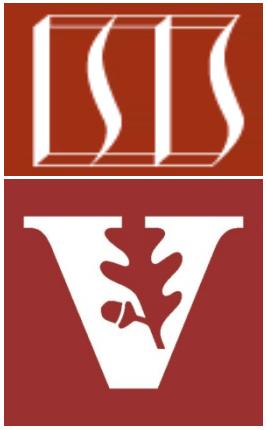
Framework Examples: Part 1

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt



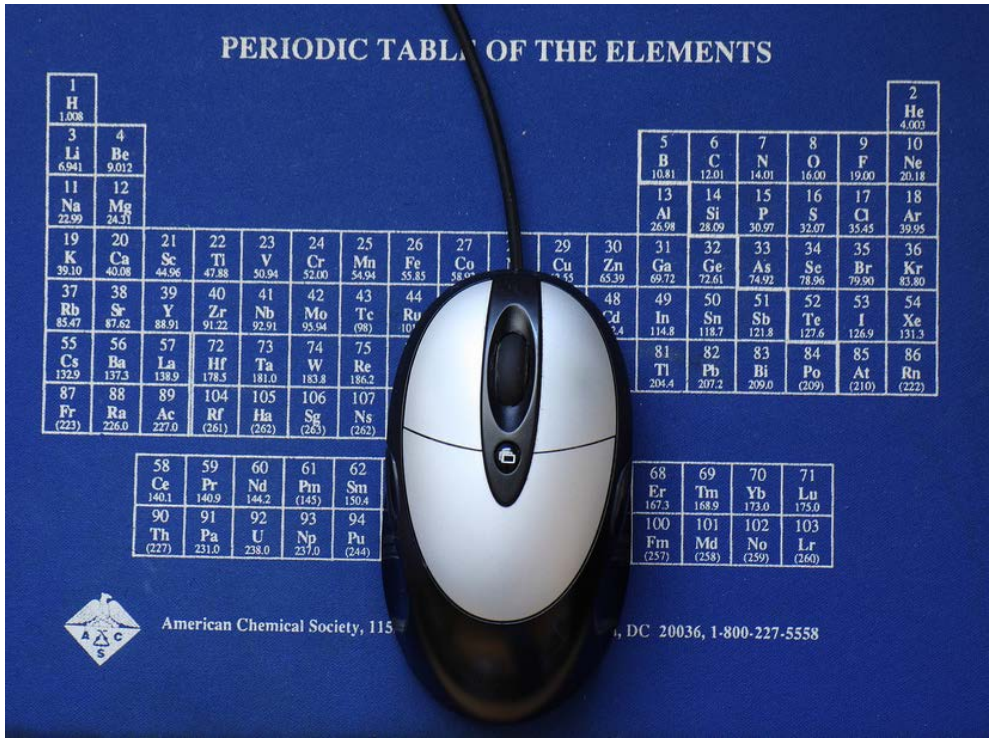
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

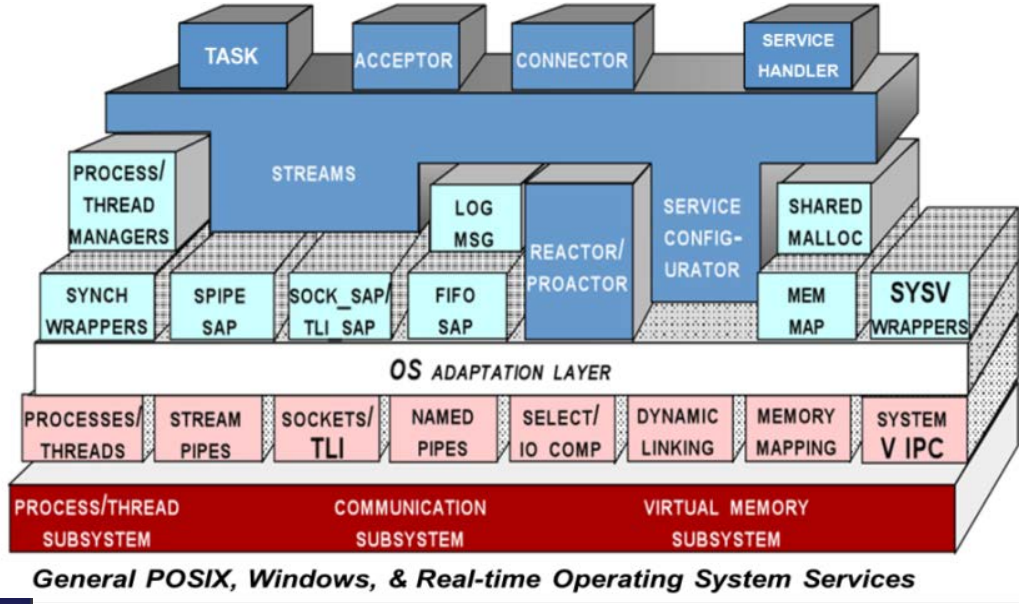
Topics Covered in this Part of the Module

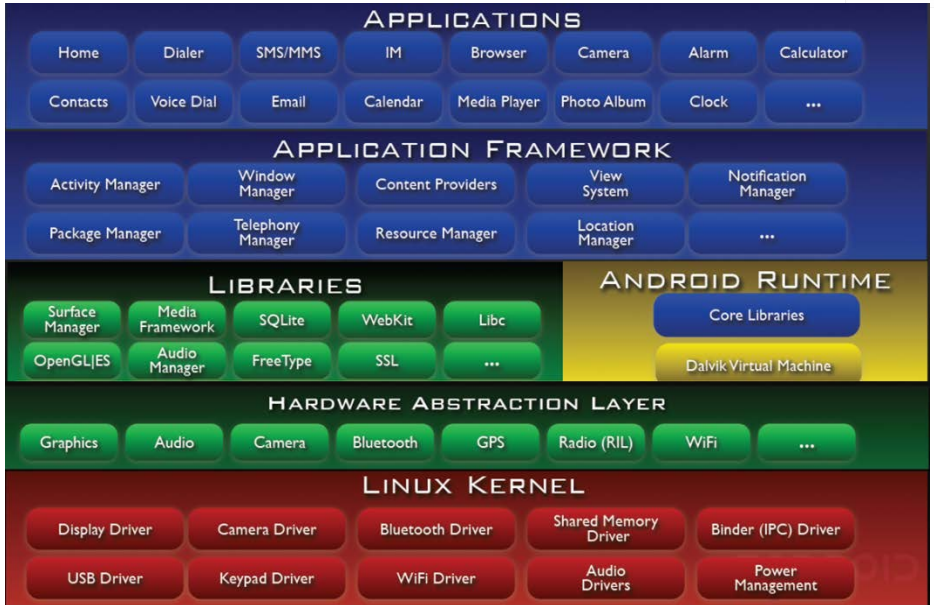
- Present *Scope, Commonality, & Variability* (SCV) analysis as a method for developing & applying software product-lines & frameworks



Topics Covered in this Part of the Module

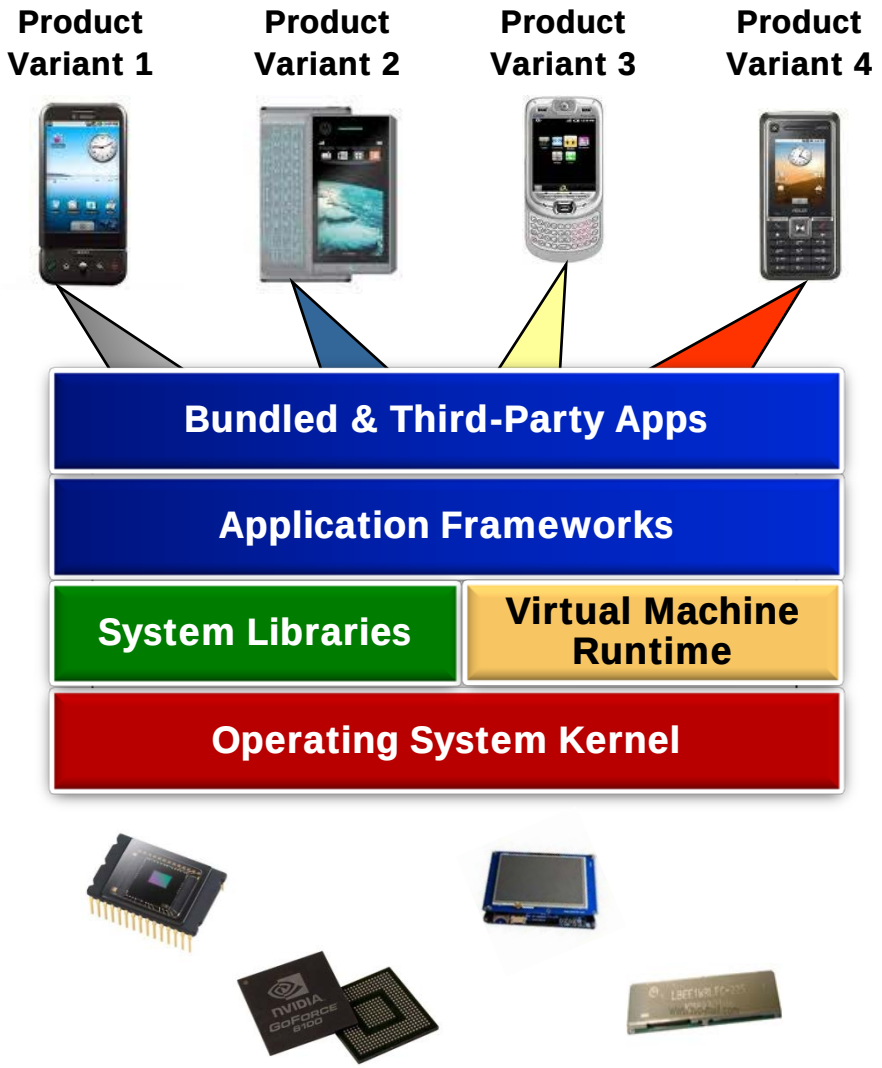
- Present *Scope, Commonality, & Variability* (SCV) analysis as a method for developing & applying software product-lines & frameworks
- Illustrate the application of SCV to Android & ACE





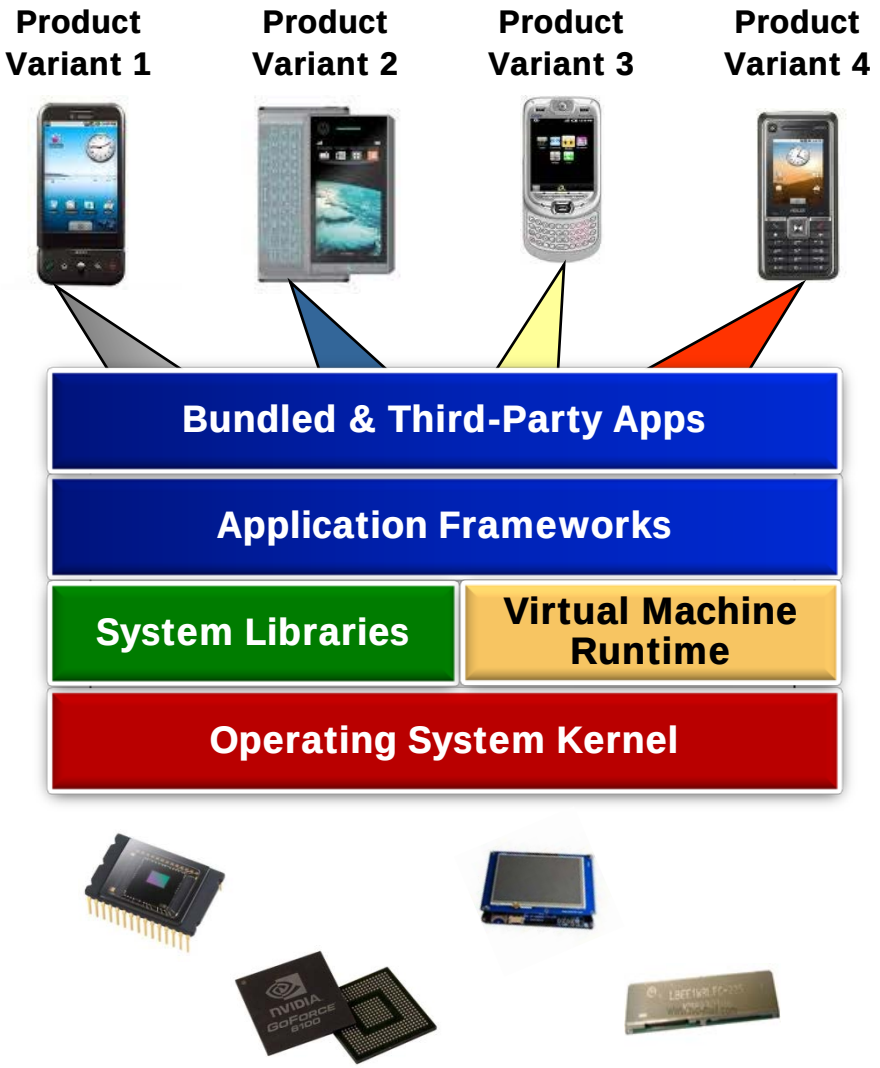
Overview of Software Product-Lines

- A *software product line* (SPL) is a form of systematic software reuse
  - An SPL a set of software-intensive systems
  - These systems share a common, managed set of features satisfying the specific needs of a particular market segment or mission
  - They are developed from a common set of core assets in a prescribed way



Overview of Software Product-Lines

- A *software product line* (SPL) is a form of systematic software reuse
  - An SPL a set of software-intensive systems
  - These systems share a common, managed set of features satisfying the specific needs of a particular market segment or mission
  - They are developed from a common set of core assets in a prescribed way
- Frameworks (& patterns) can help define & improve core SPL assets by factoring out many reusable general-purpose & domain-specific services from application responsibility



See www.sei.cmu.edu/productlines for more info on software product-lines

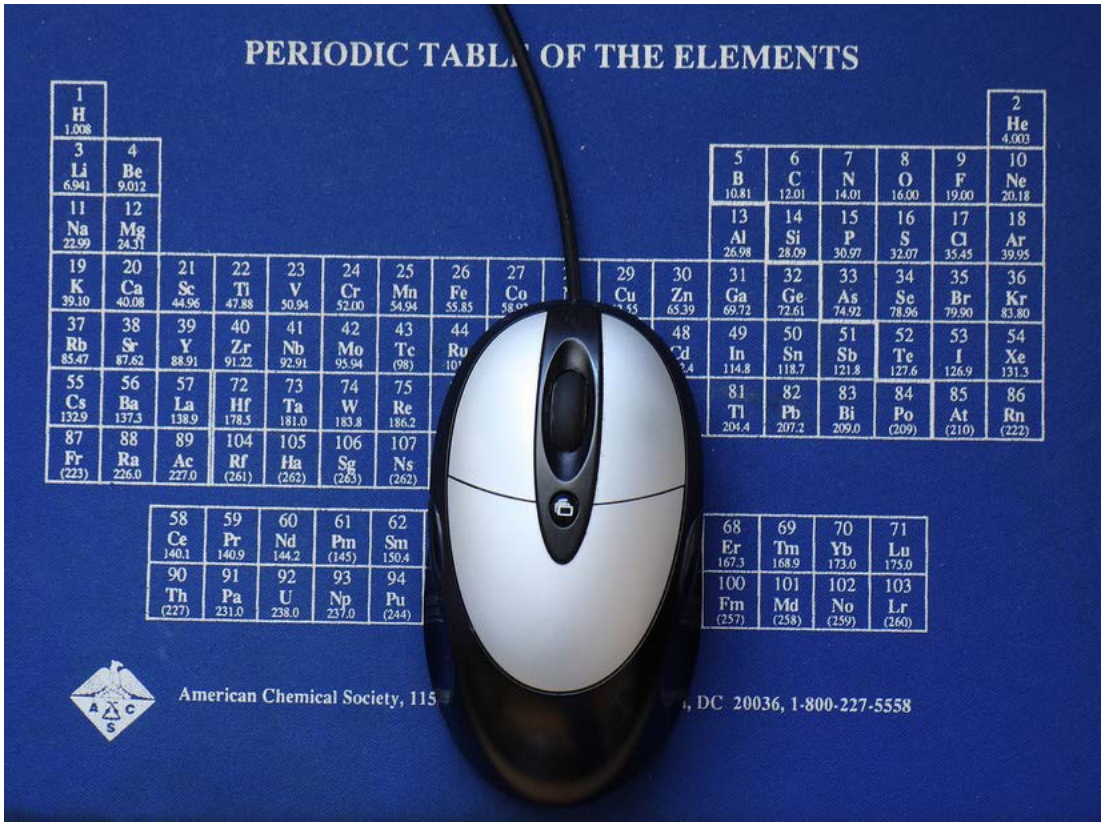
Scope, Commonality, & Variability Analysis

- Key software product-line & framework structure & behavior can be captured systematically via *Scope*, *Commonality*, & *Variability* (SCV) analysis

Scope, Commonality, & Variability Analysis

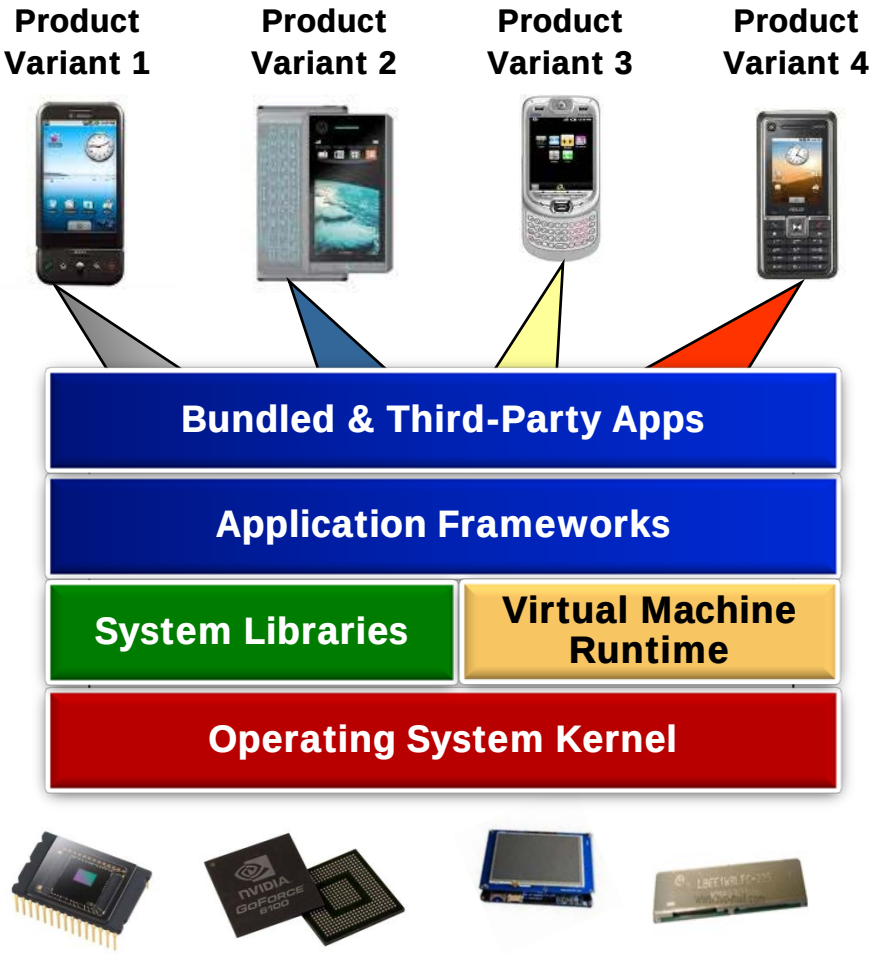
- Key software product-line & framework structure & behavior can be captured systematically via *Scope*, *Commonality*, & *Variability* (SCV) analysis

- This process can be applied to identify commonalities & variabilities in a domain



Scope, Commonality, & Variability Analysis

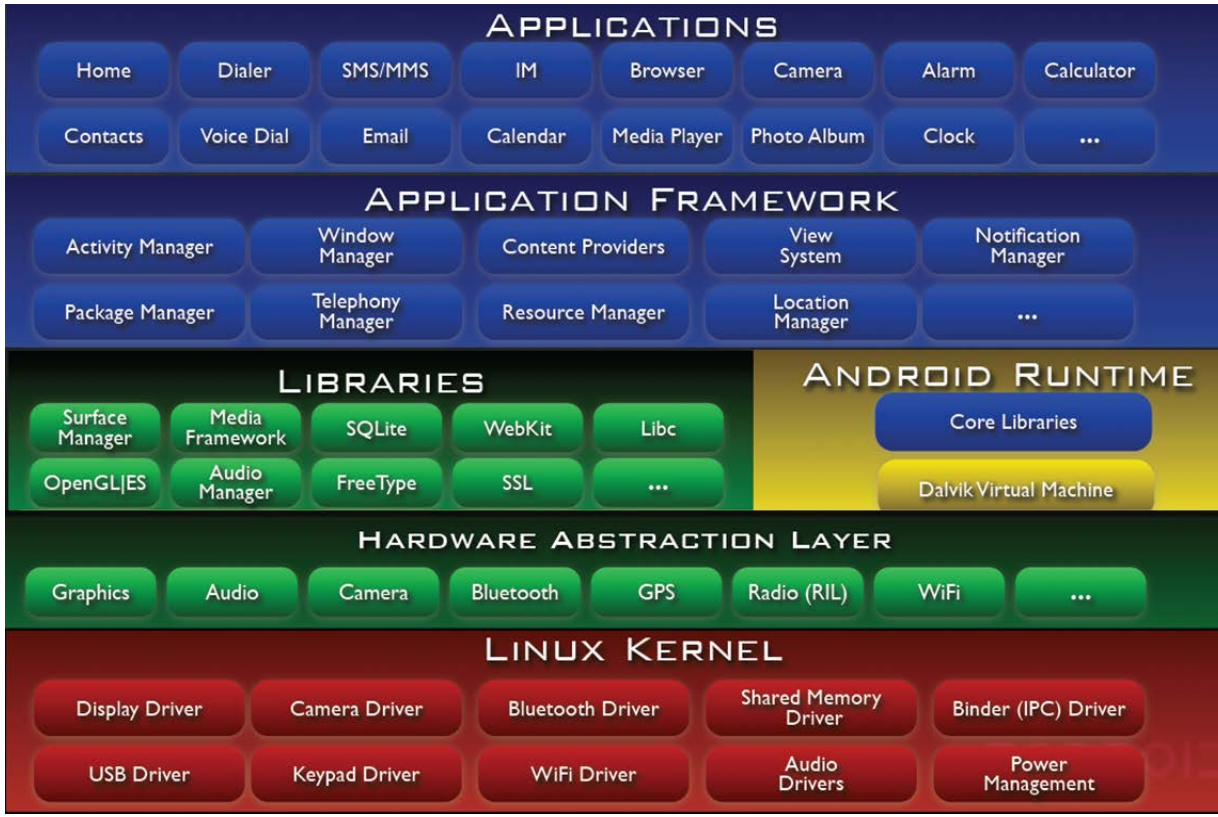
- Key software product-line & framework structure & behavior can be captured systematically via *Scope*, *Commonality*, & *Variability* (SCV) analysis
- This process can be applied to identify commonalities & variabilities in a domain
- This analysis can guide the development & application of software product-lines & frameworks



www.cs.iastate.edu/~cs309/references/CoplienHoffmanWeiss_CommonalityVariability.pdf

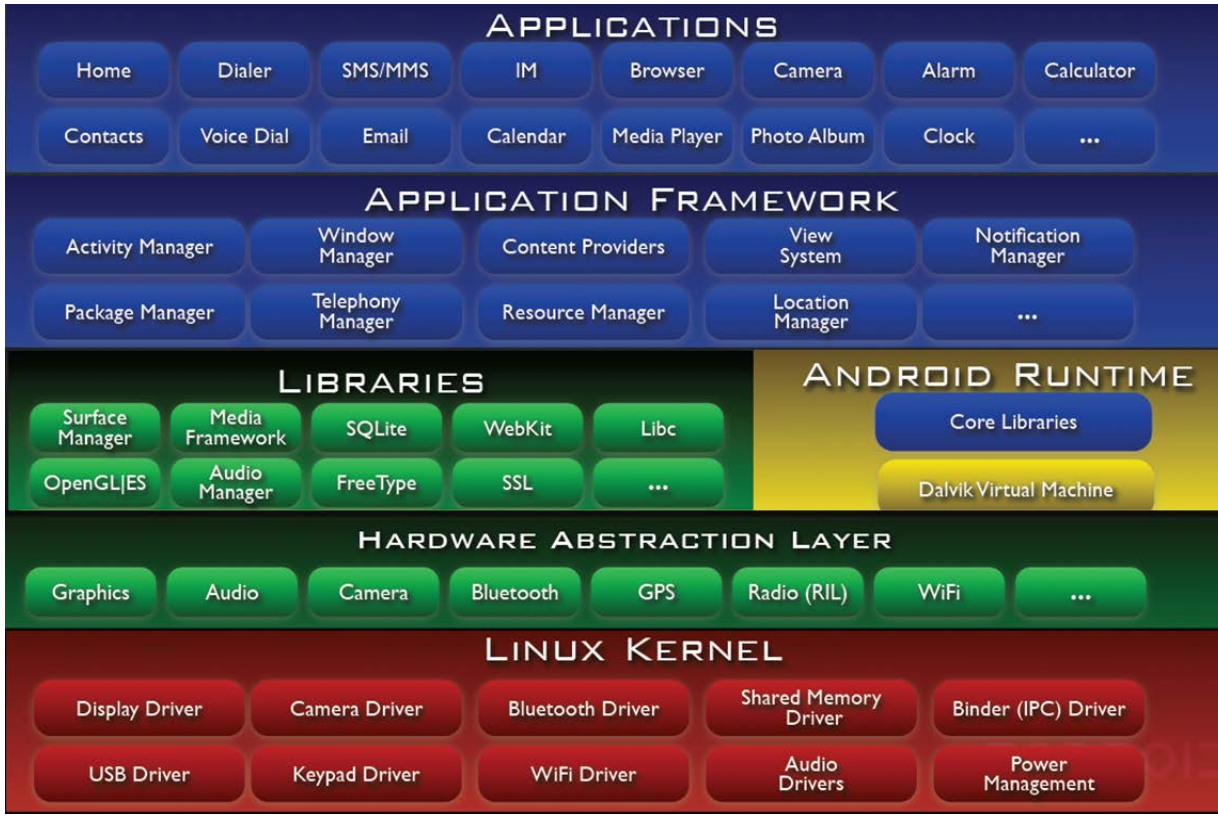
Applying SCV to Android

- **Scope** defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points



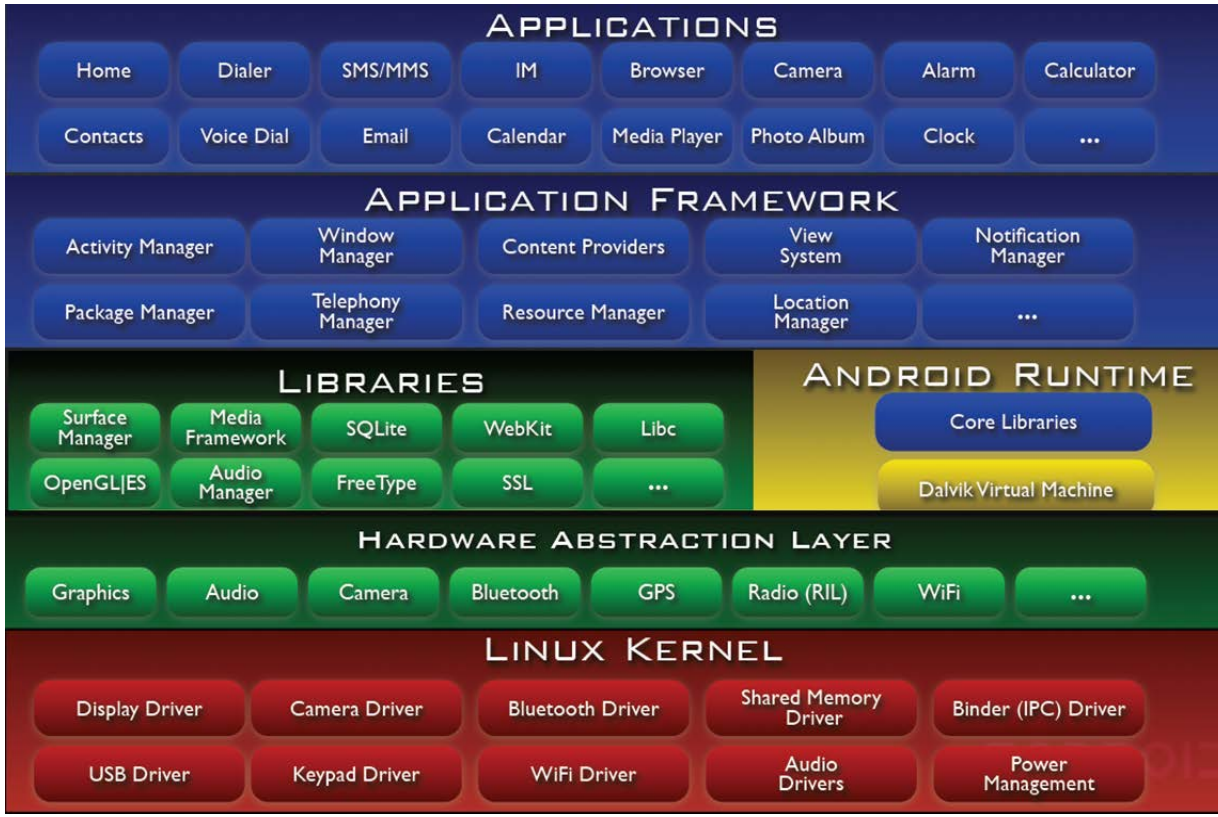
Applying SCV to Android

- **Scope** defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points
  - Touch-based user interfaces



Applying SCV to Android

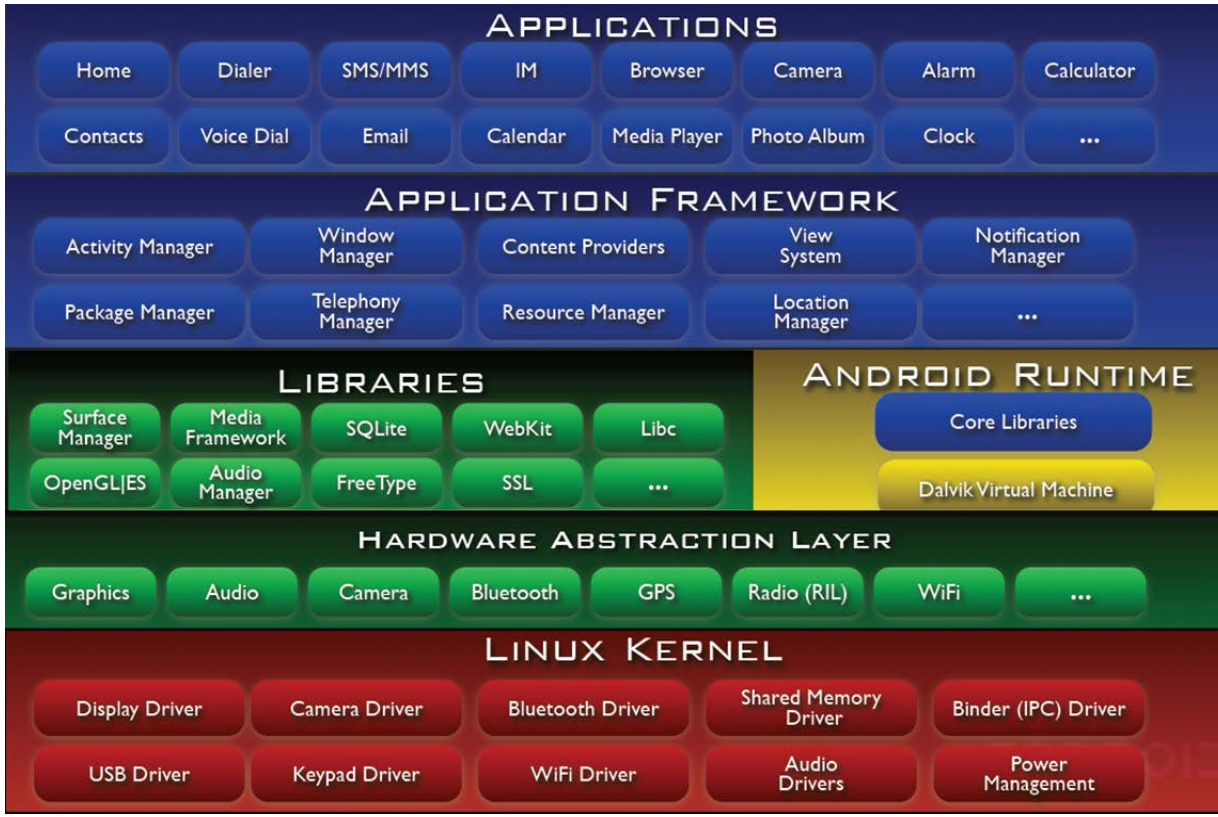
- **Scope** defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points
  - Touch-based user interfaces



- (Largely) open-source, vendor- & hardware-agnostic ecosystem

Applying SCV to Android

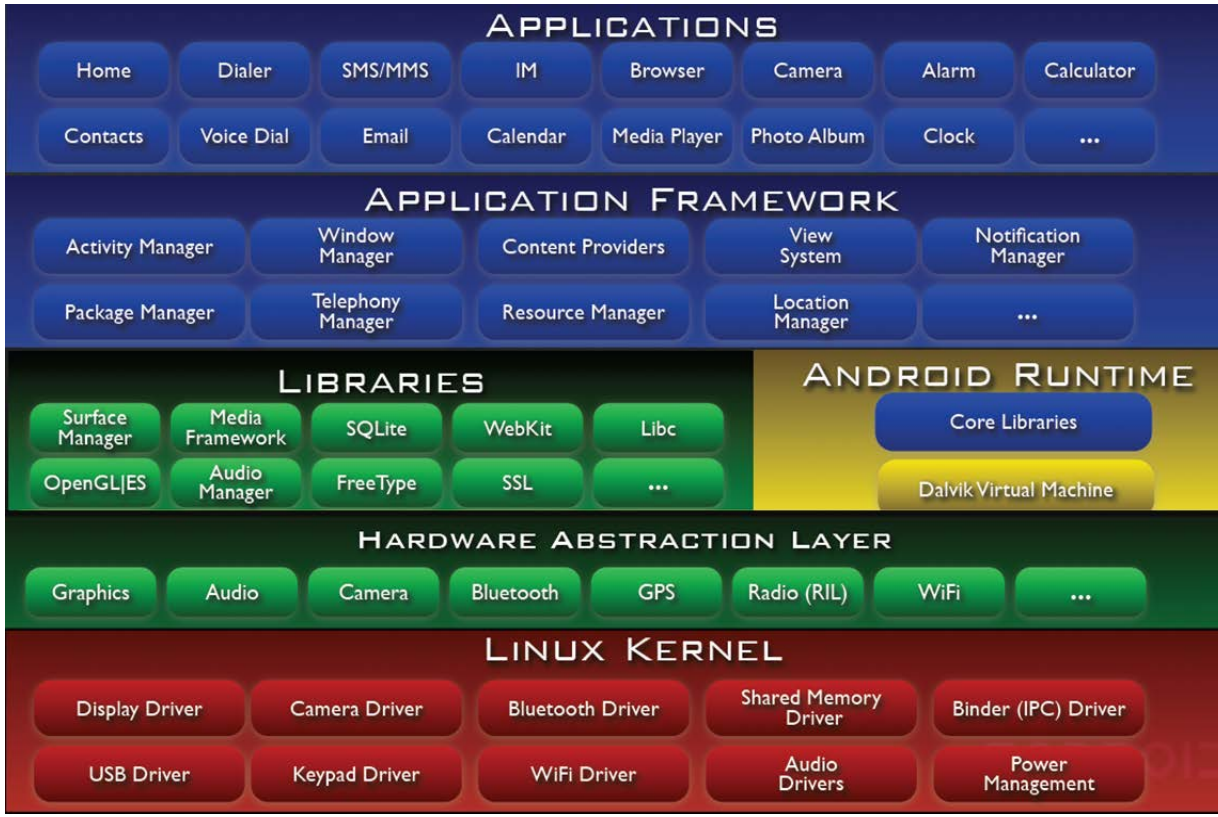
- **Scope** defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points
  - Touch-based user interfaces
  - (Largely) open-source, vendor- & hardware-agnostic ecosystem
  - Focus on installed-base of Java app developers



See developer.android.com for more info on Android

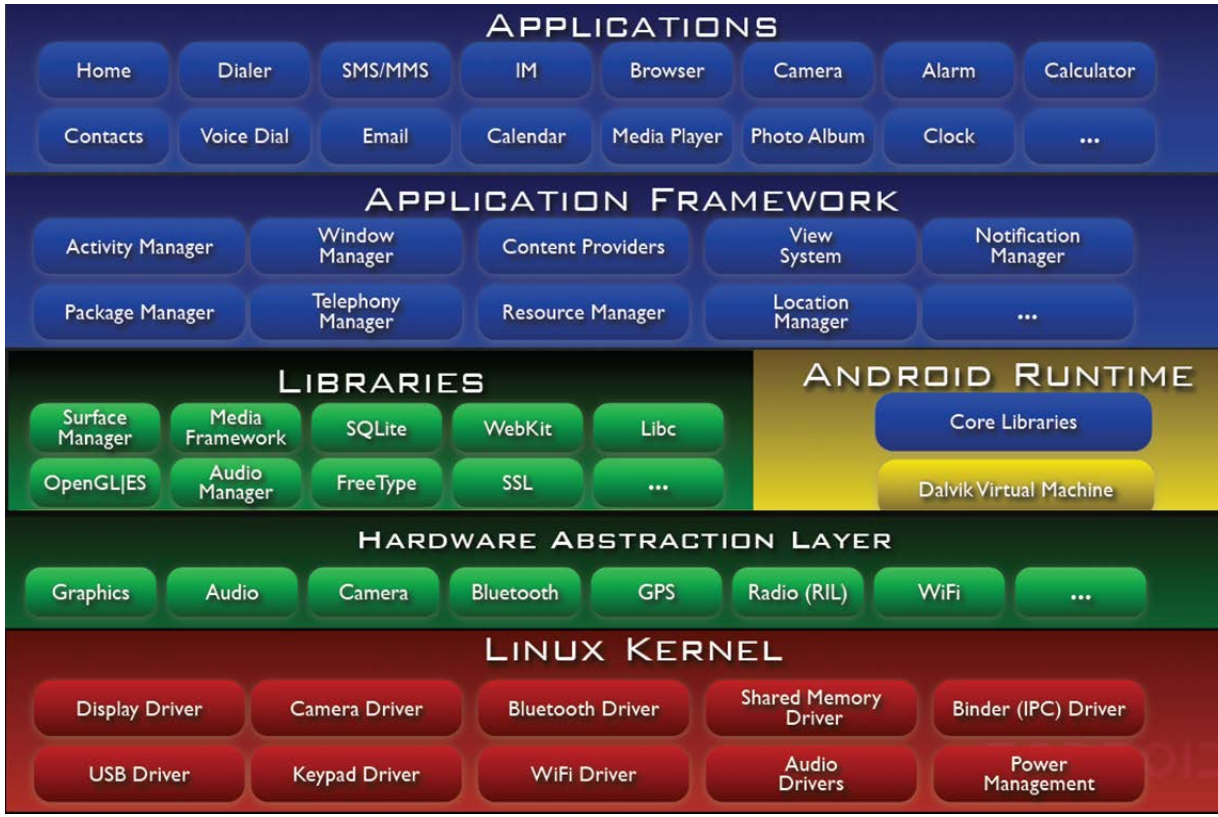
Applying SCV to Android

- **Commonalities** describe the attributes common across all instances of Android
  - *Common framework components*
    - e.g., Activities, Services, Content Providers, & Broadcast Receivers



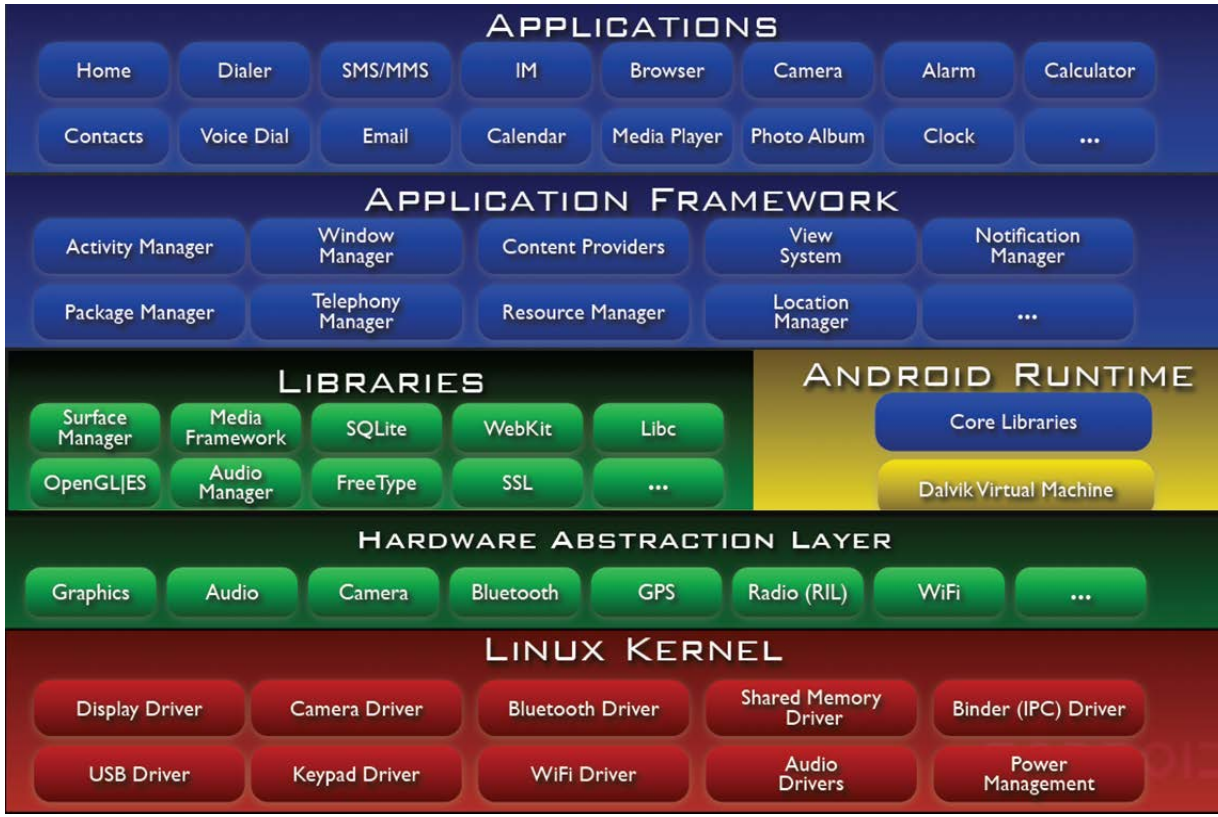
Applying SCV to Android

- **Commonalities** describe the attributes common across all instances of Android
  - *Common framework components*
    - e.g., Activities, Services, Content Providers, & Broadcast Receivers
  - *Common application frameworks*
    - e.g., Activity Manager, Package Manager, Telephony Manager, Location Manager, Notification Manager, etc.



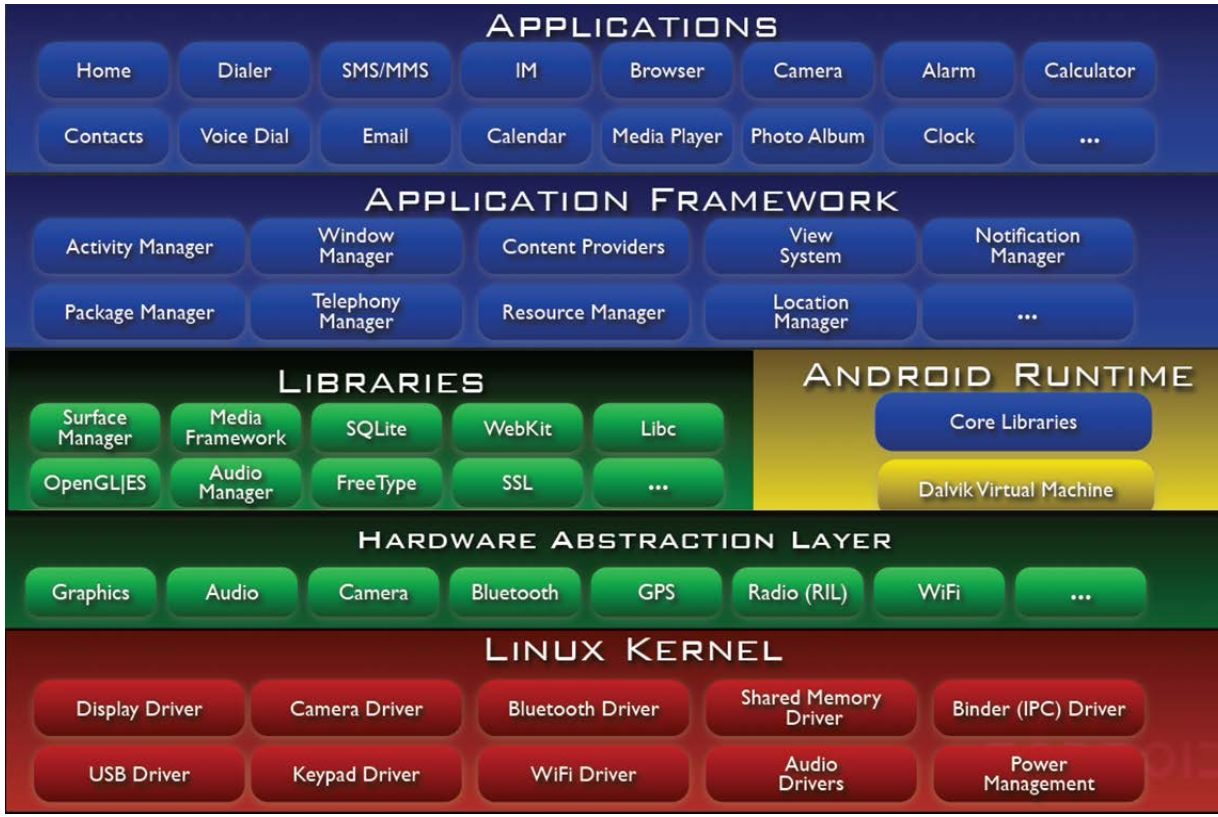
Applying SCV to Android

- **Commonalities** describe the attributes common across all instances of Android
  - *Common framework components*
    - e.g., Activities, Services, Content Providers, & Broadcast Receivers
  - *Common application frameworks*
    - e.g., Activity Manager, Package Manager, Telephony Manager, Location Manager, Notification Manager, etc.
  - *Common infrastructure*
    - e.g., Intent framework, Binder, Webkit, Hardware Abstraction Layer, OS device driver frameworks etc.



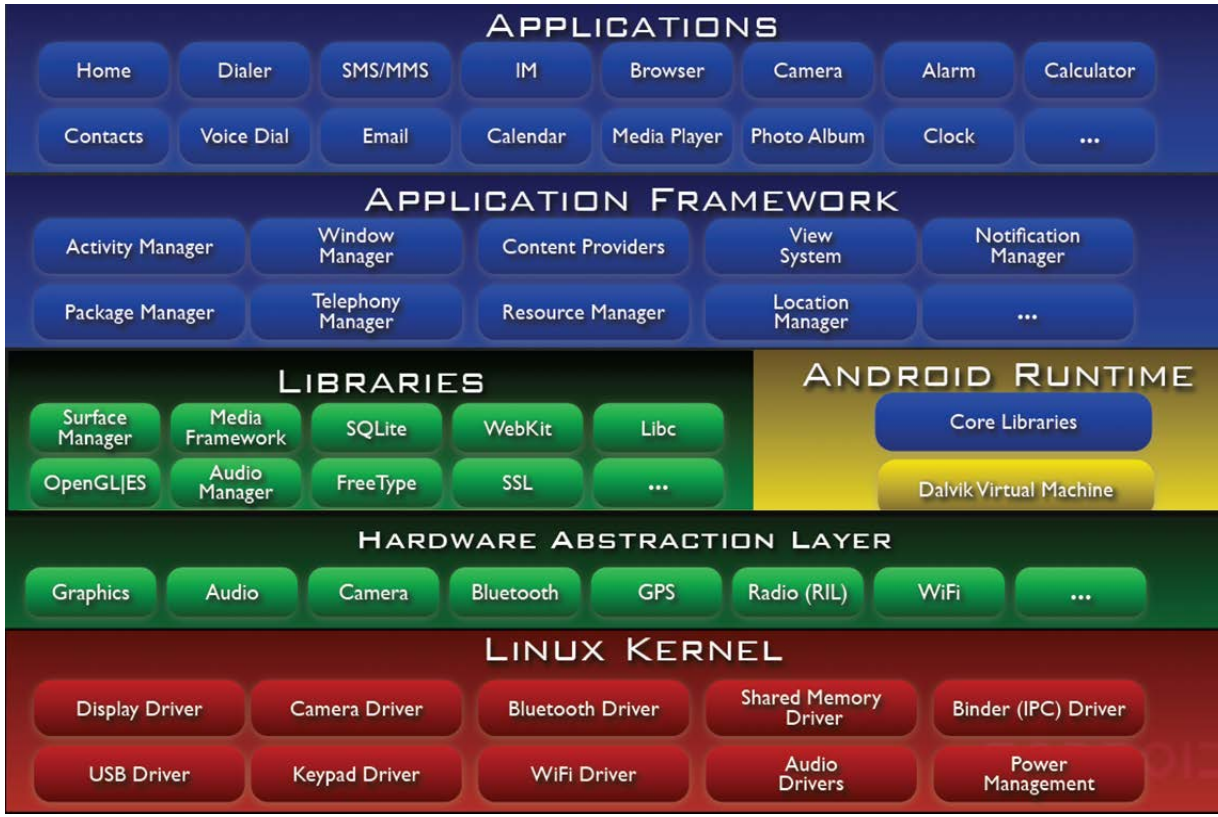
Applying SCV to Android

- **Variabilities** describe the attributes unique to different instantiations of Android
  - *Product-dependent components*
    - e.g., different "look & feel" variants of vendor-specific user interfaces, sensor & device properties, etc.



Applying SCV to Android

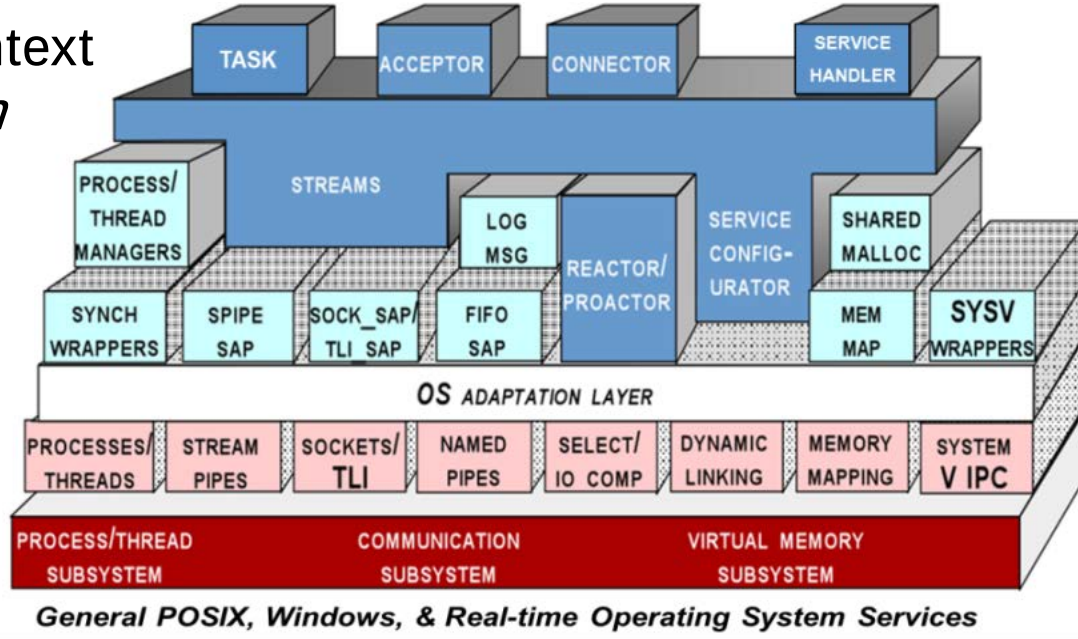
- **Variabilities** describe the attributes unique to different instantiations of Android
  - *Product-dependent components*
    - e.g., different "look & feel" variants of vendor-specific user interfaces, sensor & device properties, etc.
  - *Product-dependent component assemblies*
    - e.g., different bundled apps, CDMA vs. GSM & different hardware, OS, & network/bus configurations, etc.



SCV can also be applied recursively for all the Android frameworks & layers

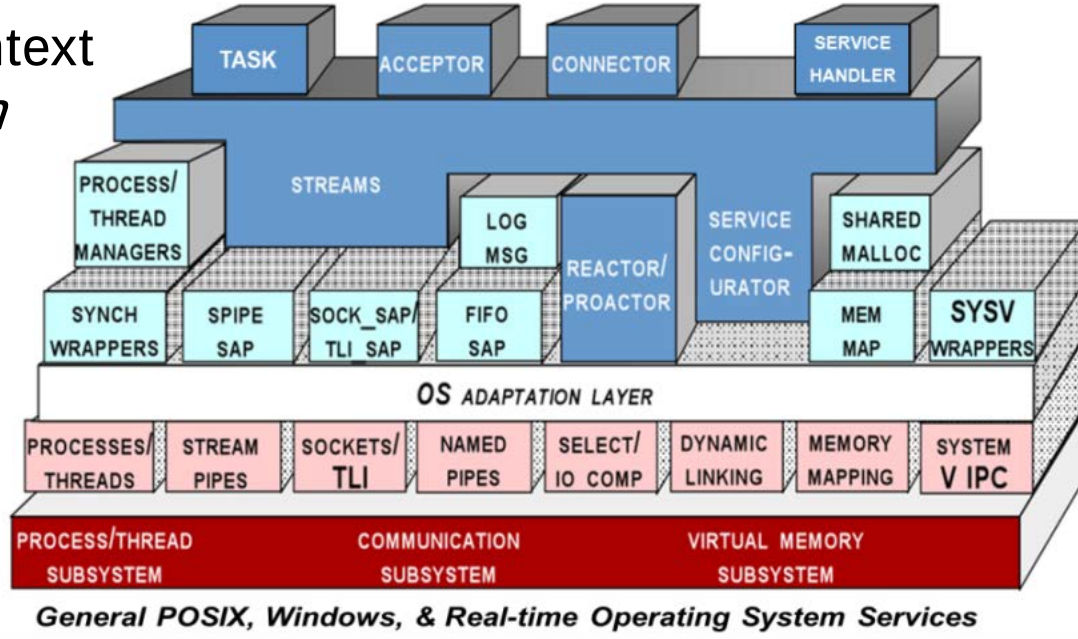
Applying SCV Analysis to ACE

- **Scope** defines the domain & context of the *ADAPTIVE Communication Environment* (ACE)
- e.g.,
  - Object-oriented host infrastructure middleware
    - *Encapsulates* many tedious & error-prone aspects of low-level OS APIs for concurrent & networked software
    - *Enhances* native OS mechanisms to create reusable C++ components



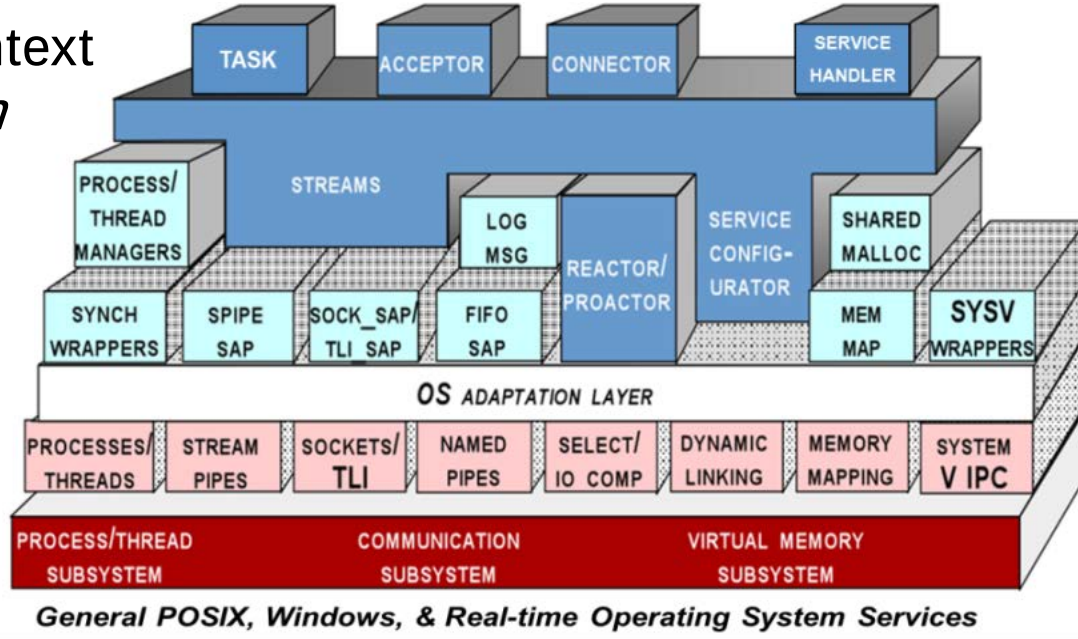
Applying SCV Analysis to ACE

- **Scope** defines the domain & context of the *ADAPTIVE Communication Environment* (ACE)
- e.g.,
  - Object-oriented host infrastructure middleware
    - *Encapsulates* many tedious & error-prone aspects of low-level OS APIs for concurrent & networked software
    - *Enhances* native OS mechanisms to create reusable C++ components
  - Open-source vendor- & OS-agnostic ecosystem



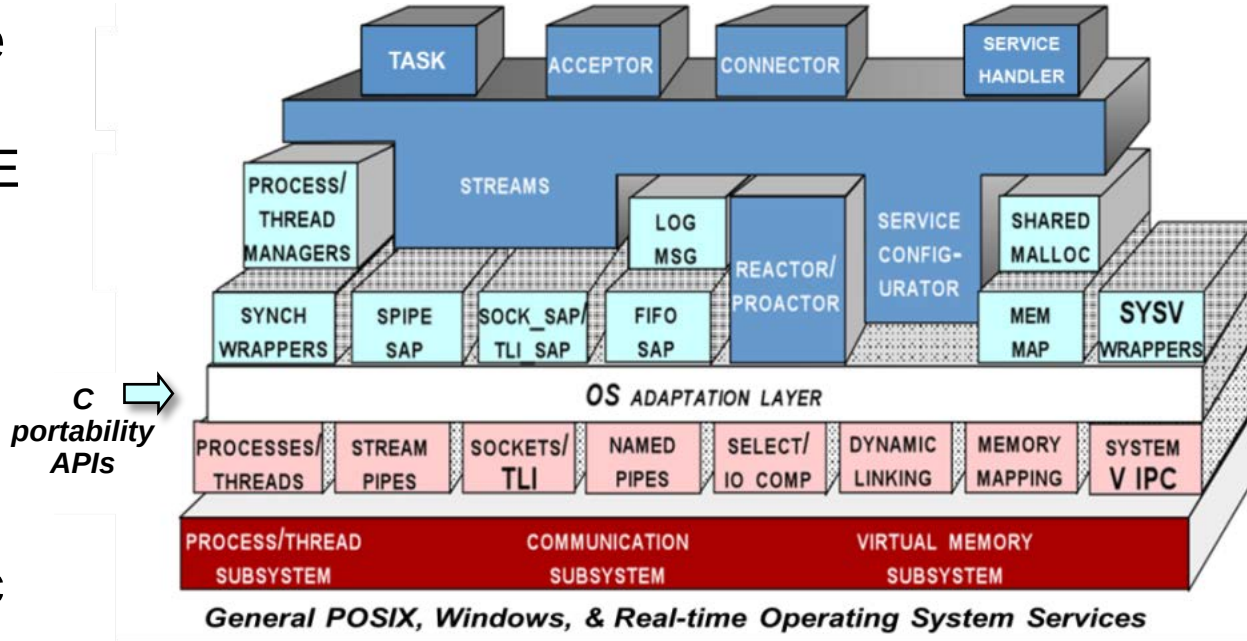
Applying SCV Analysis to ACE

- **Scope** defines the domain & context of the *ADAPTIVE Communication Environment* (ACE)
- e.g.,
  - Object-oriented host infrastructure middleware
    - *Encapsulates* many tedious & error-prone aspects of low-level OS APIs for concurrent & networked software
    - *Enhances* native OS mechanisms to create reusable C++ components
  - Open-source vendor- & OS-agnostic ecosystem
  - Focused on C++ software developers, particularly those developing infrastructure software & distributed real-time & embedded (DRE) systems



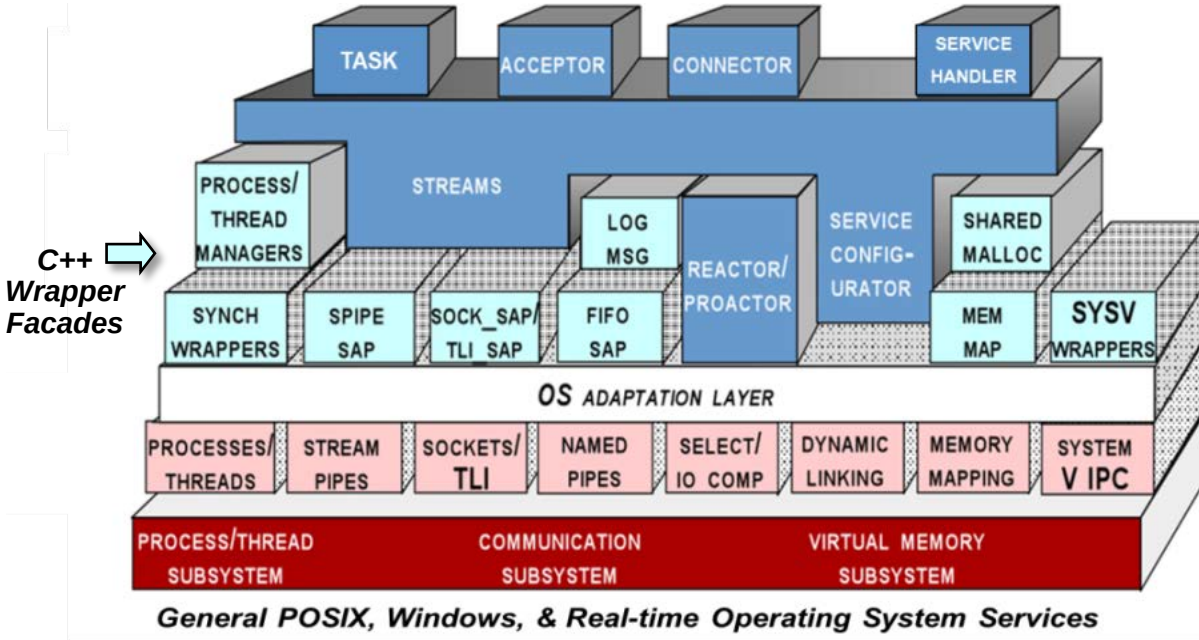
Applying SCV Analysis to ACE

- **Commonalities** describe the attributes common across all instances of ACE
  - *C portability APIs*
    - "POSIX-like" OS adaptation layer that shields other layers & components in ACE from platform-specific dependencies



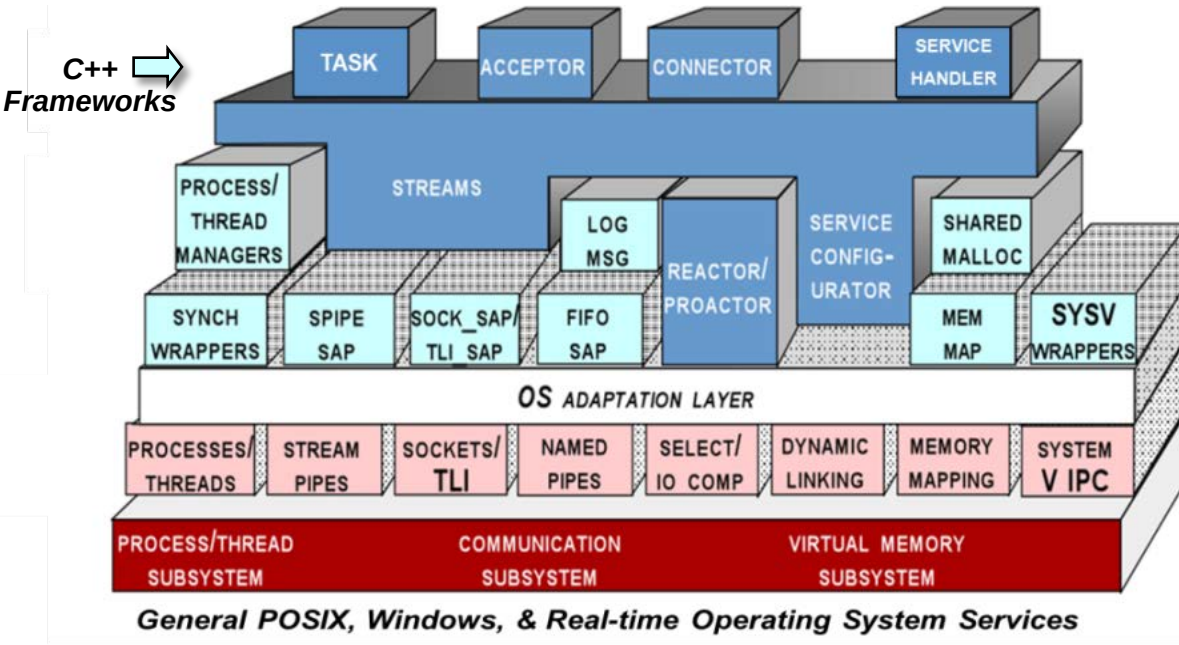
Applying SCV Analysis to ACE

- **Commonalities** describe the attributes common across all instances of ACE
  - *C portability APIs*
    - "POSIX-like" OS adaptation layer that shields other layers & components in ACE from platform-specific dependencies
  - *C++ wrapper facades*
    - Provides type-safe C++ interfaces that encapsulate native OS concurrency, communication, memory management, event demultiplexing, dynamic linking, & file system APIs



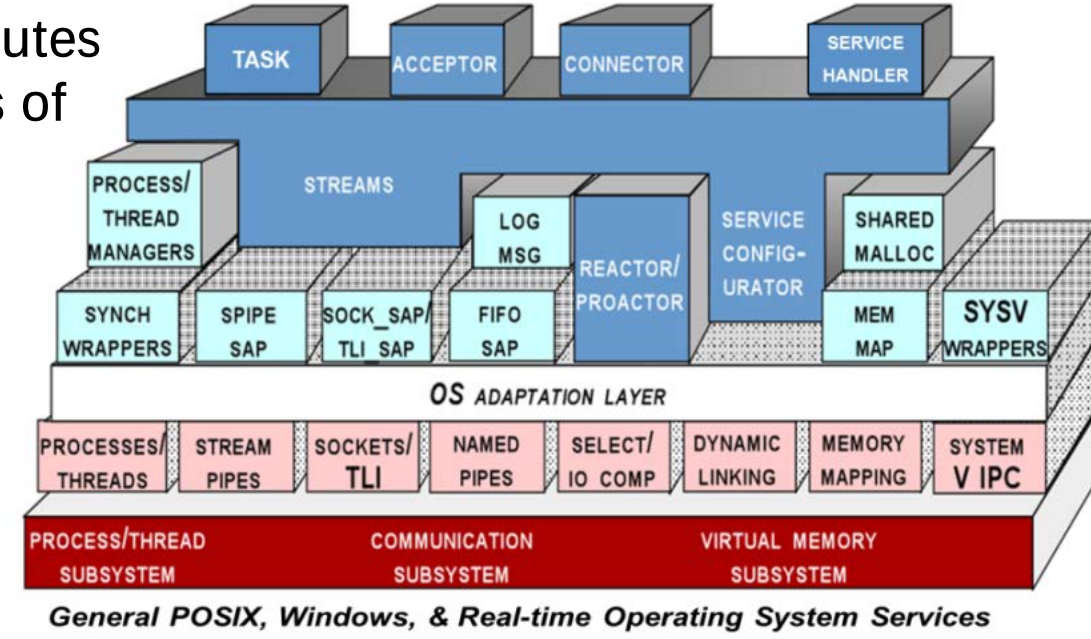
Applying SCV Analysis to ACE

- **Commonalities** describe the attributes common across all instances of ACE
  - *C portability APIs*
    - "POSIX-like" OS adaptation layer that shields other layers & components in ACE from platform-specific dependencies
  - *C++ wrapper facades*
    - Provides type-safe C++ interfaces that encapsulate native OS concurrency, communication, memory management, event demultiplexing, dynamic linking, & file system APIs
  - *C++ frameworks*
    - e.g., Sync/async event handling, connection setup & service initialization, dynamic configuration, concurrency, & layered service composition



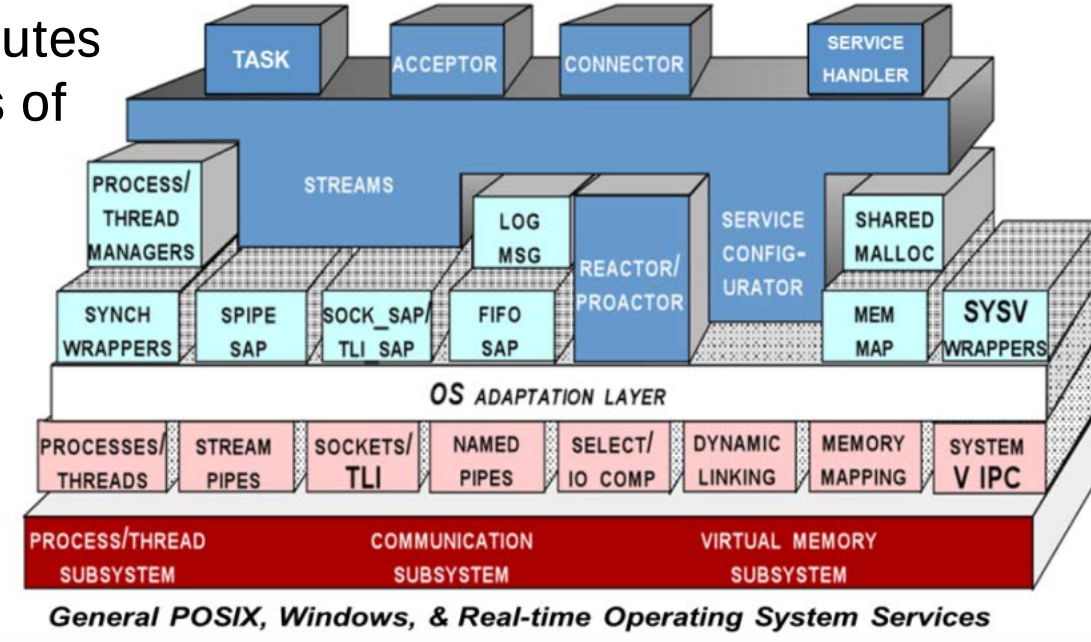
Applying SCV Analysis to ACE

- **Variabilities** describe the attributes unique to different instantiations of ACE
  - *Product- & domain-dependent components*
    - e.g., different internal implementations depending on the underlying OS



Applying SCV Analysis to ACE

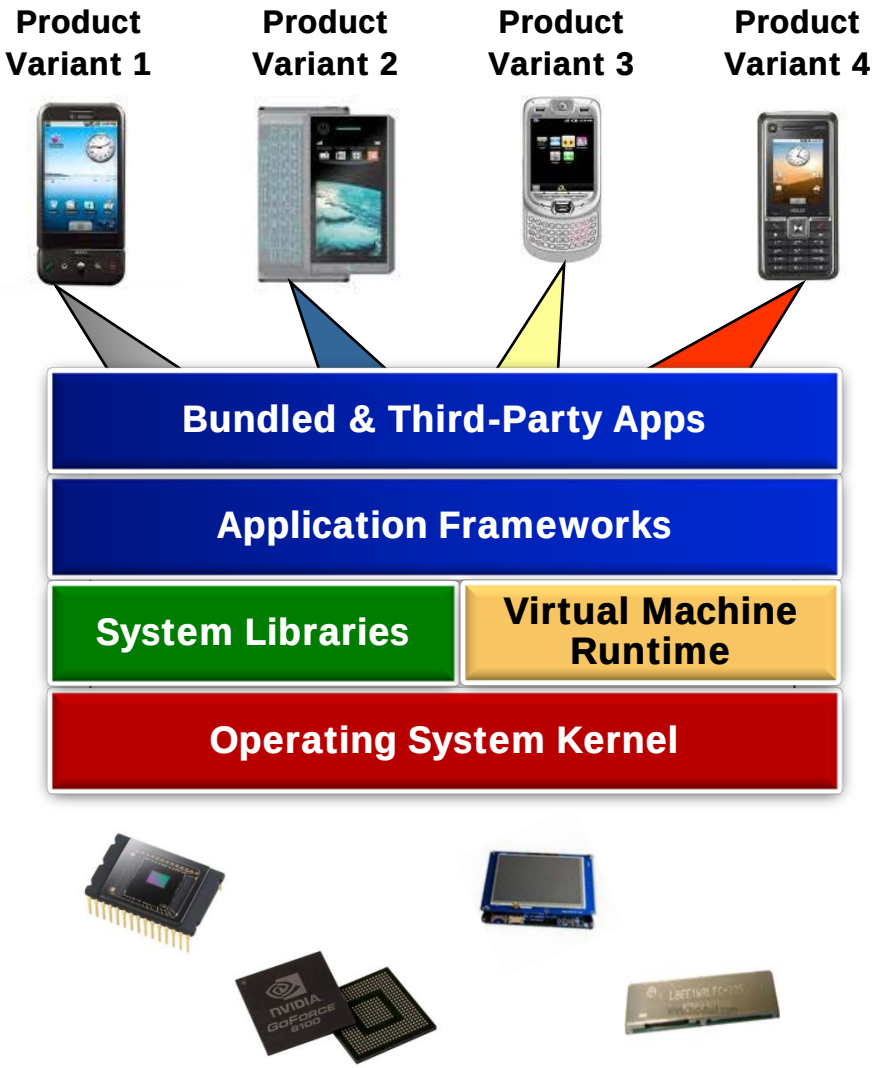
- **Variabilities** describe the attributes unique to different instantiations of ACE
  - *Product- & domain-dependent components*
    - e.g., different internal implementations depending on the underlying OS
  - *Product-dependent component assemblies*
    - e.g., different subsets of ACE components depending on constraints of the operating environment & user domain requirements



SCV can also be applied recursively for all the ACE frameworks & layers

Summary

- *Scope, Commonality, & Variability* (SCV) analysis is an advanced systematic reuse technique
  - It helps developers alleviate problems associated with maintaining many versions of the same product that have large amounts of similar software created to satisfy new & diverse requirements



Summary

- *Scope, Commonality, & Variability* (SCV) analysis is an advanced systematic reuse technique
  - It helps developers alleviate problems associated with maintaining many versions of the same product that have large amounts of similar software created to satisfy new & diverse requirements
- The frameworks in Android & ACE form software product-lines that enable systematic software reuse across a wide range of apps & infrastructure platforms