#### **Overview of Frameworks: Part 3**



Douglas C. Schmidt

<u>d.schmidt@vanderbilt.edu</u>

www.dre.vanderbilt.edu/~schmidt

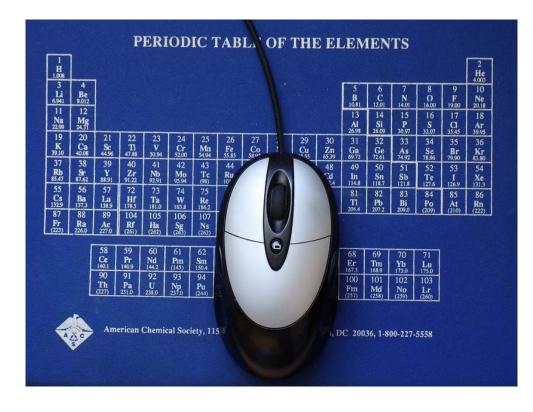
Institute for Software Integrated Systems Vanderbilt University Nashville, Tennessee, USA



CS 282 Principles of Operating Systems II
Systems Programming for Android

### Learning Objectives of this Module

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



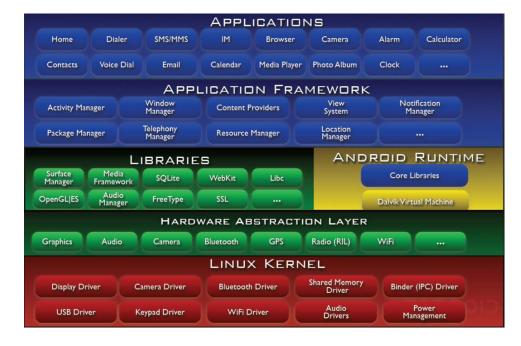




#### Learning Objectives of this Module

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



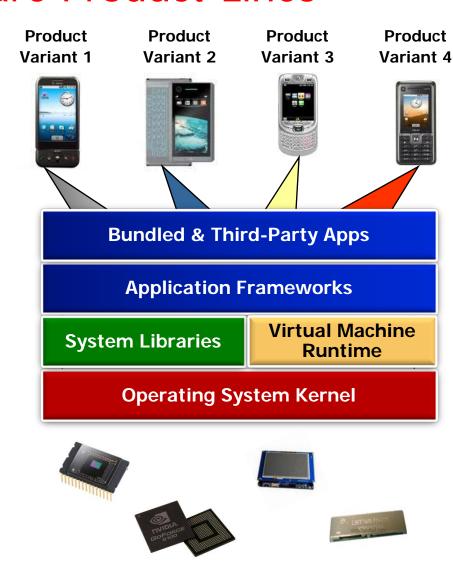






#### Overview of Software Product-Lines

- 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 can help define & improve core SPL assets by factoring out many reusable general-purpose & domain-specific services from application responsibility



See <a href="https://www.sei.cmu.edu/productlines">www.sei.cmu.edu/productlines</a> for more info on software product-lines

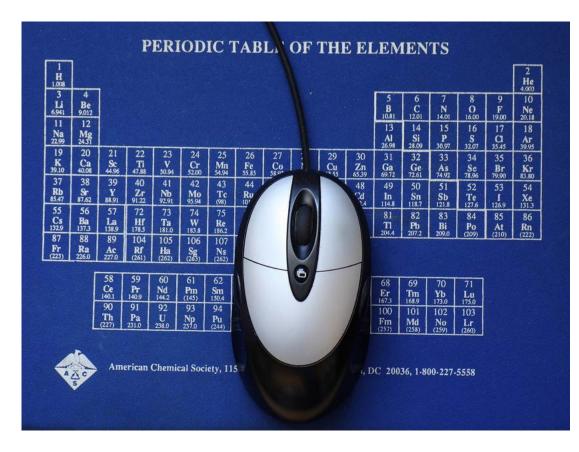
 Key software product-line & framework structure & behavior can be captured systematically via Scope, Commonality, & Variability (SCV) analysis







- 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



- 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
  - Often used to guide the development & application of software product-lines & frameworks







- 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
- General method
  - Identify common portions of a domain & define stable interfaces (fairly easy)

#### AsyncTask

execute()
onPreExecute()
doInBackground()
onProgressUpdate()
onPostExecute()





- 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
- General method
  - Identify common portions of a domain & define stable interfaces (fairly easy)
  - Identify variable portions of a domain & define stable interfaces (harder)

Params – Types used in background work Progress – Types used when indicating progress Result – Types of result Params, Progress, Result ! AsyncTask execute() onPreExecute() doInBackground() onProgressUpdate() onPostExecute() **DownloadAsyncTask** onPreExecute() doInBackground() onPostExecute()





- 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
- General method
  - Identify common portions of a domain & define stable interfaces (fairly easy)
  - Identify variable portions of a domain & define stable interfaces (harder)
  - Create different implementations of the variable portions as plug-ins

Params – Types used in background work Progress – Types used when indicating progress Result – Types of result Params, Progress, Result ! AsyncTask execute() onPreExecute() doInBackground() onProgressUpdate() onPostExecute() **DownloadAsyncTask** onPreExecute() onPreExecute() doInBackground() doInBackground() onPostExecute() onPostExecute()



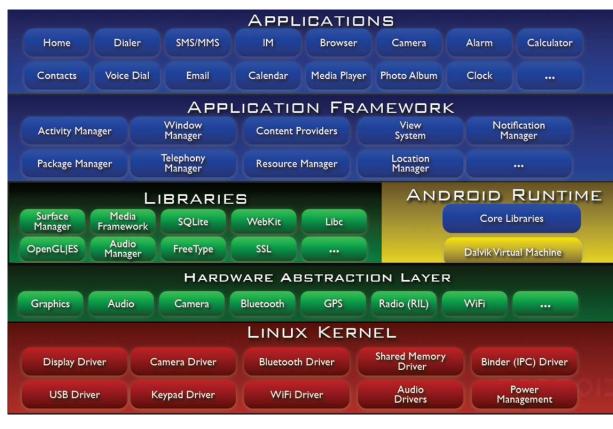


- Key software product-line & framework structure & behavior can be captured systematically via Scope, Commonality, & Variability (SCV) analysis
- This process can be applied to identify commonalities & variabilities in a domain
- General method
  - Identify common portions of a domain & define stable interfaces (fairly easy)
  - Identify variable portions of a domain & define stable interfaces (harder)
  - Create different implementations of the variable portions as plug-ins

Params – Types used in background work Progress – Types used when indicating progress Result - Types of result Params, Progress, Result! AsyncTask execute() onPreExecute() doInBackground() onProgressUpdate() onPostExecute() **DownloadAsyncTask** onPreExecute() onPreExecute() doInBackground() doInBackground() onPostExecute() onPostExecute()

en.wikipedia.org/wiki/SOLID\_(object-oriented\_design) has more info

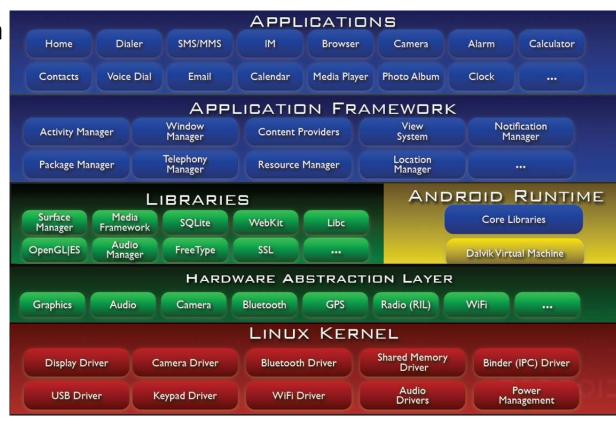
- Scope defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points







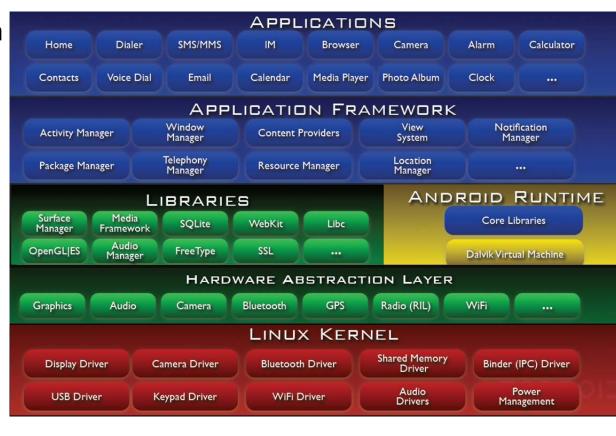
- Scope defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points
  - Touch-based user interfaces







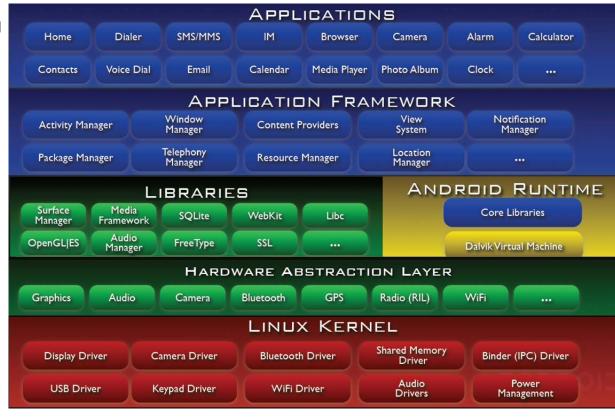
- Scope defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points
  - Touch-based user interfaces
  - (Largely) open-source, vendor- & hardware-agnostic ecosystem







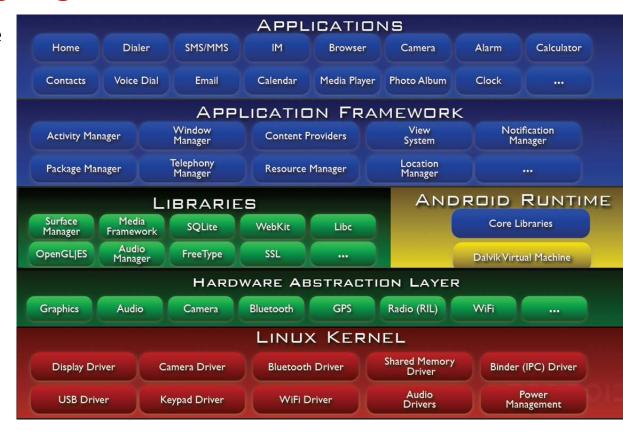
- Scope defines the domain & context of Android & its various frameworks & components
- e.g.,
  - Resource-constrained mobile devices
    - e.g., limited power, memory, processors, network, & price points
  - Touch-based user interfaces
  - (Largely) open-source, vendor- & hardware-agnostic ecosystem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







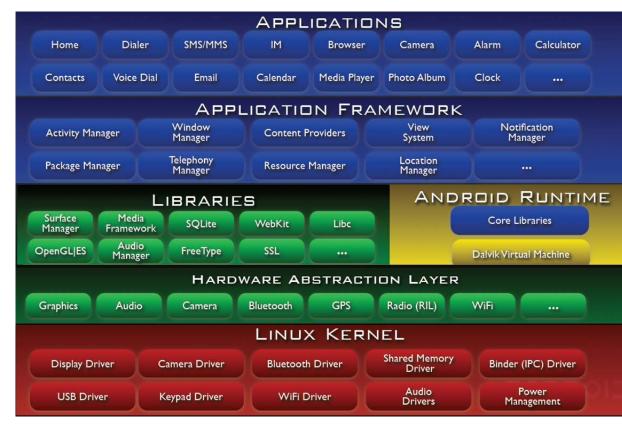
- Commonalities describe the attributes common across all instances of Android
  - Common framework components
    - e.g., Activities, Services, Content Providers, & Broadcast Receivers







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



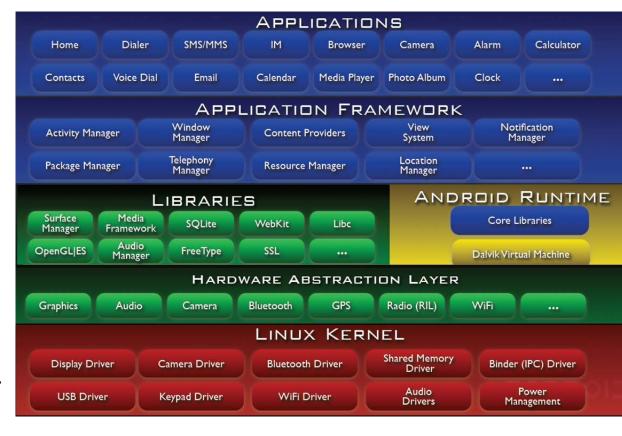
 e.g., Activity Manager, Package Manager, Telephony Manager, Location Manager, Notification Manager, etc.





- Commonalities describe the attributes common across all instances of Android
  - Common framework components
    - e.g., Activities,
       Services, Content
       Providers, &
       Broadcast Receivers
  - Common application frameworks
- APPLICATIONS Dialer SMS/MMS IM Camera Alarm Calculator Home Browser Contacts Voice Dial Email Calendar Media Player Photo Album Clock ... APPLICATION FRAMEWORK Window Notification View **Activity Manager** Content Providers Manager System Manager Telephony Manager Location Package Manager Resource Manager Manager ANDROID RUNTIME LIBRARIES Surface Media Core Libraries **SQLite** WebKit Libc Manager Framework Audio SSL **OpenGLIES** FreeType Dalvik Virtual Machine Manager HARDWARE ABSTRACTION LAYER Graphics Audio **GPS** WiFi Camera Bluetooth Radio (RIL) LINUX KERNEL Shared Memory Display Driver Camera Driver Bluetooth Driver Binder (IPC) Driver Driver Audio Power **USB** Driver Keypad Driver WiFi Driver Drivers Management
- 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.

- 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.







- 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
- Dialer SMS/MMS IM Camera Alarm Home Browser Calculator Contacts Voice Dial Email Calendar Media Player Photo Album Clock ... APPLICATION FRAMEWORK Notification Window View **Activity Manager** Content Providers Manager System Manager Telephony Manager Location Package Manager Resource Manager Manager ANDROID RUNTIME LIBRARIES Surface Media Core Libraries **SQLite** WebKit Libc Manager Framework Audio OpenGL|ES SSL FreeType Dalvik Virtual Machine Manager HARDWARE ABSTRACTION LAYER Audio Camera **GPS** Radio (RIL) WiFi Graphics Bluetooth LINUX KERNEL Shared Memory Display Driver Camera Driver Bluetooth Driver Binder (IPC) Driver Driver Audio Power **USB** Driver Keypad Driver WiFi Driver Drivers Management

APPLICATIONS

 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

#### 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 form software product-lines that enable systematic software reuse across a wide range of apps & infrastructure platforms

