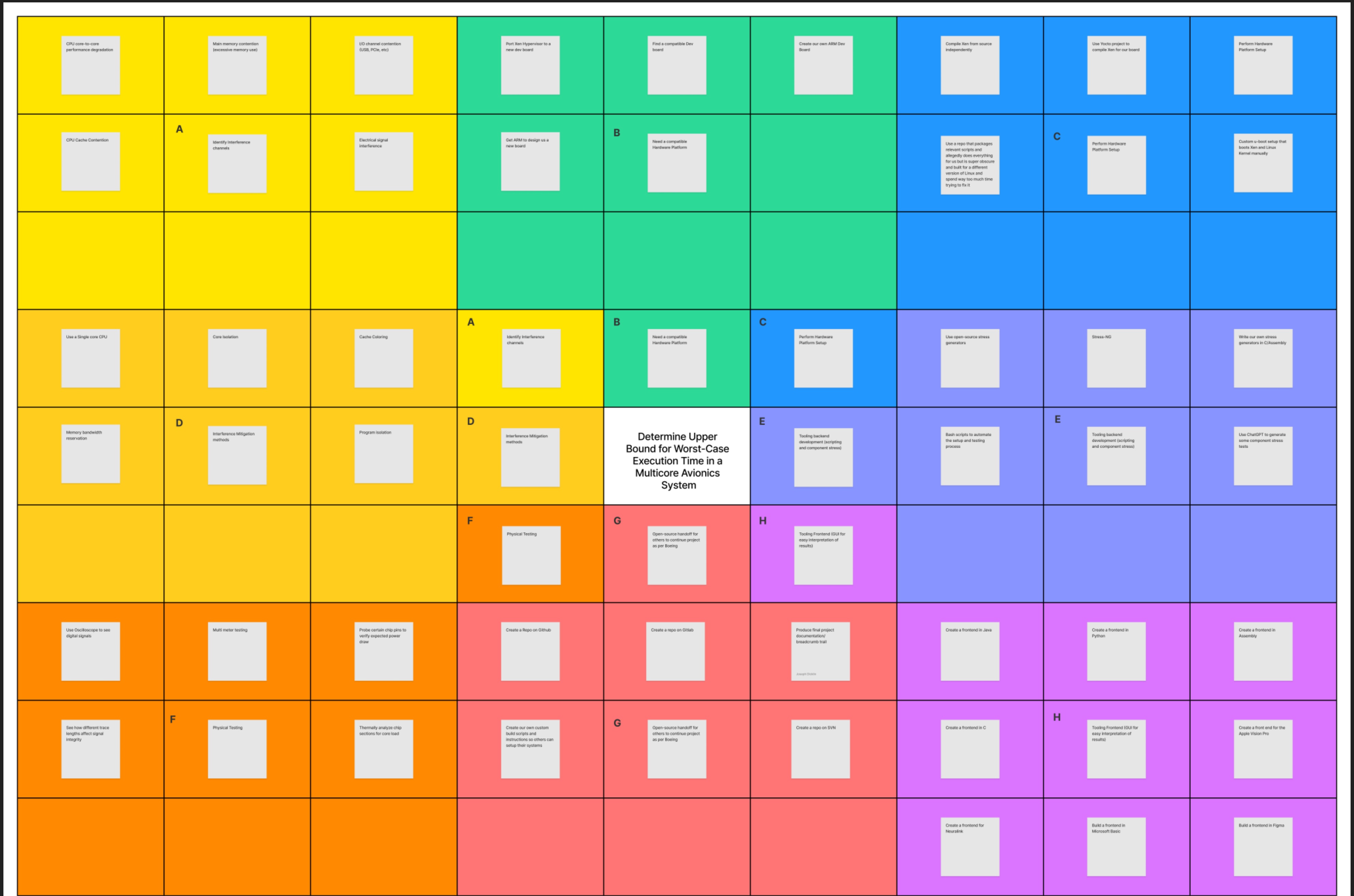
- Our project addresses a need for a suite of open-source tools to characterize interference modes in multicore avionics systems
  - Identify potential interference channels on a multicore platform
    - "Control tests" as a baseline that target each channel for use in analysis
  - Set of tools to apply stress and contention to the identified subsystems in a controlled manner
  - Set of tools/methods to demonstrate mitigation of interference channels
  - Integrate testing and analysis tools into unified suite

Artifacts – Journey Map



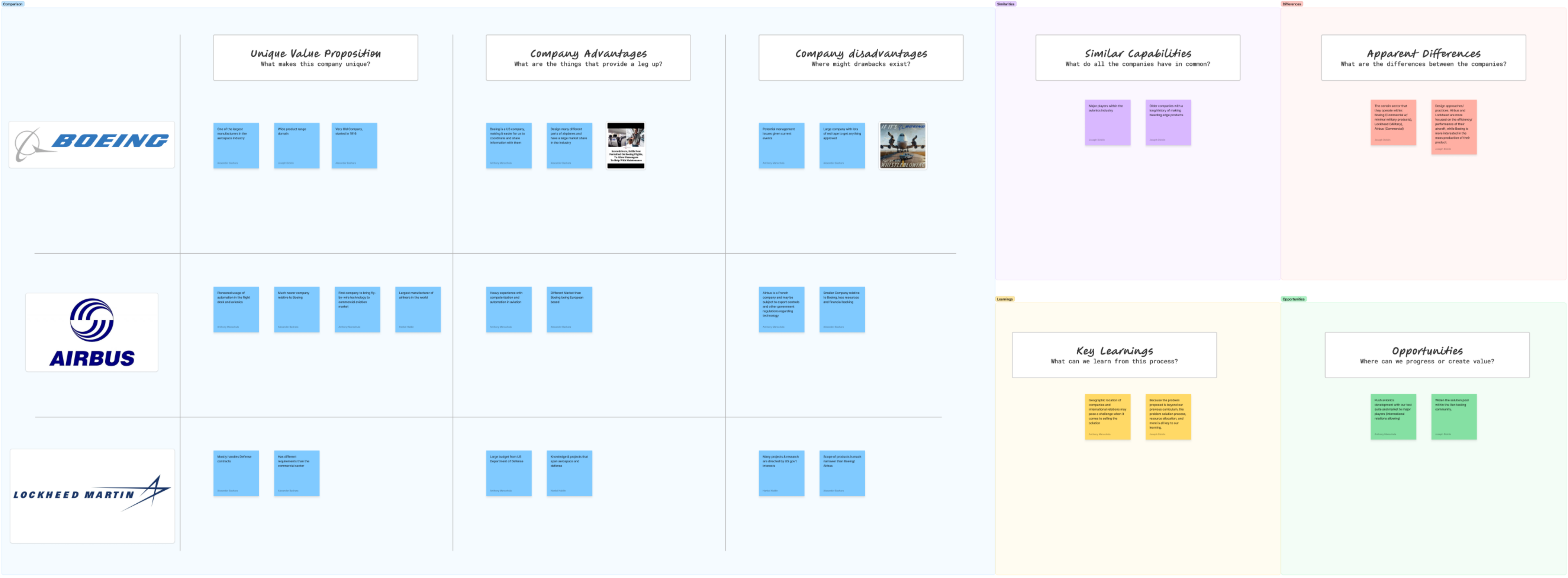
Key Experiences Describe 5 key experiences that the user goes through.	Build Environment Config	Downloading and Installing the Tool Scripts	Documentation Research	Executing the Program	Results Analysis
Actions What does the user do?	Builds Xen image for the target development platform. Interacts with configuration parameters to adapt to their hardware	Runs script that sets up directories, source programs, and configuration files for the tools for later execution	Views team's technical docs created throughout the design process, as well as hardware manuals	Invokes tool suite via team-provided scripts	Views debug output and data produced by the tool to determine the effectiveness of the tool suite in achieving the desired results
Touchpoints What part of the product/service the user interacts with?	The user interacts with the command line and various text files to set configuration parameters and execute commands.	Automatic script that does source file configuration automatically	Folder of PDFs and design documents	Form / set of parameters to test for the next execution	A set of graphs / reports that can be exported and shared with other teams and engineers
Thoughts What is the user thinking? ● Gain ● Pain	Streamline testing Tedious and/or complex setup	Interested to set up and use the tools Potentially thinking that the experience could be more refined	Lots of information here to learn from If the information is not organized, they could face difficulty in finding what they need	Streamline testing The test could fail for an obscure reason, then the user could need to track down the cause	Streamline testing User software training to run the software
Feelings How is the user feeling?					
Pain points What problems does the user encounter?	The build process is not well documented, so the user has to stitch information from many different sources together	There are a complex set of dependencies to get the tool stack to work correctly, and there is a possibility that some of the dependencies may not automatically configure correctly	Continuous development changes the tool over time and the documentation maybe out of date Dead links could cause old documents to link to deprecated resources	Testing takes too long to run Test fails to complete	The produces the wrong output / the output is displayed in a way that makes it difficult to interpret The system doesn't output correctly and the tests will have to be rerun
Opportunities How can we improve the user's experience?	How to set up the environment is well / guides the user through setting the environment up Increasing the number of platforms supported by Xen and analysis software	Installation of dependencies is automatically handled or at least guides the user through resolving conflicts	Documentation is up to date	The tool gives the user feedback on its progress The tool gives useful error messages	New interference channels are discovered by using the tool Provide intuitive results that are easier to understand

Artifacts - Ideation



CPU core-to-core performance degradation	Main memory contention (excessive memory use)	I/O channel contention (USB, PCIe, etc)	Port Xen Hypervisor to a new dev board	Find a compatible Dev board	Create our own ARM Dev Board	Compile Xen from source independently	Use Yocto project to compile Xen for our board	Perform Hardware Platform Setup
CPU Cache Contention	A Identify Interference channels	Electrical signal interference	Get ARM to design us a new board	B Need a compatible Hardware Platform		Use a repo that packages relevant scripts and allegedly does everything for us but is super obscure and built for a different version of Linux and spend way too much time trying to fix it	C Perform Hardware Platform Setup	Custom u-boot setup that boots Xen and Linux Kernel manually
Use a Single core CPU	Core Isolation	Cache Coloring	A Identify Interference channels	B Need a compatible Hardware Platform	C Perform Hardware Platform Setup	Use open-source stress generators	Stress-NG	Write our own stress generators in C/Assembly
Memory bandwidth reservation	D Interference Mitigation methods	Program isolation	D Interference Mitigation methods	Determine Upper Bound for Worst-Case Execution Time in a Multicore Avionics System	E Tooling backend development (scripting and component stress)	Bash scripts to automate the setup and testing process	E Tooling backend development (scripting and component stress)	Use ChatGPT to generate some component stress tests
			F Physical Testing	G Open-source handoff for others to continue project as per Boeing	H Tooling Frontend (GUI for easy interpretation of results)			
Use Oscilloscope to see digital signals	Multi meter testing	Probe certain chip pins to verify expected power draw	Create a Repo on Github	Create a repo on Gitlab	Produce final project documentation/ breadcrumb trail Joseph Dicklin	Create a frontend in Java	Create a frontend in Python	Create a frontend in Assembly
See how different trace lengths affect signal integrity	F Physical Testing	Thermally analyze chip sections for core load	Create our own custom build scripts and instructions so others can setup their systems	G Open-source handoff for others to continue project as per Boeing	Create a repo on SVN	Create a frontend in C	H Tooling Frontend (GUI for easy interpretation of results)	Create a front end for the Apple Vision Pro
						Create a frontend for Neuralink	Build a frontend in Microsoft Basic	Build a frontend in Figma

Artifacts – Market/Client Research



Comparison

	Unique Value Proposition What makes this company unique?	Company Advantages What are the things that provide a leg up?	Company disadvantages Where might drawbacks exist?
BOEING	One of the largest manufacturers in the aerospace industry — Alexander Bashara Wide product range domain — Joseph Dicklin Very Old Company, started in 1916 — Alexander Bashara	Boeing is a US company, making it easier for us to coordinate and share information with them — Anthony Manschula Design many different parts of airplanes and have a large market share in the industry — Alexander Bashara Screwdrivers, Drills Now Permitted On Boeing Flights, To Allow Passengers To Help With Maintenance	Potential management issues given current events — Anthony Manschula Large company with lots of red tape to get anything approved — Alexander Bashara IF IT'S BOEING YOU'RE NOT WHISTLEBLOWING
AIRBUS	Pioneered usage of automation in the flight deck and avionics — Anthony Manschula Much newer company relative to Boeing — Alexander Bashara First company to bring fly-by-wire technology to commercial aviation market — Anthony Manschula Largest manufacturer of airliners in the world — Haskel Haldin	Heavy experience with computerization and automation in aviation — Anthony Manschula Different Market than Boeing being European based — Alexander Bashara	Airbus is a French company and may be subject to export controls and other government regulations regarding technology — Anthony Manschula Smaller Company relative to Boeing, less resources and financial backing — Alexander Bashara
LOCKHEED MARTIN	Mostly handles Defense contracts — Alexander Bashara Has different requirements than the commercial sector — Alexander Bashara	Large budget from US Department of Defense — Anthony Manschula Knowledge & projects that span aerospace and defense — Haskel Haldin	Many projects & research are directed by US gov't interests — Haskel Haldin Scope of products is much narrower than Boeing/ Airbus — Alexander Bashara

Similarities

Similar Capabilities
What do all the companies have in common?

- Major players within the avionics industry — Joseph Dicklin
- Older companies with a long history of making bleeding edge products — Joseph Dicklin

Differences

Apparent Differences
What are the differences between the companies?

- The certain sector that they operate within: Boeing (Commercial w/ minimal military products), Lockheed (Military), Airbus (Commercial) — Joseph Dicklin
- Design approaches/ practices. Airbus and Lockheed are more focused on the efficiency/ performance of their aircraft, while Boeing is more interested in the mass production of their product. — Joseph Dicklin

Learnings

Key Learnings
What can we learn from this process?

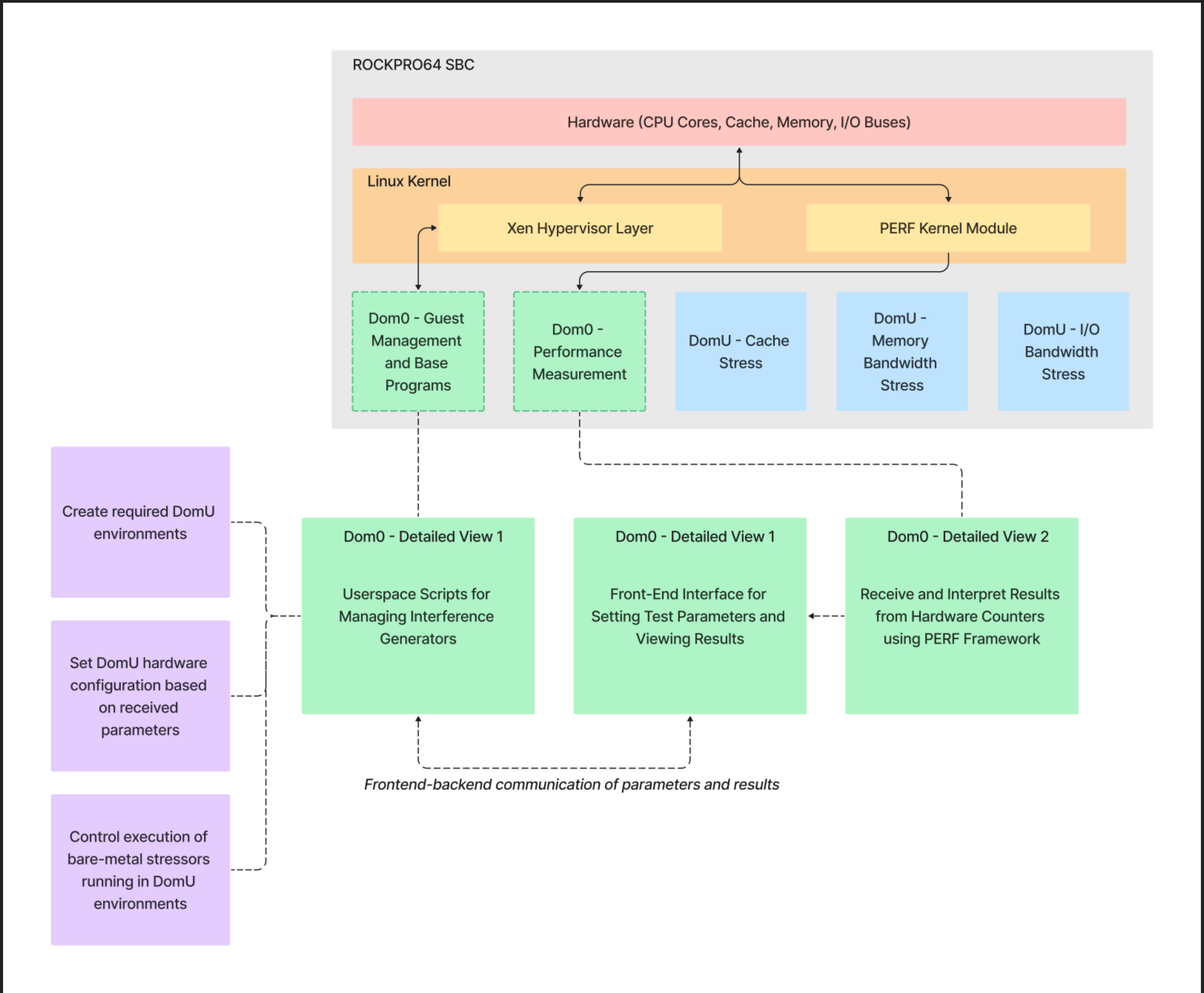
- Geographic location of companies and international relations may pose a challenge when it comes to selling the solution — Anthony Manschula
- Because the problem proposed is beyond our previous curriculum, the problem solution process, resource allocation, and more is all key to our learning. — Joseph Dicklin

Opportunities

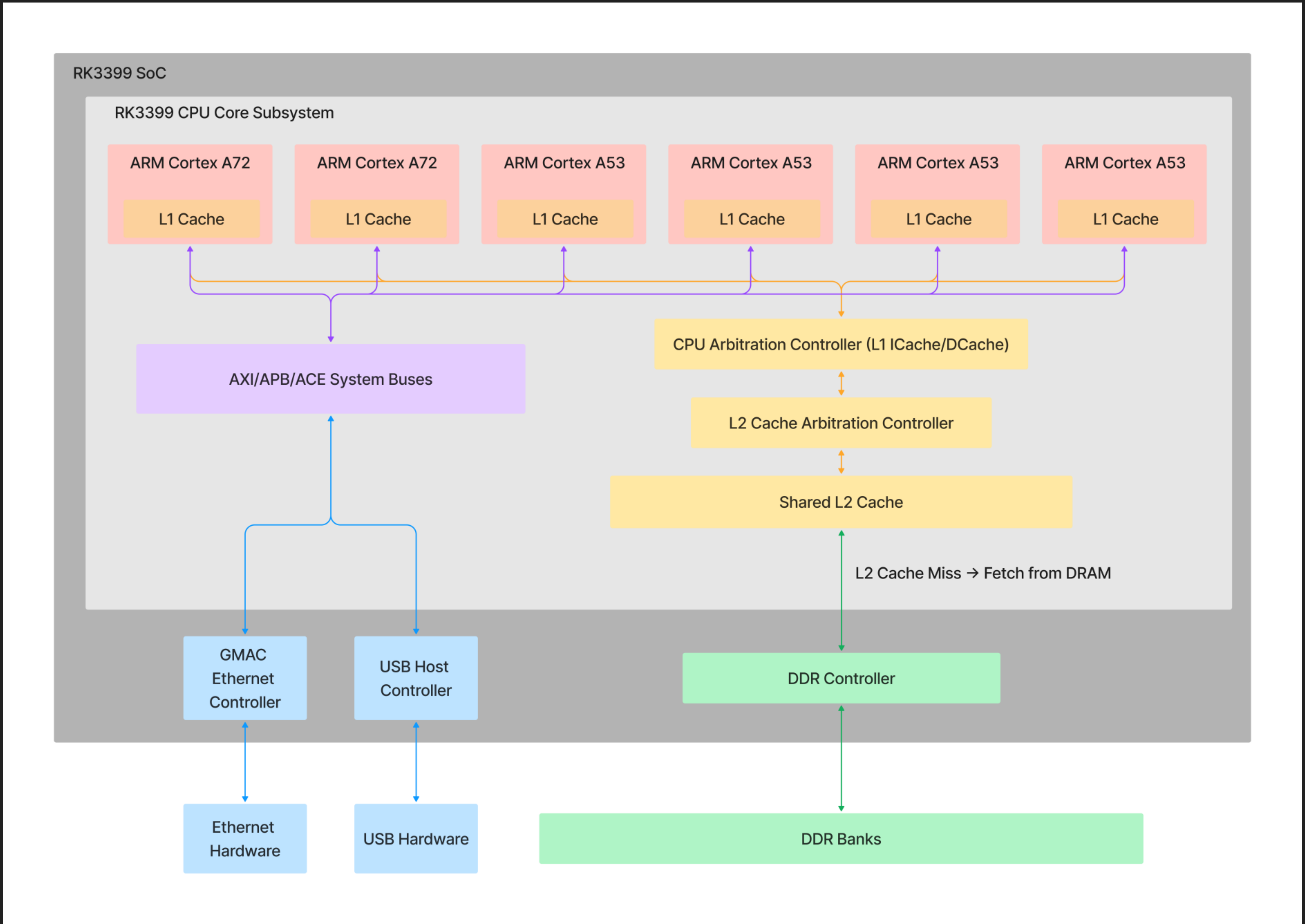
Opportunities
Where can we progress or create value?

- Push avionics development with our test suite and market to major players (international relations allowing) — Anthony Manschula
- Widen the solution pool within the Xen testing community. — Joseph Dicklin

Artifacts – System Components


ROCKPRO64 SBC
Hardware (CPU Cores, Cache, Memory, I/O Buses)
Linux Kernel
Xen Hypervisor Layer
PERF Kernel Module
Dom0 - Guest Management and Base Programs
Dom0 - Performance Measurement
DomU - Cache Stress
DomU - Memory Bandwidth Stress
DomU - I/O Bandwidth Stress
Create required DomU environments
Set DomU hardware configuration based on received parameters
Control execution of bare-metal stressors running in DomU environments
Dom0 - Detailed View 1
Userspace Scripts for Managing Interference Generators
Dom0 - Detailed View 1
Front-End Interface for Setting Test Parameters and Viewing Results
Dom0 - Detailed View 2
Receive and Interpret Results from Hardware Counters using PERF Framework
Frontend-backend communication of parameters and results

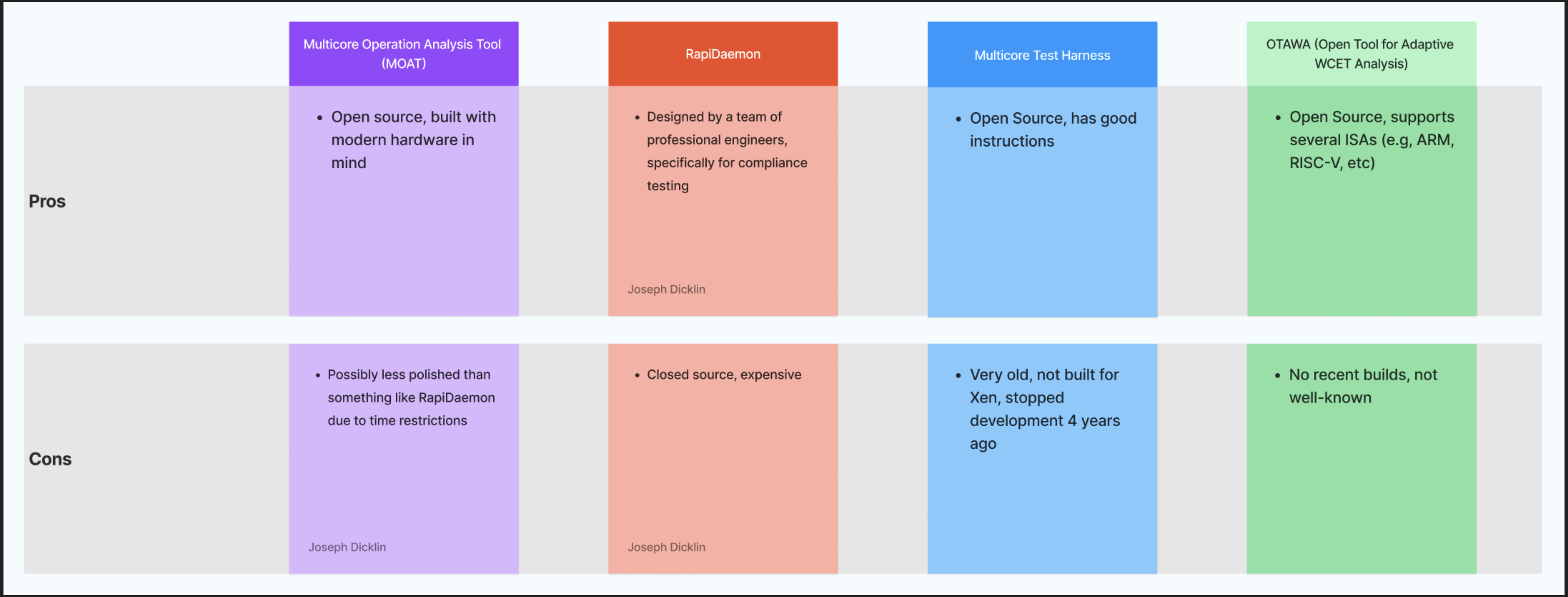
Artifacts – Contention Points



Addressing User Needs

- Existing User Needs
  - A comprehensive, open-source tool suite to accurately measures the WCET (Worst Case Execution Time) for a given hardware platform
  - Hardware platform must use an ARM based ISA (Instruction Set Architecture)
  - The output of the tool must be presentable to both technical and non-technical audiences
- Areas in which our design could improve:
  - Extend the test suite to cover interference channels other than cache, memory, and I/O interference.
  - The current implementation is a text-based command line tool
    - We could scaffold a basic GUI interface for non-technical users

Economic – Pros and Cons



	Multicore Operation Analysis Tool (MOAT)	RapiDaemon	Multicore Test Harness	OTAWA (Open Tool for Adaptive WCET Analysis)
Pros	• Open source, built with modern hardware in mind	• Designed by a team of professional engineers, specifically for compliance testing Joseph Dicklin	• Open Source, has good instructions	• Open Source, supports several ISAs (e.g, ARM, RISC-V, etc)
Cons	• Possibly less polished than something like RapiDaemon due to time restrictions Joseph Dicklin	• Closed source, expensive Joseph Dicklin	• Very old, not built for Xen, stopped development 4 years ago	• No recent builds, not well-known

Technical Complexity

- Our project approach leverages and aggregates several core computer engineering and embedded systems concepts, including
  - Memory hierarchy
  - Caching
  - Performance isolation
  - FPGAs
  - Hypervisors
  - Program analysis and performance profiling
- Our project requires us to research and thoroughly understand the layout and behavior of the systems we are working on to ensure our testing is thorough. We must:
  - Understand technical documents for the IPs that we are working with, such as the Rockchip SoC datasheets and ARM architecture documentation

For the Audience

- Conclusion
  - As a whole, the team believes that we have a solution that sufficiently addresses the needs of our client and offers features that existing solutions do not.
  - Furthermore, our project leverages and improves the technical skills of our team members.
- Questions?