Zephyr® Project

Developer Summit 2022

June 8-9, 2022

Mountain View, CA + Virtual

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Architecting an IoT Product Line: Scoping

Agenda

- Product Line Terminology
- Product Line Engineering (PLE) Scoping Process
- Application of PLE to Zephyr Ecosystem
- Q & A

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Product Line Terminology

References

Cited:

- SEVOCAB - Software and Systems Engineering **Vocabulary** (computer.org)
- ISO - ISO/IEC 26550:2015 - Software and systems engineering — **Reference model** for product line engineering and management
- ISO - ISO/IEC 26552:2019 - Software and systems engineering — **Tools and methods** for product line architecture design
- ISO - ISO/IEC 26580:2021 - Software and systems engineering — Methods and tools for the **feature-based approach** to software and systems product line engineering

Background:

- Software Product Lines Collection (cmu.edu)
- SWEBoK - Software Engineering Body of Knowledge (swebokwiki.org)

What is a "Product Line"?

From SEVOCAB for "Product Line" (emphasis mine):

> **(2)** set of products or services sharing **explicitly defined and managed common and variable features** and
>
> relying on the **same domain architecture** to meet the common and variable needs of specific markets
>
> *(ISO/IEC 26550:2015 Software and systems engineering--Reference model for product line engineering and management, 3.16)*

What is a "Domain Architecture"?

From SEVOCAB for "Domain Architecture" (emphasis mine):

> **(2) common architecture** for a product line that can **embrace variability of member products**
>
> *(ISO/IEC 26552:2019 Software and systems engineering--Tools and methods for product line architecture design, 3.5)*
>
> *Note:* The domain architecture contains **the designs that are intended to satisfy requirements** specified in the domain model. The domain architecture documents design, whereas the domain model documents requirements.

What about the product architecture?

From SEVOCAB for “Application Architecture” (emphasis mine):

> **(2) architecture concept**, including the architectural structure and rules (e.g., common rules and constraints), and architecture artifacts (such as descriptions) **that constrains a specific member product** within a product line
>
> *(ISO/IEC 26552:2019 Software and systems engineering--Tools and methods for product line architecture design, 3.1)*

Actual product architecture comes out of the product line, even if it is more complicated than the “conceptual” architecture from the logical model.

What is a "Product Line Scoping"?

From SEVOCAB for "Product Line Scoping" (emphasis mine):

> **(1)** process for **defining** the **member products** that will be produced within a product line and the **major common and variable features** among the products, analyzes the products from an economic point of view, and **controls and schedules the development, production, and marketing** of the product line and its products
>
> *(ISO/IEC 26550:2015 Software and systems engineering--Reference model for product line engineering and management, 3.20)*

What is a "Product Line Platform"?

From SEVOCAB for "Product Line Platform" (emphasis mine):

> **(1)** product line **architecture**, a **configuration management** plan, and **domain assets**, enabling application engineering to effectively reuse and produce a set of derivative products
>
> *(ISO/IEC 26550:2015 Software and systems engineering--Reference model for product line engineering and management, 3.18)*

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Product Line Engineering (PLE) Scoping Process

Scoping Process Overview

- What is the scope of the problem?
  - Captured as User Needs & Stakeholder Requirements
- How much of the problem are you going to solve?
  - Captured as System Requirements (requirements on any solution)
- What does your solution look like?
  - Captured as System Architecture Definition
  - Specifies the interfaces and behavior at the system boundaries
  - Decompose into functional blocks (System Elements), resulting in System Element Requirements and new Interface Specifications.
- Schedule development & marketing plan

Decomposition Activity:

- Given a specified set of interfaces and behaviors
  - Identify (sub-)elements that collectively produce the behaviors and implement specified interfaces.
  - Assign responsibilities and behaviors to each (sub-)element, generating (sub-)element Requirements.
  - Specify communication interfaces between each (sub-)element, generating additional Interface Specifications.

Recursively decompose until composable elements (components) are:

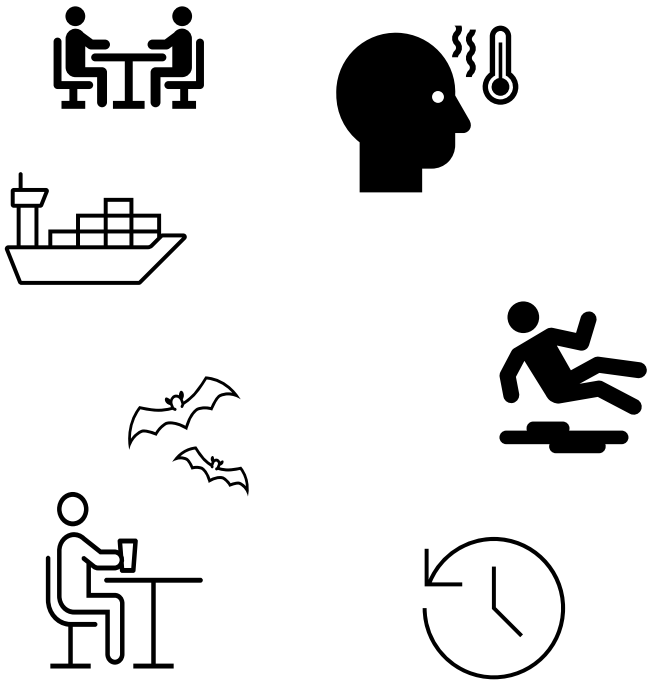
- Single-responsibility, composable, configurable, low coupled, /high cohesion

Process Overview Extensions for Product Line

- System Architecture Definition identifies
  - Common and varying functionality between the System Elements
  - How variations will be managed and configured
  - The subset of System Elements to build from this Product Line Platform
    - Constrained devices likely share no code with cloud services
    - Customer apps and internal test tool apps likely share substantial code
- Each Product Line Architecture identifies
  - Set of System Elements supported
  - Aggregate topology of components, component responsibilities, interfaces
  - Common configuration mechanism
  - Feature configurations for each supported System Element

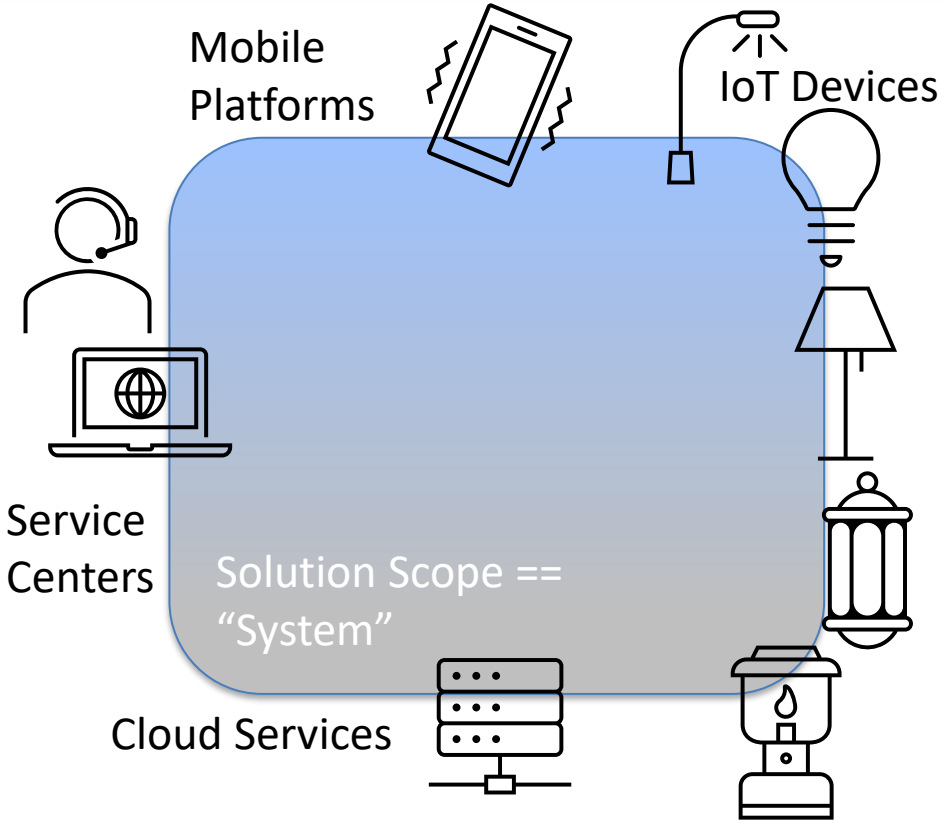
Since a picture is worth 1000 words …

Find the Problem Space – Stakeholder/User Needs



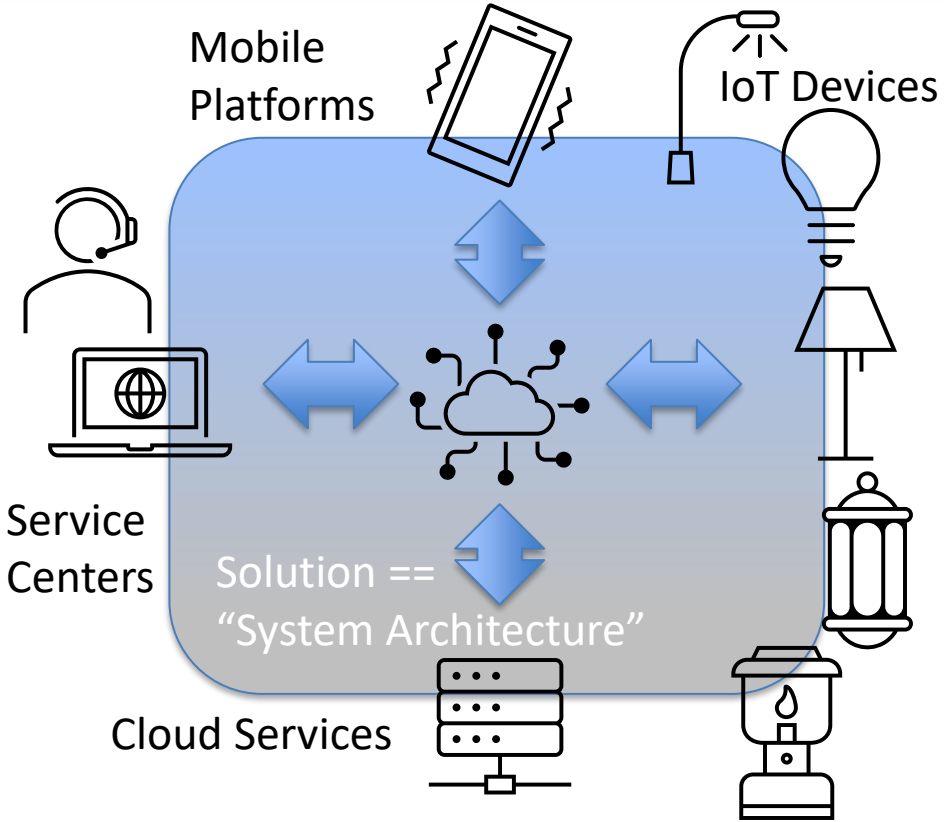
- Personas
- User Needs
- Business Needs
- Constraints
  - E.g., regulations, standards

Negotiate Your Solution Space – System Requirements



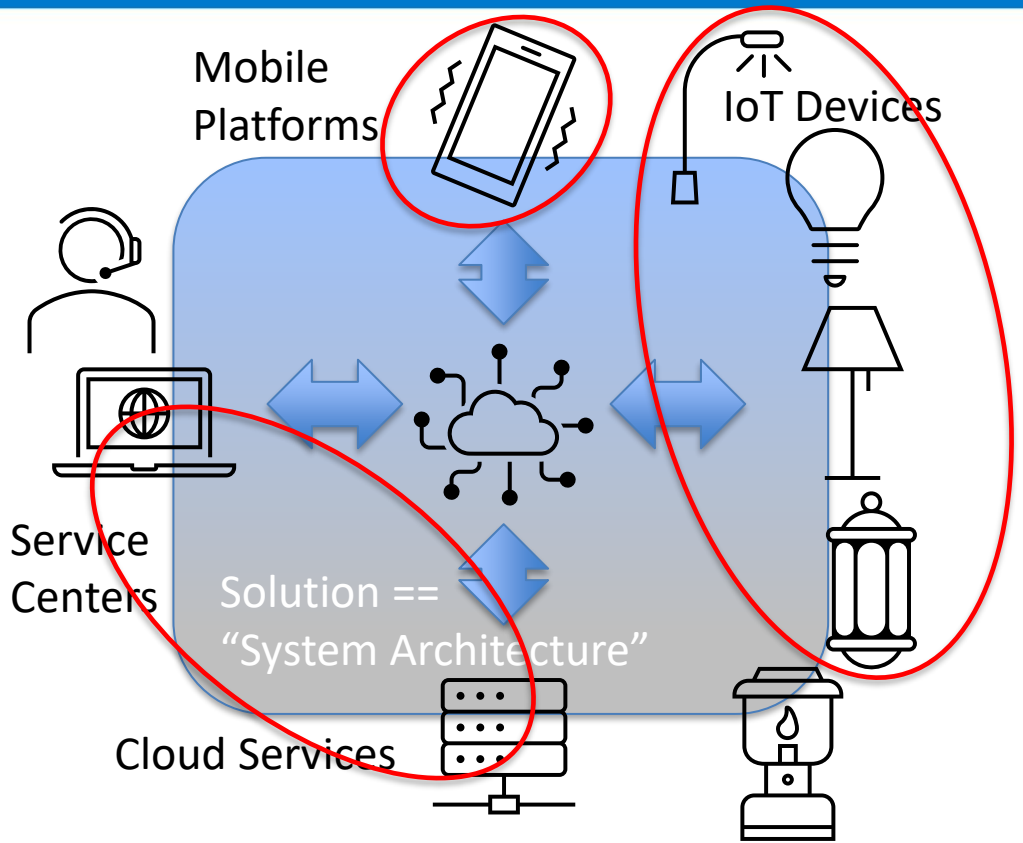
- Outside the system:
  - Everything you don't provide
- Inside the system:
  - Everything you do provide
    - E.g., devices, mobile apps, mfg tools, update services
  - What actions & behaviors your solution supports

Decide How Solution Will Work – System Architecture



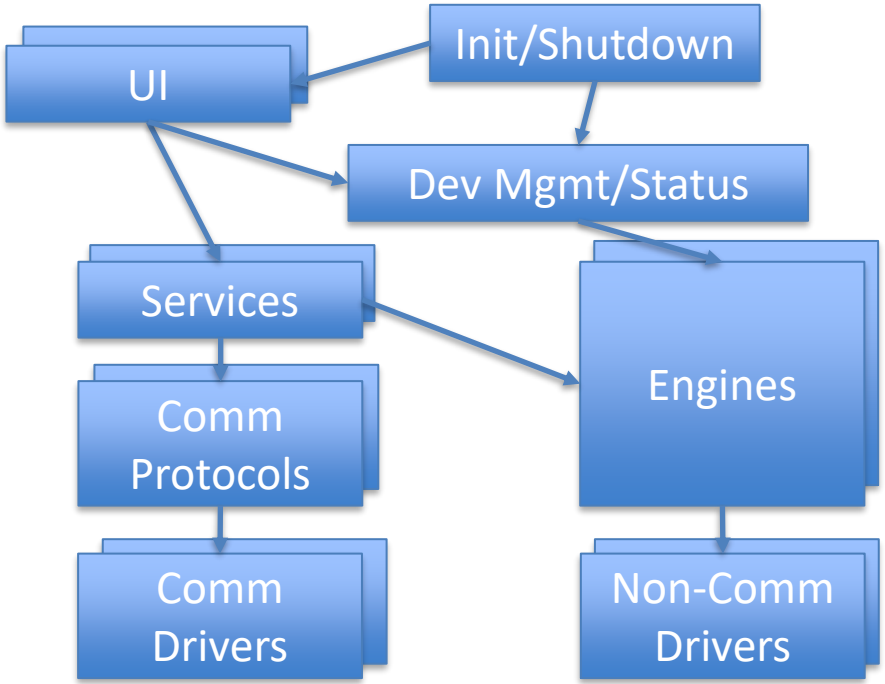
- Systems decompose into System Elements:
  - Specific devices, mobile apps, cloud services, ...
- Specify in detail:
  - External Interfaces to System
  - Responsibilities of each System Element
  - Interfaces between Elements

Find Commonalities and Variations – Product Line Scoping



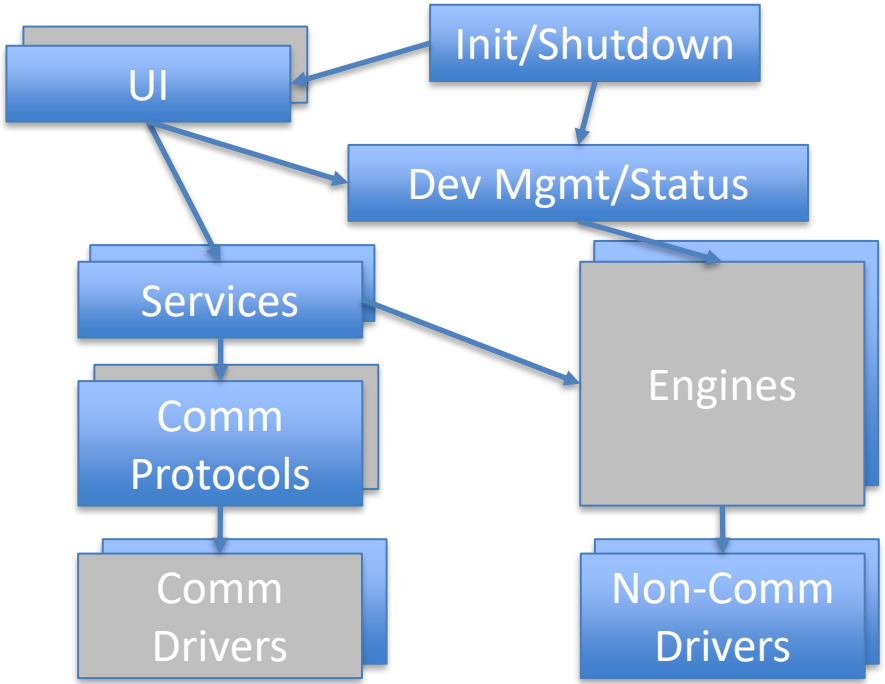
- Red ovals enclose potential product lines in this system.
- Opportunities for reuse in:
  - UX
  - Algorithms/Policies
  - Protocols
  - Services
  - Engines
  - Drivers

Design **Aggregate** Topology for PL Architecture to support similar System Elements



- Partitioned / configured **around variation points**
- Client->Server topology
  - Prioritized by tolerable latencies/dependency order
- Components:
  - Single responsibility
  - Depend on *interfaces*, not implementations
  - Thread-safe; often with own thread
- Variation in
  - UI, services, protocols, engines, drivers, …

Specify Configurations of System Elements



- "product" is just an architecture reduction based on:
  - A feature set configuration
  - Targeted at an execution env.
- Over time
  - almost nothing stays product-unique
  - the domain scope will change
  - product-specific optimizations just add complexity

Schedule Product Line Development & Marketing

Nothing different here.

Zephyr® Project
Developer Summit 2022

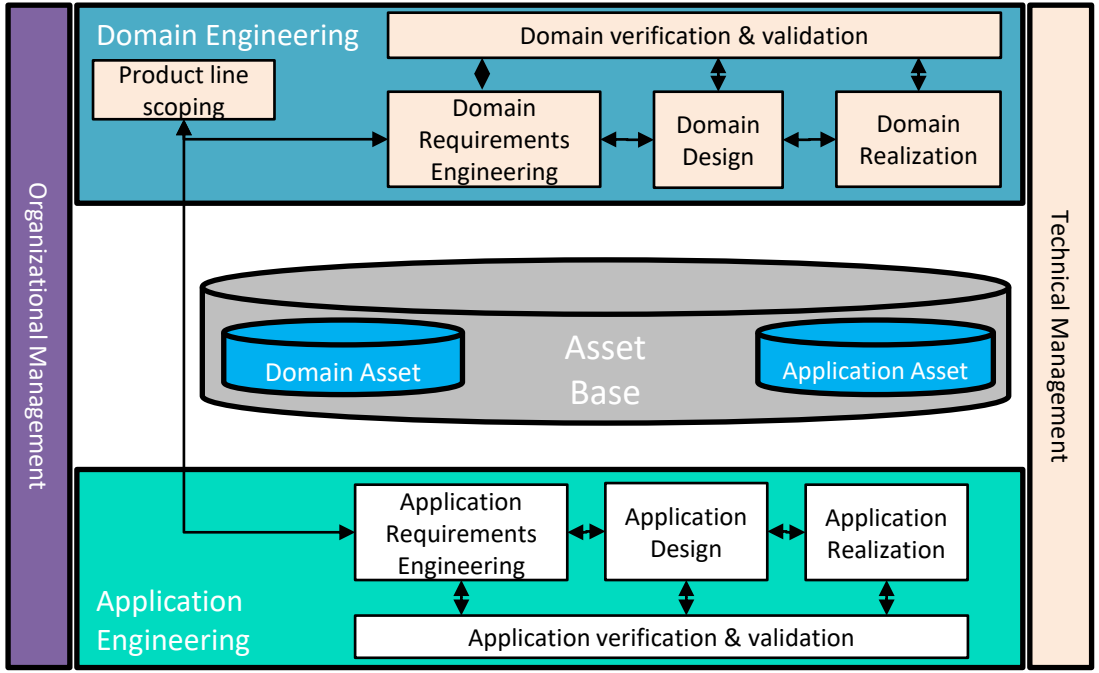
June 8-9, 2022
Mountain View, CA + Virtual

Application of PLE to Zephyr Ecosystem

Standard Product Line Models

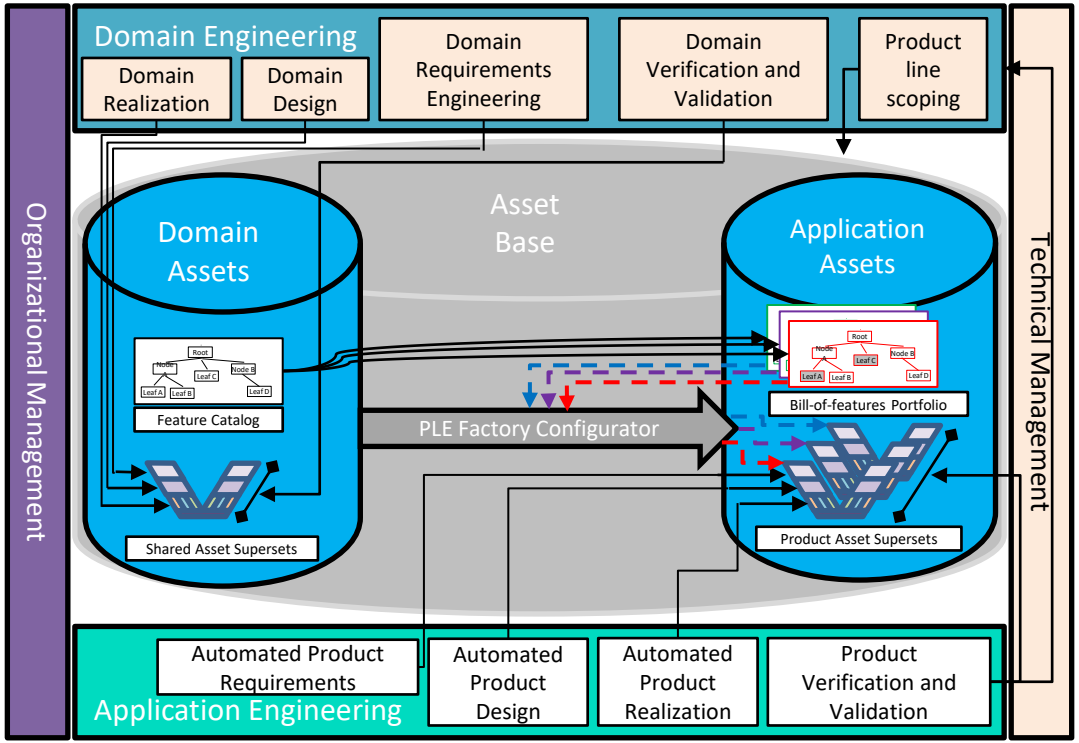
Avoid re-inventing the wheel

ISO/IEC 26550 – System and Software Product Line Engineering & Management



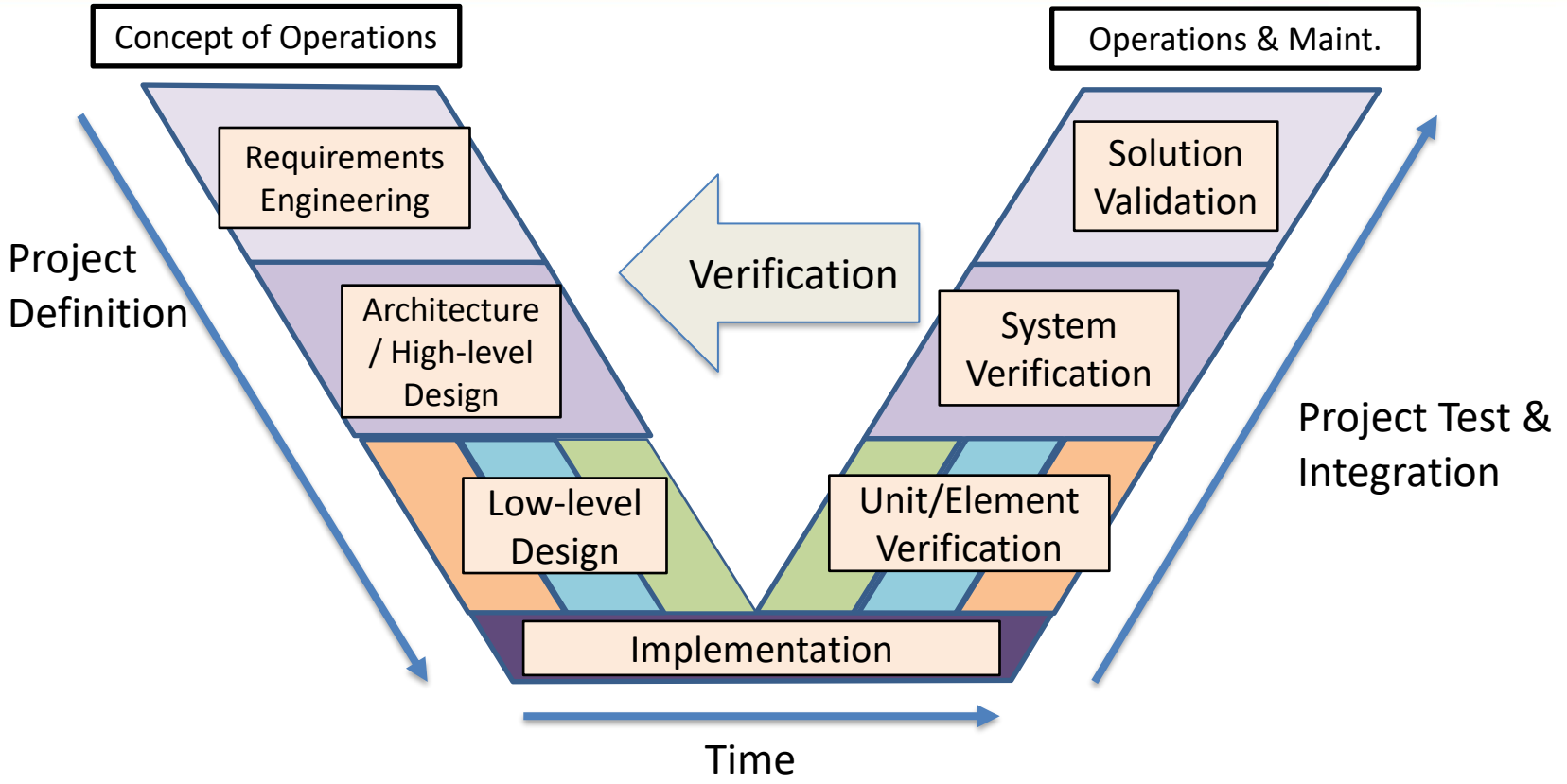
- Product line scoping:
  - Zephyr Project Mission Statement, Zephyr Technical Steering Committee
  - Downstream Zephyr user
- Domain Assets:
  - Build system, SBOM generation, Docs, License checks
  - Requirements, Modules, Kconfigs, Subsystems, Drivers, Boards
- Domain Verification:
  - Tests, Test runners, code coverage
- Application Assets:
  - example-application, samples
  - Downstream Zephyr user modules

ISO/IEC 26580 – Feature-based Specialization of ISO/IEC 26550



- Feature Catalog:
  - est manifest + Kconfig settings
- Shared Asset Supersets:
  - github.com/zephyrproject-rtos/
- Bill-of-Features:
  - prj.conf + overlays
- PLE Factory Configurator:
  - build system, doc generation, SBOM generation, licensing checks

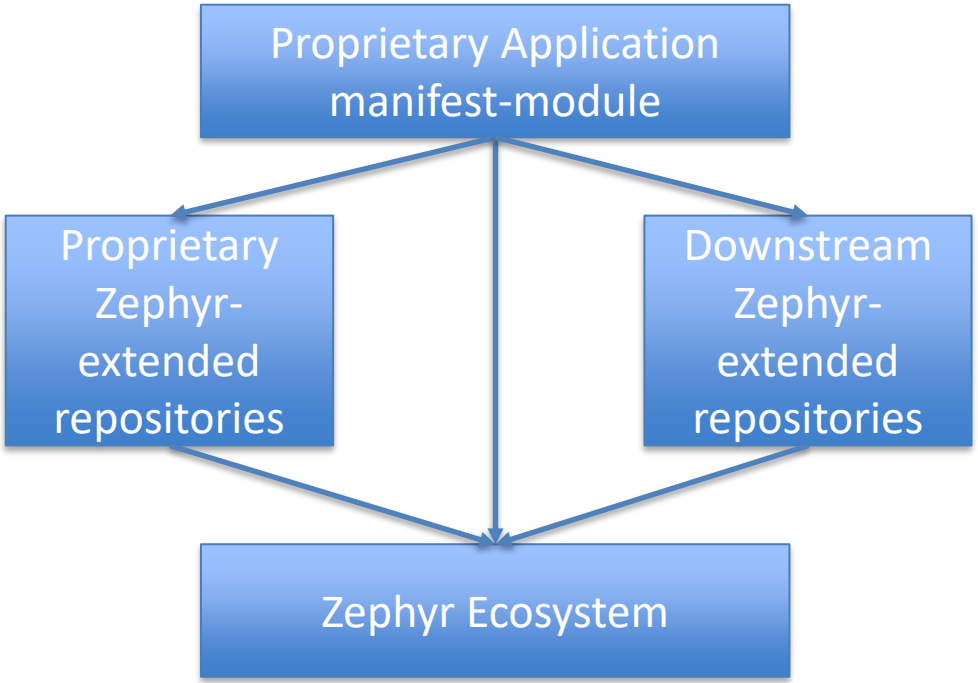
"Vee" Model of Systems Engineering Process



Guidance for All

- Re-use when possible, extend when necessary.
  - Follow the “open-closed principle”: closed for modification, open for extension.
- Design for reuse, security, composability, configurability, verifiability.
  - Minimize coupling
  - Maximize cohesion (keep interdependent/very closely related things together)
  - Components and modules will be re-used in unforeseen products.
  - Zephyr mainline will need to sustain the processes needed for the auditable branch – and for some downstream users.
  - Only the end users know enough about the system to optimize it.

Guidance for Users



- Use an application manifest-module
- Reuse:
  - Compose configurations from shared feature-set overlays
  - Assume no code is application-unique
- Partition into modules based on
  - Upstream repos
  - Licensing

Cautions to Users through Zephyr v3.1.0

- Safety & Security quality processes depend on requirement traceability. Nothing is in place yet.
- Extensibility mechanisms not continuously verified
- Numerous issues still inhibit reuse:
  - Namespace management
  - cooperative scheduling
  - default thread priorities
  - common work queue
  - ISRs restricted to interrupt context

Summary

Summary

- ✓ Product Line Terminology
  - No need to invent proprietary terms
- ✓ Product Line Engineering (PLE) Process
  - Maps well to standard processes – can go farther
- ✓ Application of PLE to Zephyr Ecosystem
  - Achievable now – with room for improvement

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Q & A