

# Assignment 2

## **Inspect Element**

Jonathon Gallucci, Joseph Gravenor, Jack Guinane,  
Maxwell Keleher, Matthew Pollock, Roberto Ruiz De La Cruz

# Intro

0. Intro
1. Derivation Process
2. Concrete Architecture
3. Justification
4. Sequence Diagram
5. Concrete Architecture of the UI Component
6. Concurrency
7. Components
8. A3 Feature
9. Lessons learned
10. Chrome-clusion



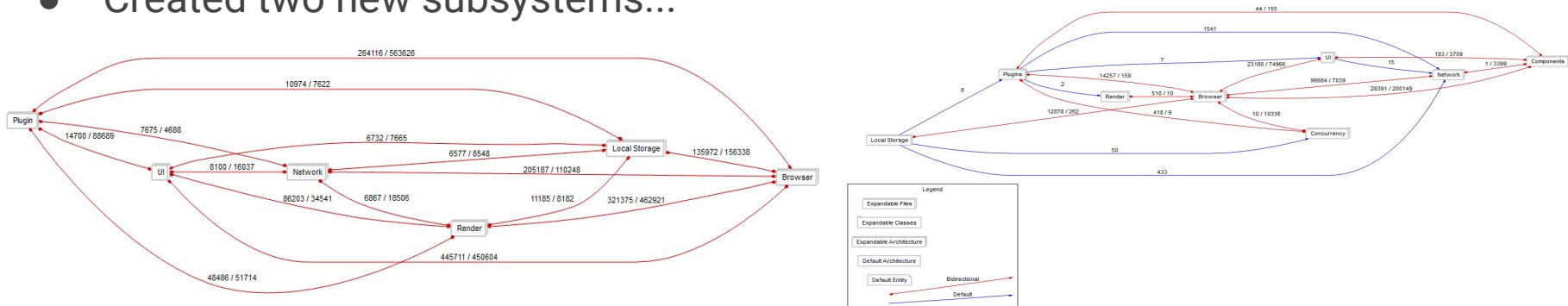
# Derivation Process

- To maximize the cohesion of our architecture, we looked through each directory in Understand one by one.
- This optimization for cohesion came with a price in the form of coupling. Each element is fully connected with high numbers of dependencies.
- This violates the benefits of our architecture style, as this level of coupling makes it impossible to change the implementation of one subsystem without affecting other subsystems
- It became clear to us that our architecture needed to change to change to correct this violation

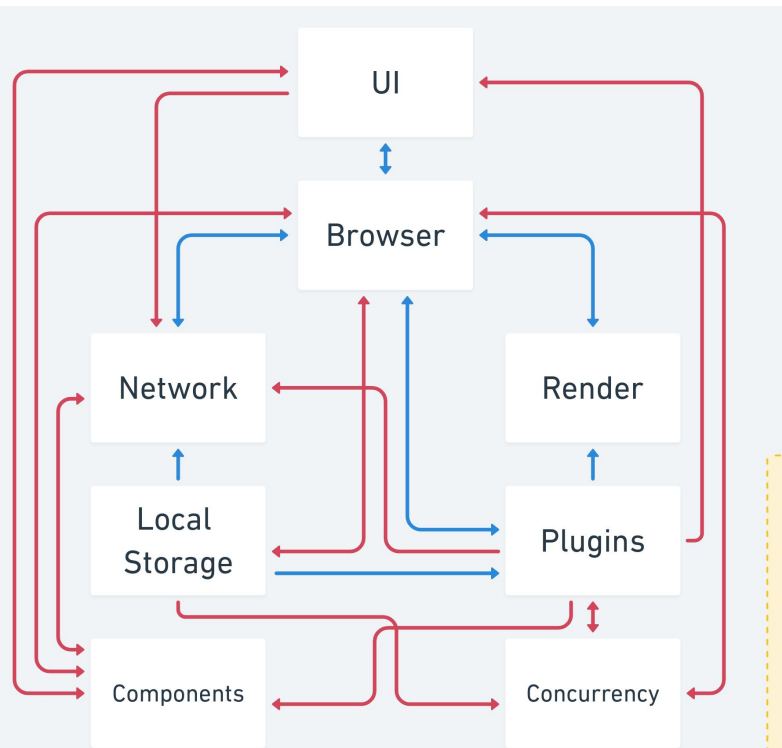


# Derivation, Continued

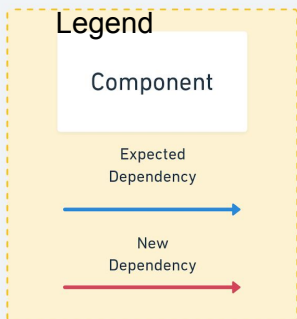
- To reduce coupling, we changed our approach
- Focusing on the purpose of the classes within the directory, we moved directories as units, choosing the most common purpose for each
- Although lower cohesion, we believe that these changes better reflect the concrete architecture
- Created two new subsystems...



# Concrete Architecture

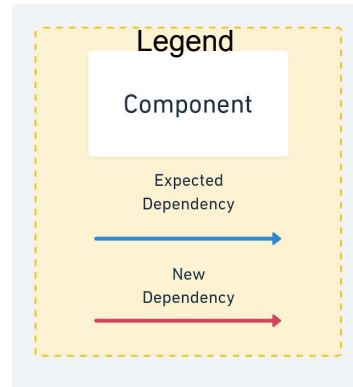
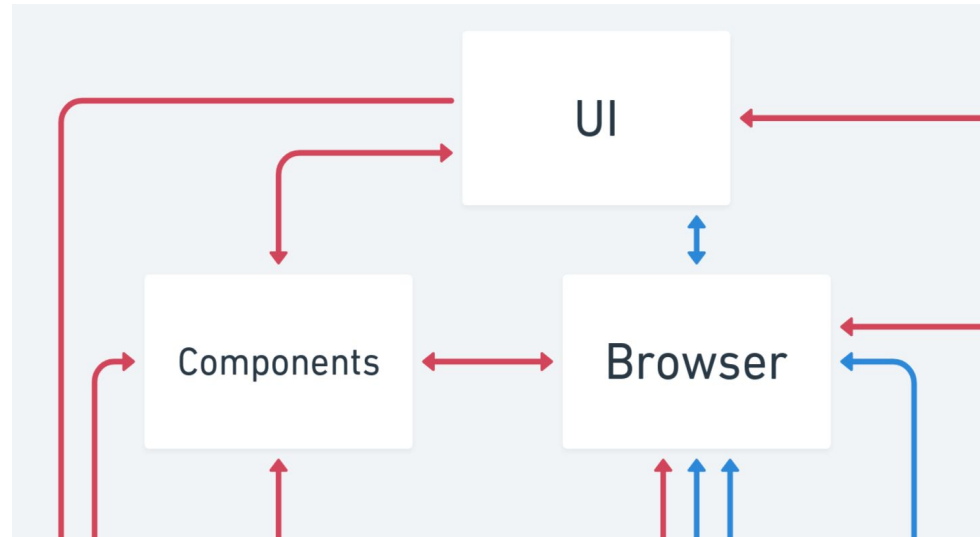


- This is the concrete architecture
- Directories were searched and sorted based on relations and services
- High coupling and high cohesion
- Best reflects Chrome as it is today

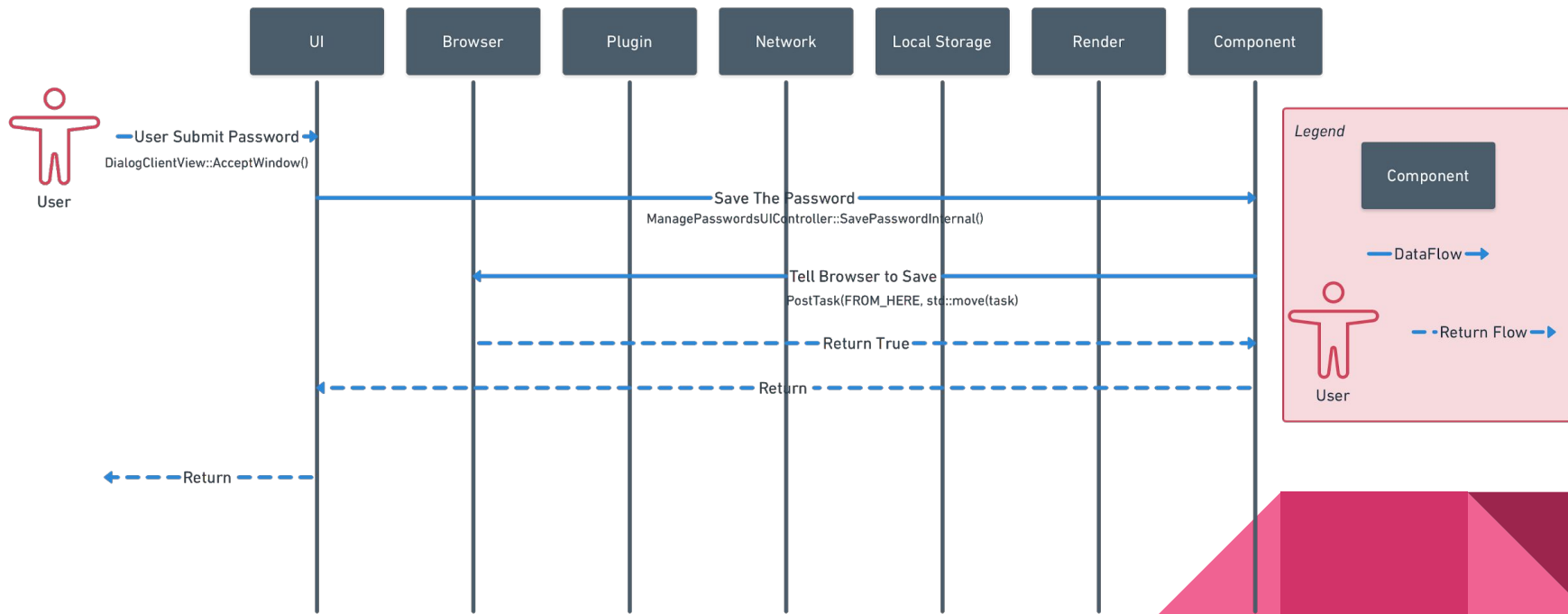


# Justification

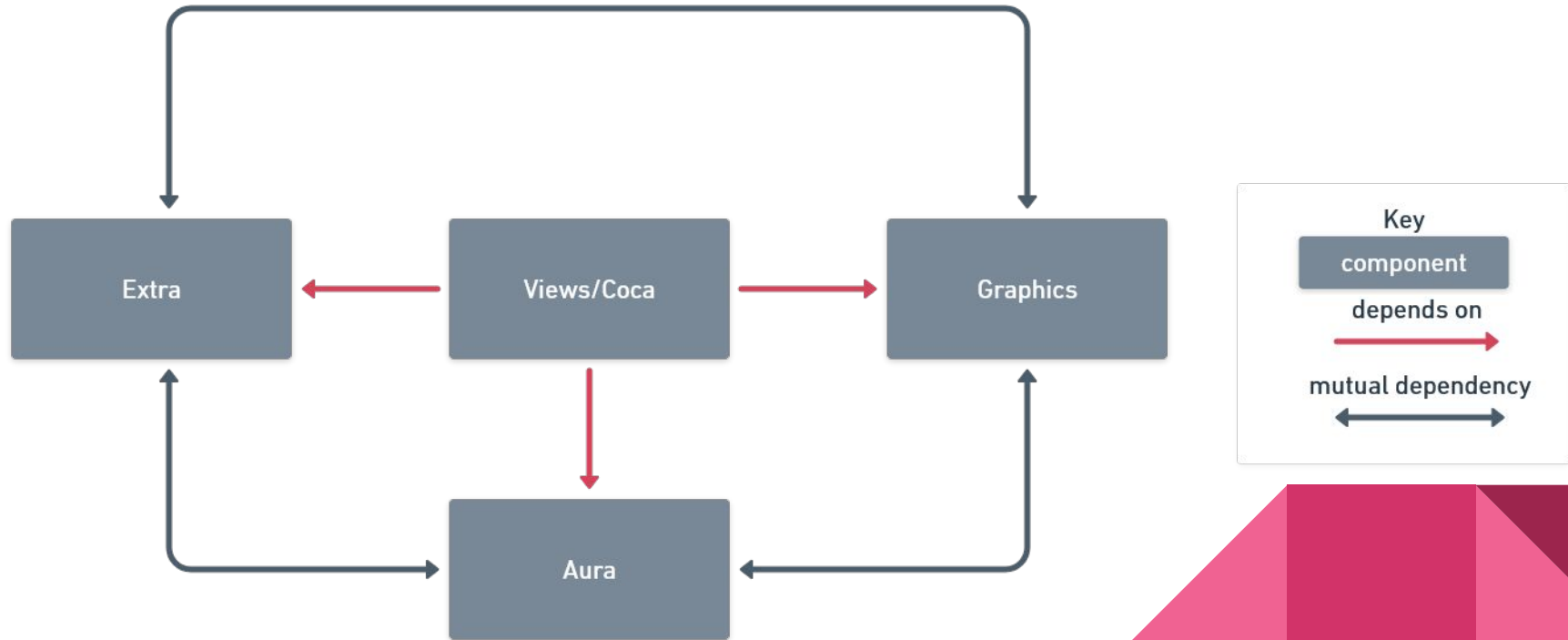
- The concrete is object-oriented and is *not* layered
  - Too many dependencies
- UI -> Browser becomes UI <-> Browser
  - Platforming handling
- New subsystem Components is codependent on UI



# Sequence Diagram



# Concrete UI





# Concurrency

- We originally required a “centralized concurrency controller”
- Previously determined to be a part of Browser subsystem
- Upon researching the concrete architecture, we discovered that this was its own subsystem
- Our previous report had noted that Mojo was the system they were using
- Thus, we created the “Concurrency” subsystem
- Main dependencies:
  - Plugin: for controlling sandboxed plugins, broker interfaces
  - Browser: for handling tabs, windows, iframes



# Components

- Our previous report suggested multiple extensions assessing the plugin subsystem
- We neglected to include these in any subsystem
- After viewing the codebase, the components directory housed most of these extensions, however were given more permissions
- Main dependencies:
  - UI: for interaction with the application window (bookmarks' 'star')
  - Network: for network access & network interruptions



# A3 Idea

[d.vena@queensu.ca](mailto:d.vena@queensu.ca)



New Message — ↗ ✕

To [d.vena@queensu.ca](mailto:d.vena@queensu.ca) ✕ | Cc Bcc

Subject

Hello Dan,

This is an example email to illustrate Inspect Element's proposed integrated email feature. As Google further invests into Gmail, a possible future development is to **provide Gmail directly in the browser**. This would allow Chrome to treat Gmail as the user's default mail client (whereas mailto links currently open whatever is installed: Outlook, Mail, etc). It would work like this:

- User clicks mailto link
- Email overlay opens in Chrome
- Write and send message without leaving Chrome

This additional feature would need to interact with the **UI subsystem for the generated overlay** as well as the **Renderer to override the current mailto links** and read their values. The integrated email client would likely be **built in the Components subsystem**.

Let me know what you think,  
Inspect Element

Send 🔗 📎 🔗 😊 🖼️ 🕒 🗑️ ⋮

# Lessons Learned

## Thick coupling & Low Cohesion

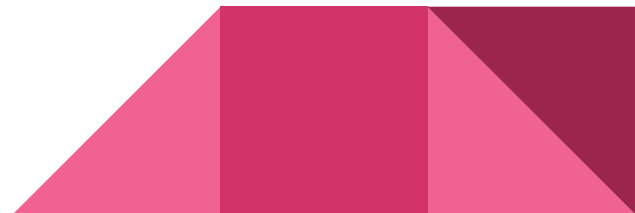
- Not layered style, all object oriented
- Hack-y files and classes

## New subsystems & new dependencies

- Components & Concurrency added
- Local Storage component does not work as expected

## Concrete Architecture of Chrome has difficult modifiability

- Interfaces vs. Huge interdependent codebase
- Not layered, less modifiable



# Lessons Learned: Chrome Team Issues

Amount of interfaces creates modifiability difficulties

- Comes from thick coupling

Open Source issues expanded

- Created a huge codebase
- Some poorly documented subdirectories included

Object oriented style side effects in a huge codebase

- Next to impossible to test in the concrete architecture

# Lessons Learned: Limits of Findings

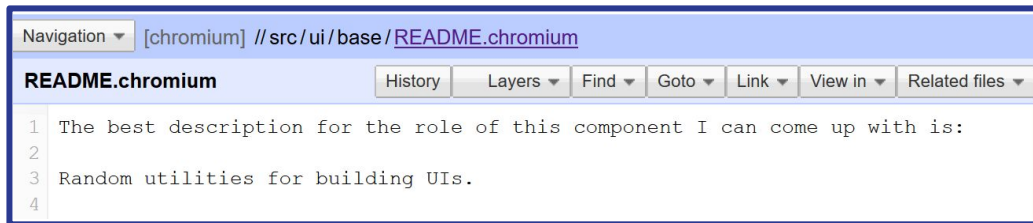
Owners of the code are listed only by emails

- Possible to search codebase for email

Comments on lots of main directories do not include README.md

- Certain README.md files are just not helpful

Understand software can be slow on such a huge codebase



The screenshot shows a code editor interface. At the top, there is a navigation bar with a dropdown menu labeled 'Navigation' and a breadcrumb path: '[chromium] //src/ui/base/README.chromium'. Below the navigation bar, the file name 'README.chromium' is displayed. To the right of the file name are several buttons: 'History', 'Layers', 'Find', 'Goto', 'Link', 'View in', and 'Related files'. The main content area shows a code snippet with line numbers 1 through 4. The code reads: '1 The best description for the role of this component I can come up with is:', '2', '3 Random utilities for building UIs.', and '4'.

# Lessons Learned: Process of Using Understand

## The Good: **What works**

- Familiar aesthetic interface
- Functional
  - Good visual representation of systems & subsystems

## The Bad: **Inconveniences**

- Long wait times
- Workarounds needed
- Exporting/Importing

## The Ugly: **Ruins functionality**

- Crashes and saving



# Lessons Learned: Our Team Issues

## Achievements:

- Regular meetings
- Individual work then discussion

## Working on:

- Better concurrency
- Working consistently





# Chrome-clusion

- Chrome is a very large piece of software making it hard to *Understand*
- Documentation was still hard to follow and dive through (occasionally no ReadMe's)
- Chrome has very high coupling within its Concrete Architecture
- Chrome's Concrete Architecture contains two more components than we previously thought
- More frequent group meetings

