Extract “Feature Architecture” of your Software Application

Tajmilur Rahman, PhD
Assistant Professor
Gannon University
rahman007@gannon.edu
Tajmilur Rahman, PhD, is an assistant professor in the department of Computer and Information Science at Gannon University in Erie, Pennsylvania, United States. Dr. Rahman received his doctorate degree in 2018 from Concordia University, Montreal QC, Canada.

His overarching research interest is to investigate release engineering practices in software systems. His research interests also include software engineering & data science, understanding the significance of software architecture for long lasting software systems, and providing tool support to the community to nurture software feature architecture. He is also interested in studies on software engineering education.

Dr. Rahman is the first author who extracted feature-architecture. His currently ongoing research works include, software release management, software quality prediction in rapid-release, predict potential architectural drift, and software engineering education.

rahman007@gannon.edu
Outline

- Software architecture
- Why we must be aware of architecture
- How do we extract software architecture
- Traditional approach
- Modern release engineering
- Trunk based development
- Feature toggles
- Feature architecture
- Extracting Feature Architecture
- Benefits of Feature Architecture
Software Architecture

- Fundamental structure/organization of a system
Software Architecture

- Fundamental structure/organization of a system

"Architecture is an abstract description of the entities in a system and relationships between them."

- Zhang Jianfei, Senior Technical Expert, Alibaba
Software Architecture

“Software architecture is a draft of the system.”¹

- Zhang Jiang, Senior Technical Expert, Alibaba

“Software architecture is the design decisions that need to be made early in a project.”⁴

- Martin Fowler, ThoughtWorks
Why architecture matters?

Poor architecture contributes to generating a lot of “cruft”

Any software system has a certain amount of essential complexity required to do its job...

... but most systems contain cruft that makes it harder to understand.

The technical debt metaphor treats the cruft as a debt, whose interest payments are the extra effort these changes require.
Software Architecture Categories
Business Architecture

● Directed by
  ○ Business model analysts
  ○ Business domain experts
  ○ Industry experts
Application Architecture

- Directed by
  - Application Analysts:
    - Design application layers
    - Application specifications
    - API and Data integration / interaction protocols

rahman007@gannon.edu
Distributed System Architecture

- Architect needs to keep balance among
  - Consistency
  - Availability
  - Partition tolerance
Data Architecture

- Data governance
  - Important for large systems
  - Unify data definition specifications
  - Standardize data expression
  - Unified data processing platform

rahman007@gannon.edu
Physical Architecture

- Focuses on the layout
  - Data center construction
  - Network topology
  - Servers
  - Storage
  - Hosts
Physical Architecture

- Types
  - Conceptual Architecture
  - Concrete Architecture
Conceptual Architecture
Conceptual and Concrete Architecture - Linux Kernel

Conceptual Architecture

Concrete Architecture

rahman007@gannon.edu
Responsibilities of Architects

- Make complex things simple
- Refine thinking for better understanding of complex systems
- Build an easy-to-understand arch.
Architecture & Agility

Many developers now prefer no architectural design for greater agility. They think it is fine just to get started on the work.

“It can mean anything from a high-level design sketch with little relationship to technology, code, or the actual system being built\(^2\) to a big, rigid up-front design with a lot of class and code-level minutiae.\(^3\)
Why be Aware of Architecture?

“Project teams aren’t paid for meeting a process but for delivering running software!”

rahman007@gannon.edu
Extract Architecture for Better Agility

“Continuous attention to technical excellence and good design enhances agility
“Continuous attention to technical excellence and good design enhances agility"
Too many patches
Continuous attention to the architectural drift enhances agility

- Tajmilur Rahman PhD

Better Agility

rahman007@gannon.edu
Extract Architecture

- To keep track of architectural drift
- Build awareness about adding new features
- Know when you increasing coupling or decoupling
Extract Architecture

How?
Approaches to Extract Architecture

- Traditional physical (modular) architecture
- Feature toggles to extract architecture
Modern Release Engineering

- Rapid release / shorter version of release cycle
- Trunk based development
- Continuous integration and continuous delivery
Chrome’s 4 week rapid release plan
Feature Toggles

- A variable used in an if condition
- Hides unfinished code already merged into the trunk
- Allows controlled release of features
Feature Toggles

- Feature Toggles / Flags / Flips

**File: config.properties**

```properties
graphic_features {
    ENABLE_3D_GRAPHIC=true,
    ENABLE_TOUCH_SCREEN=true
}
```

**File: RenderGraphics.java**

```java
if(graphic_features.ENABLE_3D_GRAPHIC) {
    render3DGraphics();
} else {
    render2DGraphics();
}
```
GPU Switches / Toggles in G-Chrome

```cpp
// Copyright 2015 The Chromium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#include "gpu/config/gpu_switches.h"

namespace switches {
  // Disable GPU rasterization, i.e. rasterize on the CPU only.
  // Overrides the kEnableGpuRasterization flag.
  const char kDisableGpuRasterization[] = "disable-gpu-rasterization";

  // Disables mipmap generation in Skia. Used a workaround for select low memory
  // devices, see https://crbug.com/1138979 for details.
  const char kDisableMipmapGeneration[] = "disable-mipmap-generation";

  // Allow heuristics to determine when a layer tile should be drawn with the
  // Skia GPU backend. Only valid with GPU accelerated compositing.
  const char kEnableGpuRasterization[] = "enable-gpu-rasterization";

  // Select a different set of GPU blocklist entries with the specified
  // test_group ID.
  const char kGpuBlocklistTestGroup[] = "gpu-blocklist-test-group";
```

chromium/src/refs/heads/main/gpu/config/gpu_switches.cc

rahman007@gannon.edu
Feature Architecture

- Architecture that represents the features crosscutting the physical/modular architecture
- Shows relationships that are not visible in a modular architecture
- Shows which features are crosscutting which modules
- Shows which modules containing which features
- How two modules are dependent via common features
Feature Architecture
Extract Feature Architecture from Toggles
Conceptual Architecture

Gather knowledge on the existing architecture

- The initial architecture designed by architects
- If no initial architecture, then construct the conceptual architecture
Extract Feature Architecture from Toggles\textsuperscript{5}

1. Gather knowledge on the existing architecture  
2. Collect / construct the conceptual architecture  
3. Take the source code and ignore all third-party code and libraries  
4. Analyze naming convention, directory structure, file path  
5. Map files/directories with conceptual modules  
6. Extract call-graph and module-module relationships to construct the concrete architecture  
7. Extract feature toggles  
8. Find out feature dependencies with modules based on the map at step 5  
9. Plot the \textit{feature dependencies} and normalize based on the concrete architecture
Modular / Concrete Architecture

Traditional approach:

- Bowman et. al\textsuperscript{6}
- Used automated tool (LsEdit) to extract relations from the source code then combined these relations into a concrete system architecture.
- We can use “Understand” by SciTools
Modular / Concrete Architecture

Architecture Browser

The Architecture Browser allows you to manage architectures. It shows a list of all the defined architectures in the database and provides a way to navigate individual architectures.

For example, this window shows the auto-architectures provided with Understand: Calendar, Directory Structure, Languages. The architectures are expanded somewhat here to show the top-level nodes for an example application.

You can use the auto-architectures, create your own architectures, import and export architectures (as XML files), generate graphs and metrics for any level in an architecture hierarchy, and combine architectures through filtering.
Modular / Concrete Architecture

Architecture Browser:  

- Understand analyzes your software code and creates a database containing information about the entities and the relations between entities.
- The database can be browsed using various “graphical view” windows.
- Hierarchy views show relations between entities.
- Each view follows a relation (for instance “Calls”) from the starting entity (that you inquired about) through its children and successors.
- Structure views quickly show the structure of any entity that adds to the structure of your software (for instance a package, function, procedure, or task).
Modular / Concrete Architecture
Modular / Concrete Architecture
Modular / Concrete Architecture
Modular / Concrete Architecture

Once directories, dependencies, call-graph are extracted:
1. Map the directory/file paths with conceptual modules

<table>
<thead>
<tr>
<th>Path</th>
<th>Conceptual Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>content/public/common/speech_recognition_result.cc</td>
<td>Content</td>
</tr>
<tr>
<td>content/public/common/url_constants.cc</td>
<td>Content</td>
</tr>
<tr>
<td>chrome/common/ipc_channel_posix.cc</td>
<td>Chrome</td>
</tr>
<tr>
<td>ipc/common/mailbox_struct_traits.h</td>
<td>IPC</td>
</tr>
<tr>
<td>gpu/ipc/common/mailbox_struct.cc</td>
<td>GPU</td>
</tr>
<tr>
<td>ipc/common/ipc_switches.cc</td>
<td>IPC</td>
</tr>
</tbody>
</table>

rahman007@gannon.edu
Modular / Concrete Architecture
Modular / Concrete Architecture
Extract Feature Toggles

Extract feature toggles

(a)

```c
compositor_switches.cc (chrome/ui/compositor)
content_switches.cc (chrome/content/public/common)
gaia_switches.cc (chrome/google_apis/gaia)
```

(b)

```
107  // Disable 3D inside of flapper.
108  const char kDisableFlash3d[] = "disable-flash-3d";
109
110  // Disable using 3D to present fullscreen flash
111  const char kDisableFlashFullscreen3d[] = "disable-flash-fullscreen-3d";
```
# Map Feature Toggles with Modules

<table>
<thead>
<tr>
<th>Features (toggle set)</th>
<th>Concrete Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>content_switches</td>
<td>Browser View</td>
</tr>
<tr>
<td>content_switches</td>
<td>Content</td>
</tr>
<tr>
<td>content_switches</td>
<td>GPU</td>
</tr>
<tr>
<td>ui_base_switches</td>
<td>UI</td>
</tr>
<tr>
<td>media_switches</td>
<td>Content</td>
</tr>
</tbody>
</table>
Feature Architecture for Each Concrete Module

[Diagram showing the architecture with various nodes and connections representing different features and modules.]
Benefits of Feature Architecture

1. Better understand the dependencies among the modules
2. How features are cross-cutting modules
3. Developers need to know which modules are involved with a feature when modifying that feature
4. Helps making decision of adding new changes
For more details

“The modular and feature toggle architectures of Google Chrome”

References

8. A school project on Chromium: https://mococj.github.io/assignment2/