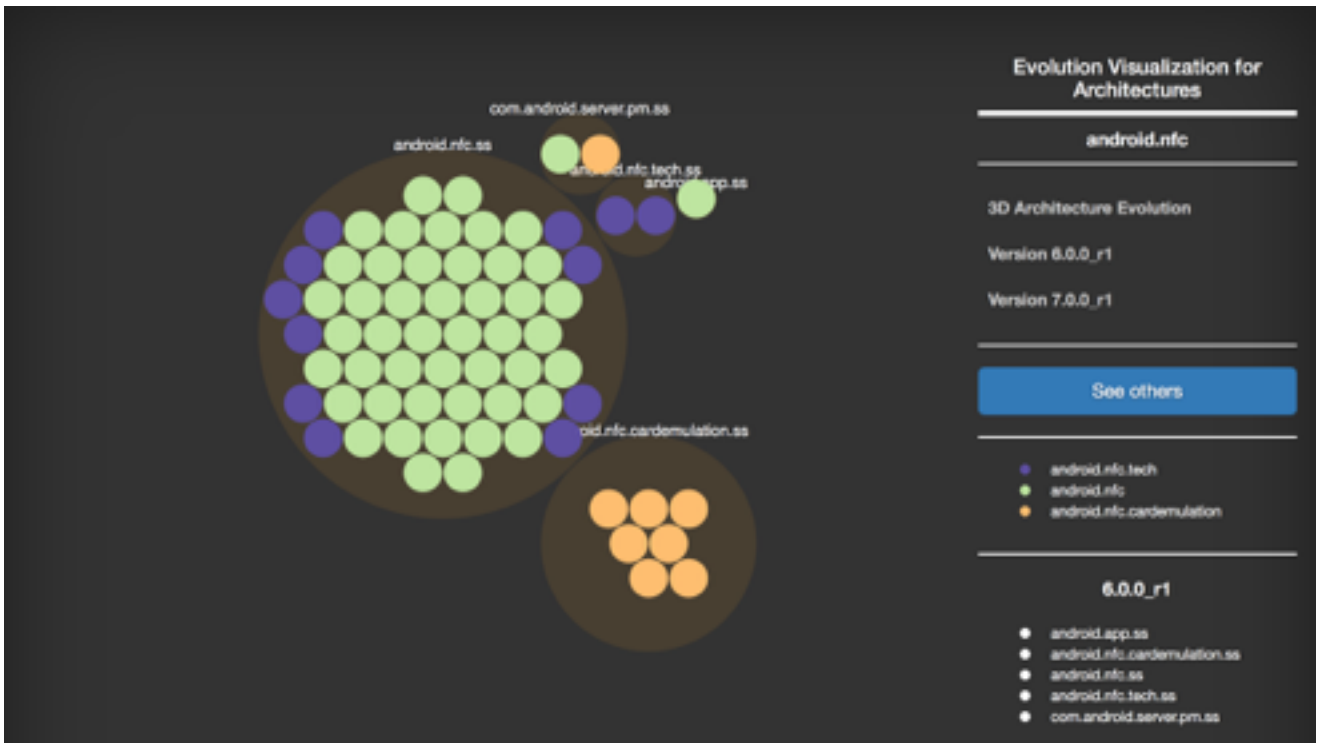
# EVA Manual

## EVA: A Tool for Visualizing Software Architectural Evolution




Computer Science Department, University of Southern California

*1st Edition*

# 1. Installing EVA

## 1.1 Requirements

The following is the operating system, execution environments, tools we have used to make sure EVA works as described in this manual. Different configurations may work, but have not beed tested.

### Operating Systems

- macOS Sierra, Version 10.12

### Browsers

- Chrome 65.0
- Safari 10.0

### Python

- Python 2.7

### Localhost

- MAMP (for mac)

### Display Size

- 27-Inch

Before you run EVA, you have to install below python packages.

- enum34 (1.1.6)
- ortools (6.6.4656)

Since EVA's visualization is developed to be shown on a web-browser using HTML, CSS and Javascript, you need a localhost environment. For Mac, you can set up the localhost environment easily using MAMP (mac) or WAMP (windows). You can also move the source codes to your own server and access the visualization through the web browser.

## 1.2 Overview

EVA is written in Python, Javascript, HTML, and CSS.

1. EVA source code can be downloaded from the following Github repository:

   https://github.com/namdy0429/EVA-Huawei

2. Place the EVA project into the localhost folder.

3. Run EVA.py to generate the input data of the EVA visualization.
4. View the EVA's visualization with your browser.

# 2. Visualizing with EVA

## 2.1 Conventions used in this and the Following Sections

- In this section, Android 6.0.0 and Android 7.0.0 are used in an invocation example throughout this manual.
- Note that the following files have been provided in the folder hierarchy of the EVA project. If all examples in Section 3 are followed in sequence, the results should be exactly as indicated.

| Item | Location |
| --- | --- |
| Input Recovered Architectures | Data/Architecture/android/{recovery method}/{layer} |
| A List of Sub-systems under an Architectural Layer | Engine/config/ |

## 2.2 Processing Data

**Synopsis:**

```
python EVA.py –outputDir –recovery –arch1 –arch2 –layer
```

**Parameters:**

- `outputDir`
  This is the dictionary where all output files go, which can be the localhost EVA folder.
- `recovery`
  This is the recovery method that is used for architecture recovery.
- `arch1, arch2`
  These are .rsf format files resulting from running ARCADE. arch1 should be the older version, and arch2 is the newer version.
- `layer`
  Because of the scalability issue, a user will recover Android's architecture partially, based on each architectural layer. This is the name of the architectural layer that a user recovered.

  Also, you have to place a list of sub-systems under "config/", using this layer name.

### Example:

```
python EVA.py --outputDir /Applications/MAMP/htdocs/EVA/Front-end --
recovery acdc --arch1 android-6.0.0_r1_acdc_clustered.rsf --arch2
android-7.0.0_r1_acdc_clustered.rsf --layer framework
```

### Resulting Files:

```
EVA/Front-End/data/android/acdc/6.0.0_r1_7.0.0_r1_framework/
├── 6.0.0_r1_7.0.0_r1_android.accessibilityservice_processed_archs.json
├── 6.0.0_r1_7.0.0_r1_android.accounts_processed_archs.json
├── …
├── 6.0.0_r1_7.0.0_r1_com.android.server.webkit_processed_archs.json
└── 6.0.0_r1_7.0.0_r1_com.android.server.wm_processed_archs.json
EVA/Front-End/data/android/acdc/6.0.0_r1_framework/
├── 6.0.0_r1_android.accessibilityservice.ss_processed_archs.json
├── 6.0.0_r1_android.accounts.ss_processed_archs.json
├── …
├── 6.0.0_r1_com.android.server.wm.ss_processed_archs.json
└── 6.0.0_r1_com.google.android.util.ss_processed_archs.json
EVA/Front-End/data/android/acdc/7.0.0_r1_framework/
├── 7.0.0_r1_android.accessibilityservice.ss_processed_archs.json
├── 7.0.0_r1_android.accounts.ss_processed_archs.json
├── …
├── 7.0.0_r1_com.android.server.wm.ss_processed_archs.json
└── 7.0.0_r1_com.google.android.util.ss_processed_archs.json
```

## 2.3 Rendering Visualization

There are two ways of accessing EVA's visualization.

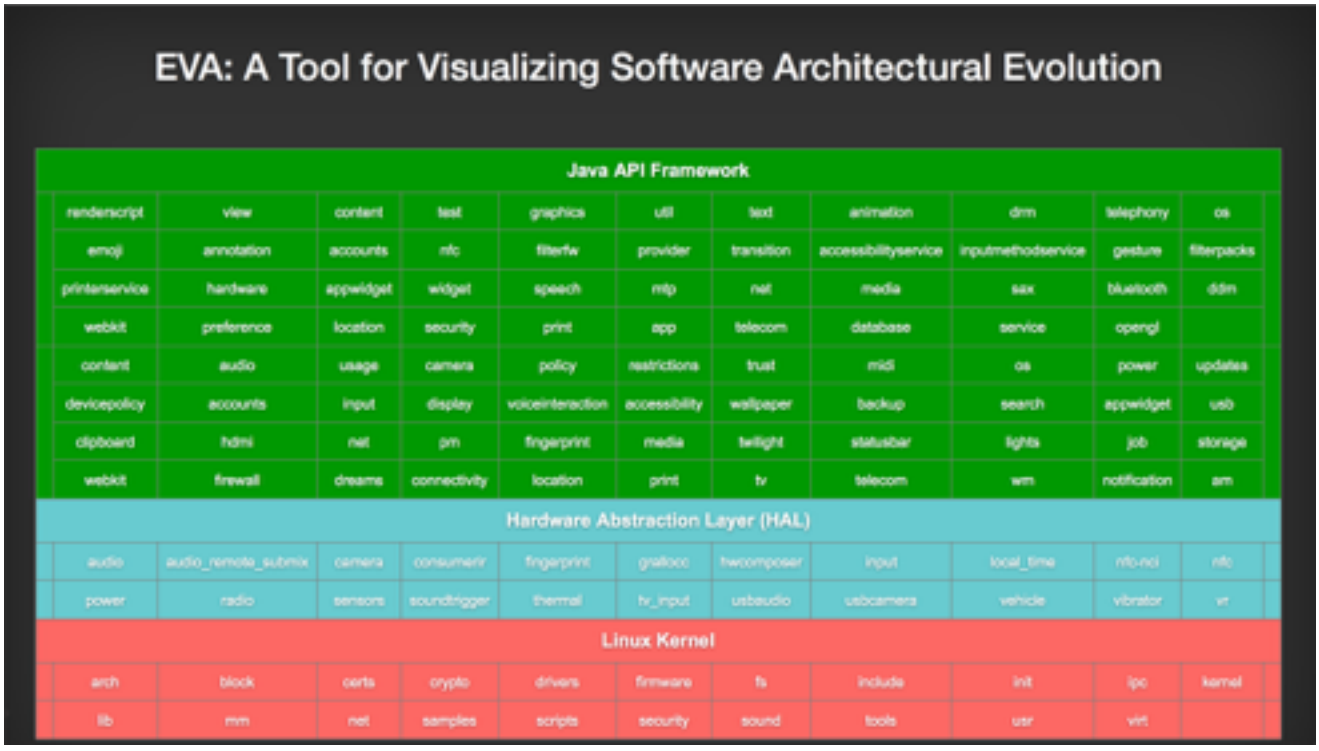


Fig 1. EVA index page.

## *(1) Start from the index page*

**URL:**

```
[localhost EVA path]/index.html?
&recovery=[recovery_method]&ver1=[version1]&ver2=[version2]&layer=[l
ayer]
```

**Parameters:**

- [localhost EVA path]: EVA Front-end directory within localhost folder.
- [recovery_method]: A recovery method used to recover the system architecture.
- [version1], [version2]: Two android versions to compare. Put the older version into [version1], and the newer version into [version2].
- [layer]: The target architectural layer for the visualization.

**Example:**

```
http://localhost:8888/EVA/Front-end/index.html?
&recovery=acdc&ver1=6.0.0_r1&ver2=7.0.0_r1&layer=framework
```

**Description:**

Each big colored block refers to one architectural layer of Android. Each small block within the architectural layer refers to a sub-system within the architectural layer.

When you click a sub-system, you can move to the corresponding visualization page.

## *(2) Query using GET*

**URL:**

```
[localhost EVA path]/index.html?
&recovery=[recovery_method]&ver1=[version1]&ver2=[version2]&layer=[l
ayer]&target_sub=[target_subsystem]
```

**Parameters:**

- [localhost EVA path]: EVA Front-end directory within localhost folder.

- [recovery_method]: A recovery method used to recover the system architecture.
- [version1], [version2]: Two android versions to compare. Put the older version into [version1], and the newer version into [version2].
- [layer]: The target architectural layer for the visualization.
- [target_subsystem]: The target sub-system for the visualization. The selected sub-system should be within the selected architectural layer.

**Example:**

`http://localhost:8888/EVA/index_combined.html?&recovery=acdc&ver1=6.0.0_r1&ver2=7.0.0_r1&layer=framework&target_sub=android.media`

**Description:**

## 2.4 Exploring System Architecture using EVA

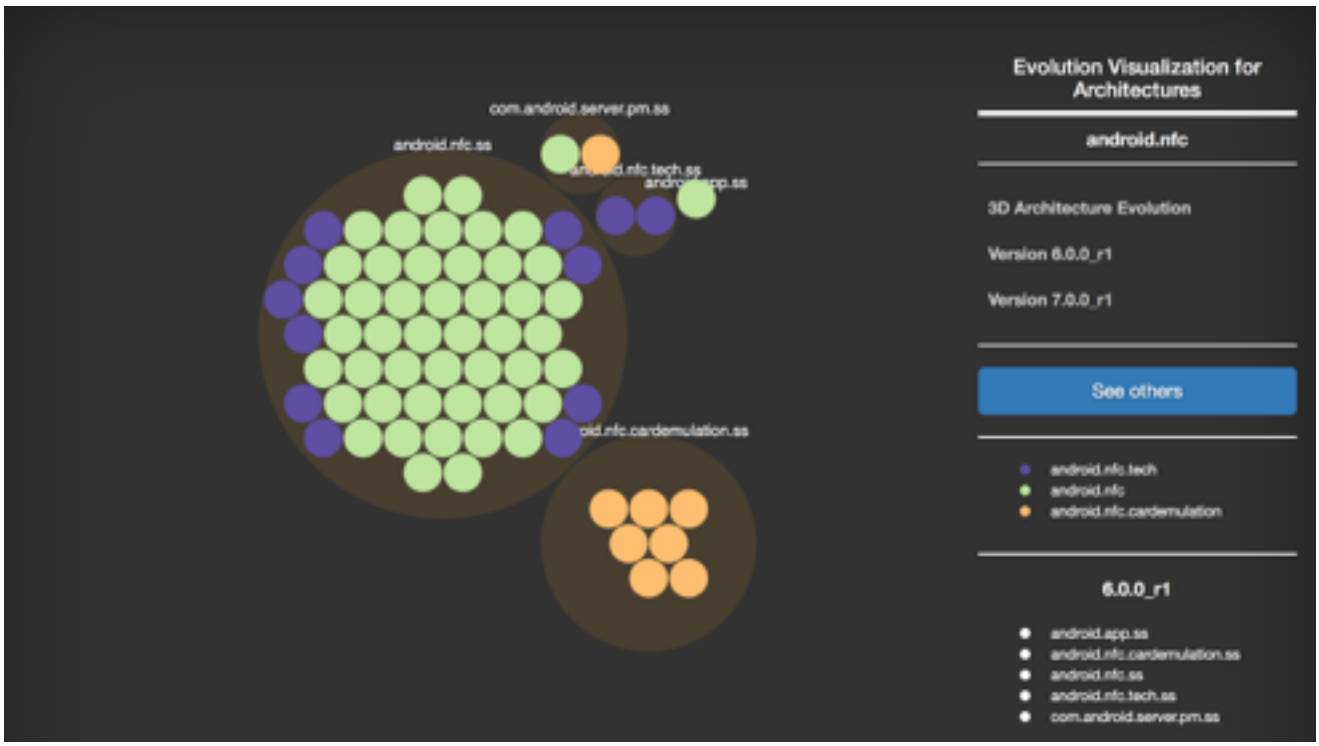



Fig 2. EVA Visualization page example. Visualizing android.nfc, which is a sub-system of Android Framework layer.

- EVA generates a visualization page for each sub-system. For example, Figure 2 shows the architecture of android.nfc in the Framework layer. When you combine all the sub-

visualizations of one layer, it will show the whole architecture of the given architectural layer. However, since Android is too big to visualize into a single page, we partitioned the visualization.

- Each bigger circle refers to a architectural component recovered using the recovery technique in ARCADE. The white texts above the circle is the name of each component.
- Each small dot refers to a code-level entities (e.g., class in java).
- The color of small dots shows the code-level's package information. For example, purple dots in Figure 2 are classes belong to android.nfc.tech package. From this color-coding, it is easy to see whether the code-level distributions corresponds to the system architecture.
- Click "See others" button to see the index table, and move to other sub-systems.
- You can see each component, without filtering for each sub-system, by clicking each component name on right list (e.g., android.app.ss).