Overview of Frameworks: Part 3

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
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 is 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 [www.sei.cmu.edu/productlines](http://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.

www.cs.iastate.edu/~cs309/references/CoplienHoffmanWeiss_CommonalityVariability.pdf
Overview of Frameworks

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.

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

Diagram:

- **Product Variant 1**
- **Product Variant 2**
- **Product Variant 3**
- **Product Variant 4**

Levels:

- Bundled & Third-Party Apps
- Application Frameworks
- System Libraries
- Virtual Machine Runtime
- Operating System Kernel
Overview of Frameworks

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.

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

**AsyncTask**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute()</td>
</tr>
<tr>
<td>onPreExecute()</td>
</tr>
<tr>
<td>doInBackground()</td>
</tr>
<tr>
<td>onProgressUpdate()</td>
</tr>
<tr>
<td>onPostExecute()</td>
</tr>
</tbody>
</table>
Overview of Frameworks

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.

- 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
```

```
AsyncTask
execute()
onPreExecute()
doInBackground()
onProgressUpdate()
onPostExecute()
```

```
DownloadAsyncTask
onPreExecute()
doInBackground()
onPostExecute()
```
Overview of Frameworks

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.

- **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.

### AsyncTask
- `execute()`
- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`

### DownloadAsyncTask
- `onPreExecute()`
- `doInBackground()`
- `onPostExecute()`

### Params, Progress, Result
- **Params** - Types used in background work
- **Progress** - Types used when indicating progress
- **Result** - Types of result
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.

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

- **AsyncTask**
  - `execute()`
  - `onPreExecute()`
  - `doInBackground()`
  - `onProgressUpdate()`
  - `onPostExecute()`

- **Params** - Types used in background work.
- **Progress** - Types used when indicating progress.
- **Result** - Types of result.

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
Overview of Frameworks

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](http://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
Overview of Frameworks

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.
Overview of Frameworks

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.
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
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.