

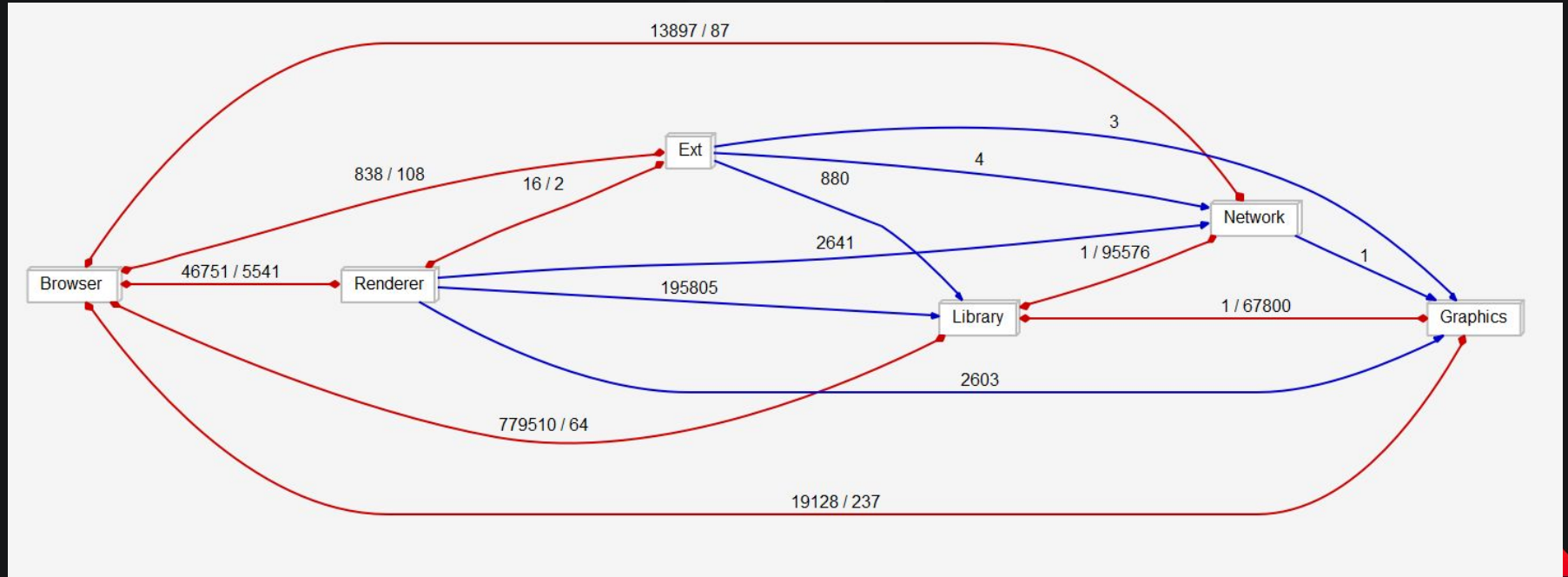


Google Chrome Concrete Architecture Presentation

By: Brynnon Picard, Dongho Han, Sam Song, Bradley Kurtz,
Alex Galbraith and Roy Griffiths

Derivation Process : Dependency Graph

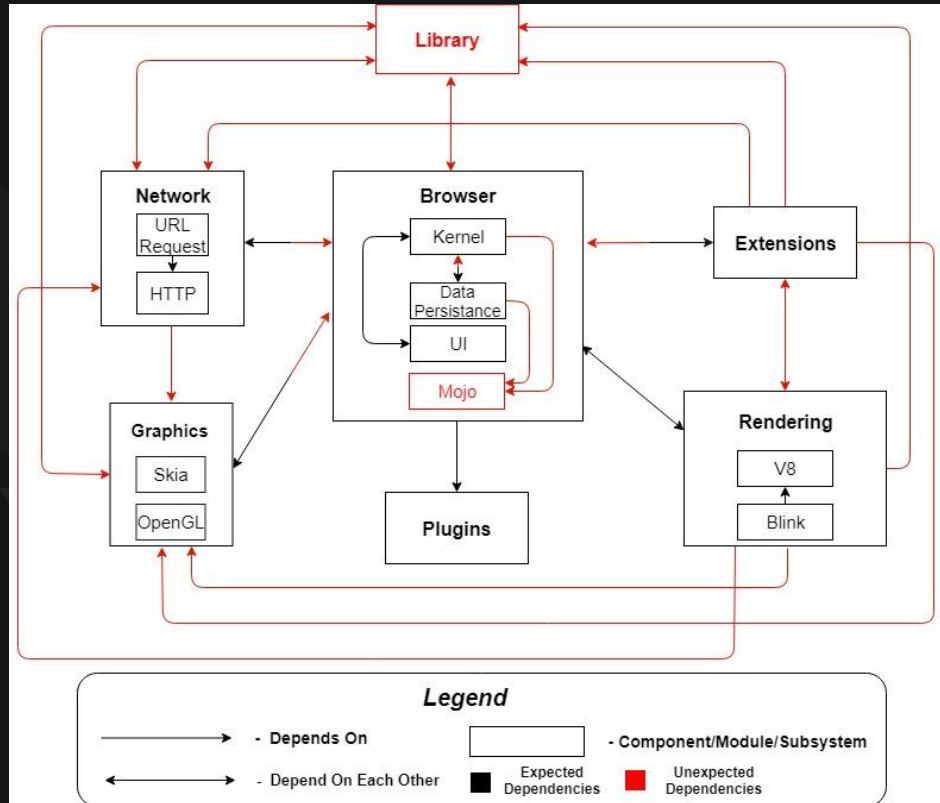
First used Understand to generate dependency graph



Derivation Process : Concrete

Next built a concrete graph and found our unexpected dependencies.

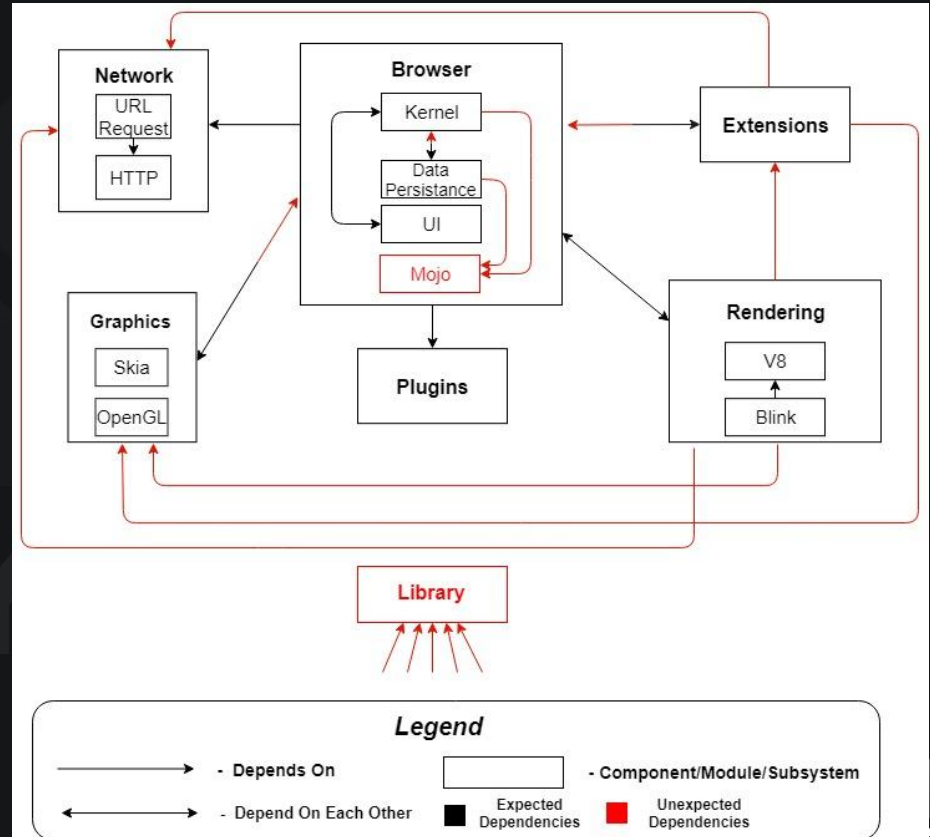
By analysing our unexpected dependencies we found and removed hacks from our concrete architecture



Derivation Process : Reflexion Analysis

This architecture is object oriented with implicit invocation and a multiprocess architecture.

Most notable differences all of libraries dependencies turned out to be understand errors or hacks, and Renderer and Extensions are much more tightly coupled to other components than we expected.



Architecture Overview

Browser: Central system that manages renderer instances and UI

Rendering: Renders pages

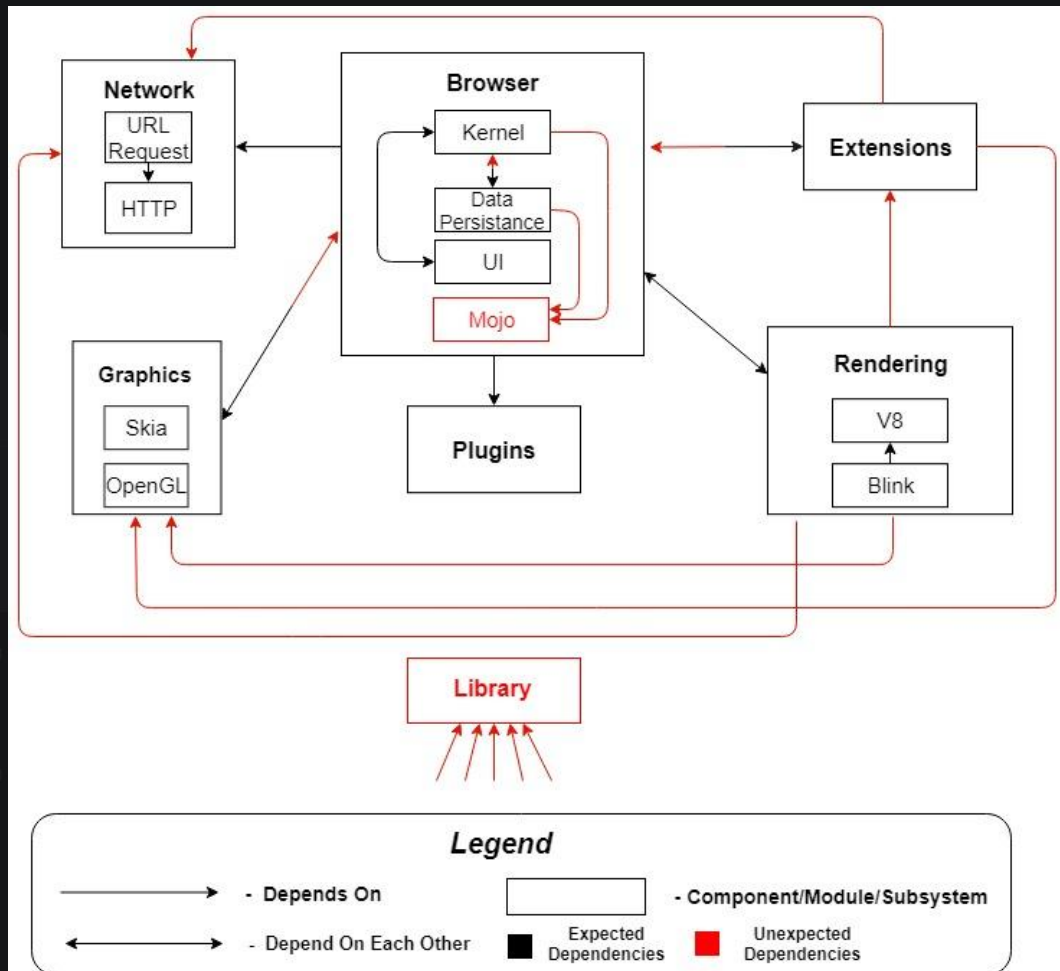
Graphics: handles low level rendering of visuals

Network: Handles network IO

Library: Shared code such as string manipulation and generic utilities

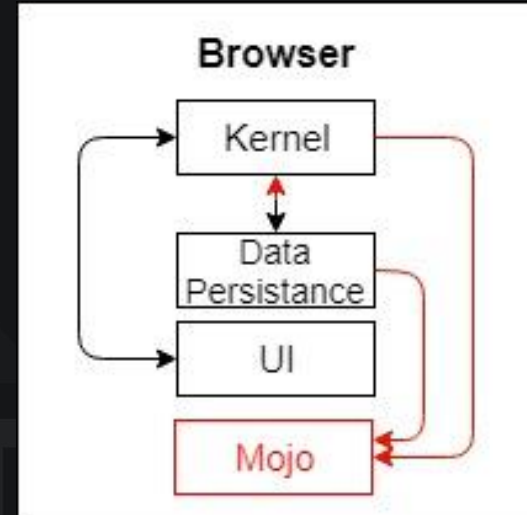
Extensions: Handles Chrome's extension system

Plugins: Handles plugins such as Flash player



Browser Subsystem

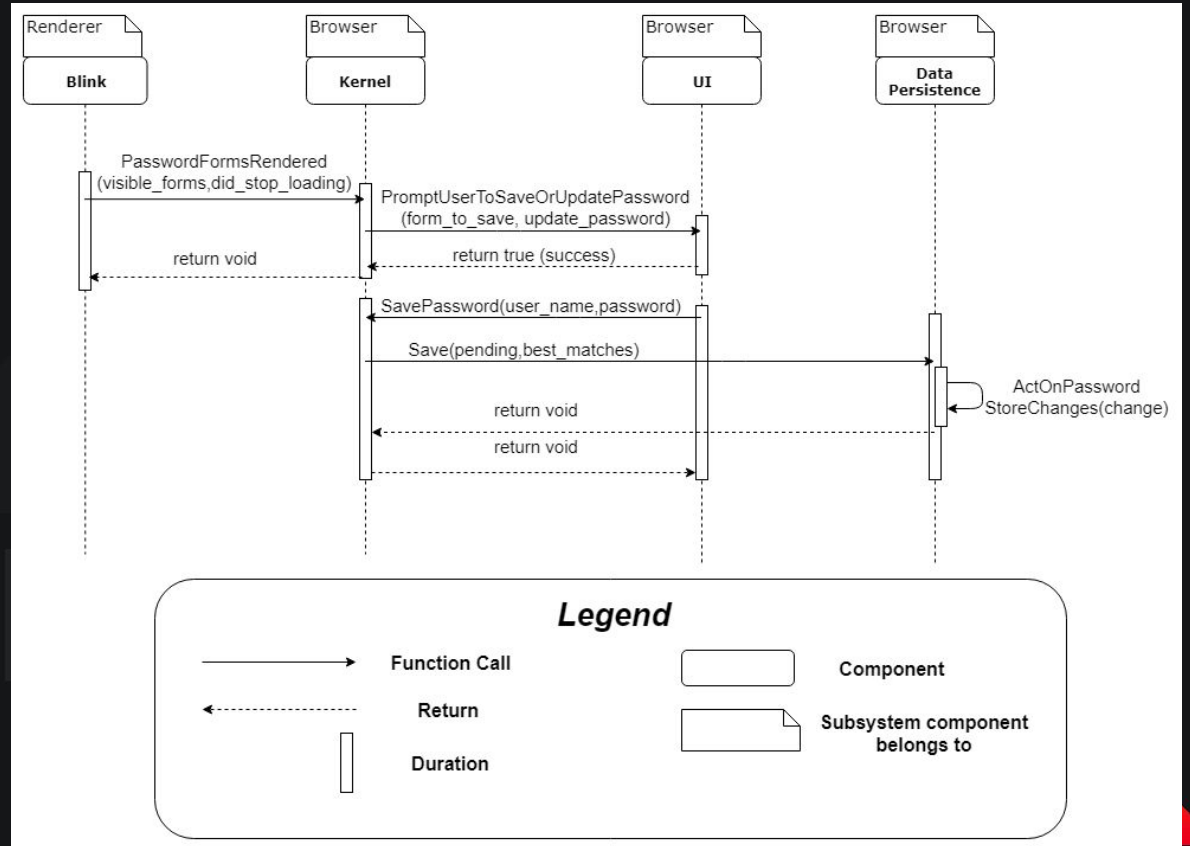
- UI handles user interface
- Data persistence stores bookmarks, passwords etc
- Mojo performs IPC
- Kernel strings everything together, is the “brain” of the browser
- DP needs Mojo to send and receive certain info.
- Kernel uses Mojo to communicate.
- DP relies on kernel to notify it of changes to the data store
- UI relies on Kernel to pass UI events to the Kernel
- Kernel relies on UI to tell UI what to display
- Kernel relies on DP to retrieve user data



Sequence Diagram (User Sign in and Credential Store)

Notable features:

- *PasswordFormsRendered* is triggered via the event system when a form is submitted in the Renderer.
- *PromptUserToSaveOrUpdatePassword* triggers the creation of the UI popup dialogue which is the end of the first call stack.
- A user click event triggers *SavePassword* from the UI dialogue box



Reflexion Analysis (Unexpected Dependencies)

New Component

- Library is depended on by everything

Interesting Dependencies

- Graphics -> Browser
 - *buffer_queue.cc* relies on *display_snapshot.h*
- Extensions -> Network
 - *url_handlers_parser.cc* relies on *net/base/network_change_notifier.h*
- Library -> Graphics
 - *shim_override_glibc_weak_symbols.h* relies on *components/viz/common/features.h*

(New) Relationships:

- Extensions -> Network
 - > Browser
 - > Graphics
- Rendering -> Extensions
 - > graphic
 - > Network
- Graphics -> Browser
- Data persistence -> Kernel
- All subsystem -> Library



Extension - Renderer Dependency

- Extensions -> Renderer
 - *chrome_extensions_client.cc* relies on *url_constants.h*
- A hack
- Stores links
- Code-reuse

```
5 // Contains constants for known URLs and portions thereof.
```

```
108
109 // "Learn more" URL for "Aw snap" page when showing "Reload" button.
110 extern const char kCrashReasonURL[];
111
112 // "Learn more" URL for "Aw snap" page when showing "Send feedback" button.
113 extern const char kCrashReasonFeedbackDisplayedURL[];
114
115 // "Learn more" URL for the "Do not track" setting in the privacy section.
116 extern const char kDoNotTrackLearnMoreURL[];
117
118 // The URL for the "Learn more" page for interrupted downloads.
119 extern const char kDownloadInterruptedLearnMoreURL[];
120
121 // The URL for the "Learn more" page for download scanning.
122 extern const char kDownloadScanningLearnMoreURL[];
123
```



Concurrency

- Looking through the code confirmed our previous assessment of Chrome's concurrency:
 - Renderers run in their own isolated process, communicating with the browser via IPC
 - Mojo sub-subsystem in Browser handles IPC communication between processes
- There are many files and dependencies that supported inter-process communication between various subsystems such as:
 - Mojo subfolder in Renderer subsystem provides hooks into the mojo IPC library
 - Mojo subfolders throughout the browser that facilitate IPC

SKYNET



Team issues

- A large team with many developers located in different locations.
- Unable to know who programmed which sections.
- Some developers commented their code well, while some developers don't seem to write many comments at all.
- Developers add many tests in the code.

```
TEST_F(LoginDatabaseIOSTest, UpdateLogin) {  
    PasswordForm form;  
    form.origin = GURL("http://0.com");  
    form.signon_realm = "http://www.example.com";  
    form.action = GURL("http://www.example.com/action");  
    form.password_element = base::ASCIIToUTF16("pwd");  
    form.password_value = base::ASCIIToUTF16("example");  
  
    ignore_result(login_db_->AddLogin(form));  
  
    form.password_value = base::ASCIIToUTF16("secret");
```

```
1: AddPattern(&shost_perm_set1_, "http://reddit.com/r/test/*");
```



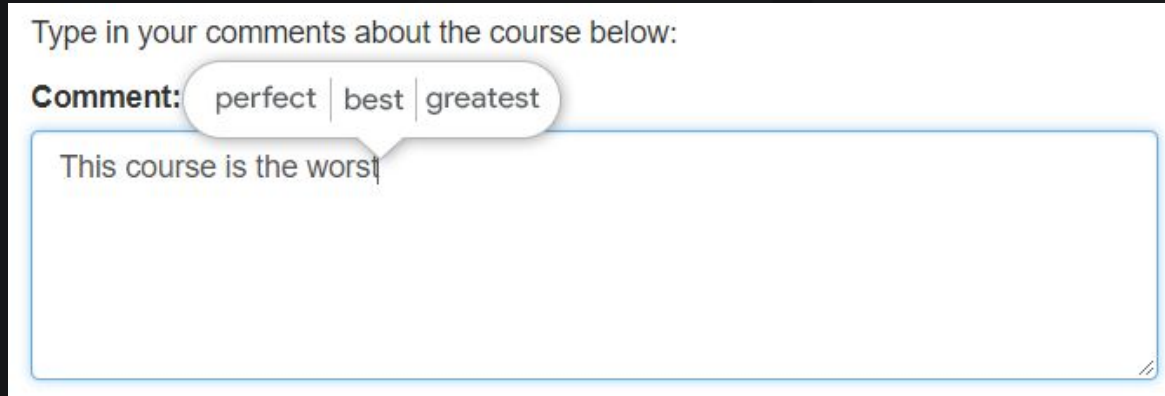
Proposed Feature for A3

- When a user is typing in an input field it will suggest words for them to use, similar to the keyboard suggestion feature available on android or iOS
- Would involve Renderer interacting with Browser to retrieve suggested words, possibly into data persistence to look for words commonly used by the user to increase personalization

Type in your comments about the course below:

Comment: perfect | best | greatest

This course is the worst

A screenshot of a web form for submitting comments. At the top, it says "Type in your comments about the course below:". Below that is a label "Comment:" followed by a dropdown menu showing three suggestions: "perfect", "best", and "greatest". The dropdown is open, and the suggestions are displayed in a light blue rounded rectangle. Below the dropdown is a large text input area with a light blue border. The text "This course is the worst" is typed into the input area. The input area has a small cursor at the end of the text.

Lessons learned/limitations

- Understand is not a very stable program, and is very resource intensive, which made it impossible for some group members to run the software, and thus made it difficult to spread out work evenly.
- Understand would often pick the wrong file as a dependency if there were multiple files with the same name.
- Understand would often crash when trying to inspect specific files.
- Learned to navigate the cs.chromium website. (Follow where a method call goes, and where methods are called from)
- Since some source code has been purposely removed, it made it impossible to verify Plugins dependencies.
- Our team was not experienced with C++, making it harder to understand the code



Conclusion

- Source code for Chrome is extremely large – impossible to review the whole code architecture.
- Confusing to work through many dependencies in different subsystems.
- Very little documentation of the function of each file.
- Architecture is an object oriented multi process architecture with implicit invocation for IPC (Inter-Process-Communication).

SKYNET

