Firefox Enhancement Proposal

By: The Other Group

Overview

- Enhancement Possibilities
- Approach Chosen
- Advantages of Approach
- Disadvantages of Approach
- Limitations
- Lessons Learned

Problems in the Firefox Architecture

- Single-processed environment.
  - If any particular tab/website crashes, so does the browser
  - Performance is directly correlated to CPU speed and load, does not scale with multi-processors
    - Industry focus is currently on CPUs with more parallel processing capability than raw processing power – lots of performance to find here
  - Not as secure (sandboxing)

- Layers contain significant amounts of inter-dependencies
  - Components are not easily swappable (one of the main advantages of OO style)
  - Simple modifications to components might require multiple modifications in other components
  - Unnecessary complexity between components

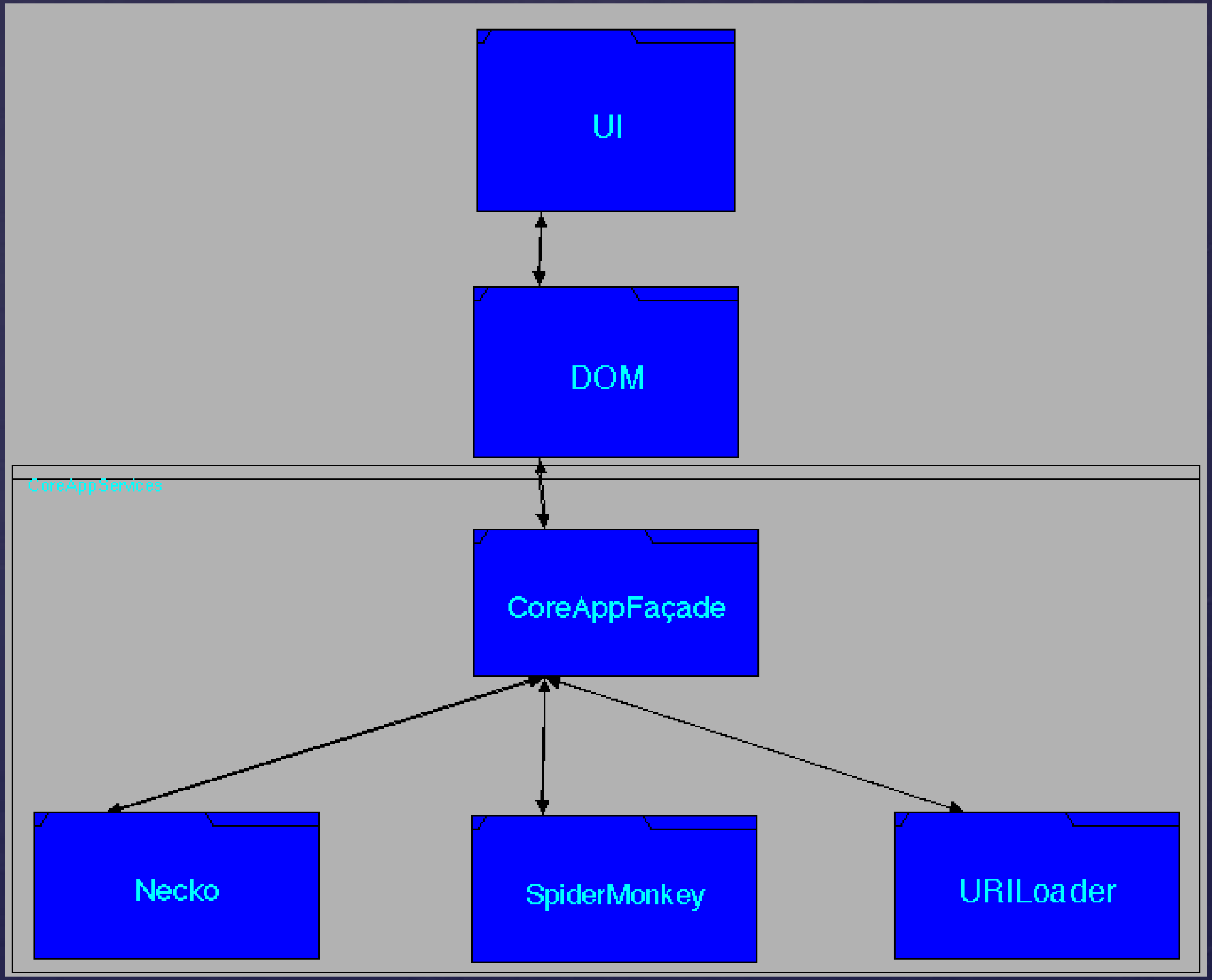
Enhancement Possibilities

- Multi-Process
  - + Better performance scaling
  - + More efficient use of available resources
  - - More complexity

- Create a Façade for the Core App Services Layer
  - + Greatly reduced complexity
  - + Roles of individual components become much more clear
  - - Introduces performance overhead
  - - Extending functionality means making changes in the class and in the façade

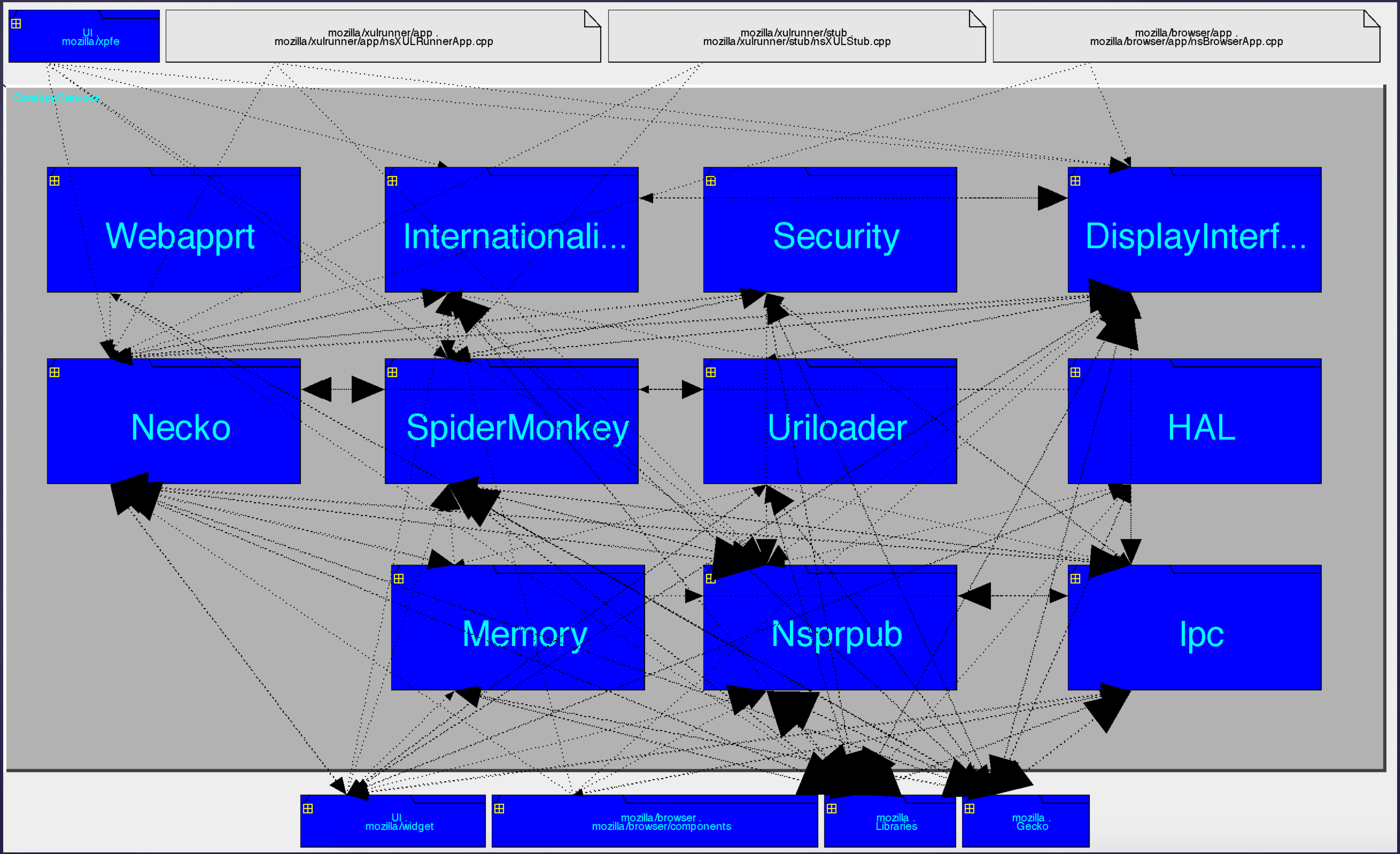
Core-App Services Façade

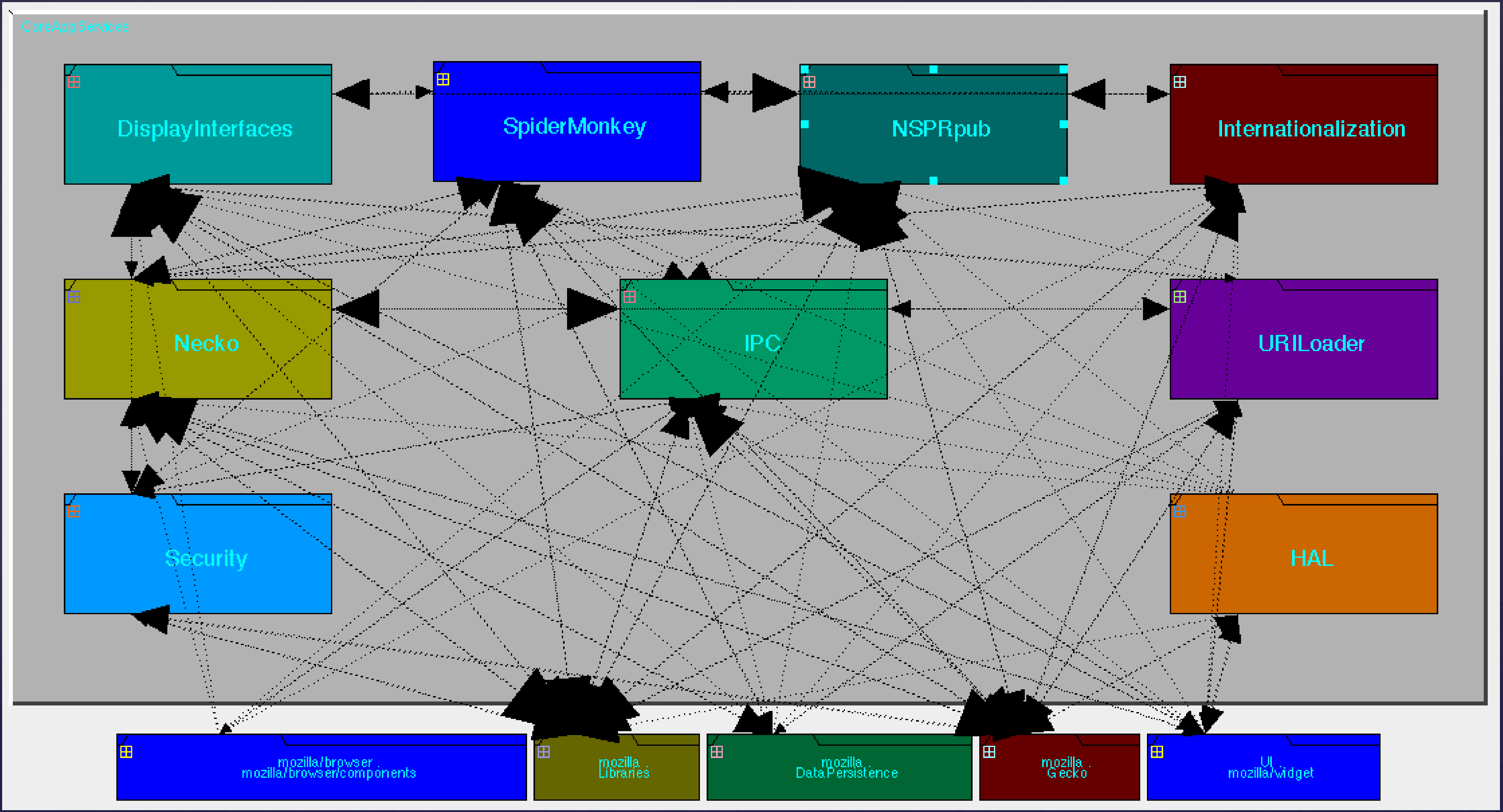
- Create a component that acts as an interface between Gecko or the UI Layer and Core-App Services
- No component outside of Core App Services should talk to any other component but the interface
- The interface should pass along requests and commands to the appropriate components – without the client having to know where or how these requests are handled



Existing Dependencies

- Most dependencies within the Core App Services Layer are:
  - For conversions such as
    - Number conversions (calls made to IPC)
    - Unicode conversions (calls made to intl)
    - Time conversions (calls made to HAL)
  - Calling methods that can be implemented by the calling class
  - Due to nature of the module, convenience (e.g. Necko)


UI
mozilla/xpfe
mozilla/xulrunner/app .
mozilla/xulrunner/app/nsXULRunnerApp.cpp
mozilla/xulrunner/stub .
mozilla/xulrunner/stub/nsXULStub.cpp
mozilla/browser/app .
mozilla/browser/app/nsBrowserApp.cpp
CoreAppServices
Webapprt
Internationali...
Security
DisplayInterf...
Necko
SpiderMonkey
Uriloader
HAL
Memory
Nsprpub
Ipc
UI .
mozilla/widget
mozilla/browser .
mozilla/browser/components
mozilla .
Libraries
mozilla .
Gecko


CoreAppServices
DisplayInterfaces
SpiderMonkey
NSPRpub
Internationalization
Necko
IPC
URILoader
Security
HAL
mozilla/browser . mozilla/browser/components
mozilla . Libraries
mozilla . DataPersistence
mozilla . Gecko
UI . mozilla/widget

Impact of chosen feature

- Firefox is an open-source project. Thousands of developers will join and leave the project for varying amounts of time. It is important that the there are as few barriers to entry as possible to attract more developers
- In class it was mentioned that there was a time when Firefox was grossly understaffed. They undertook a large project over the course of two years to improve their documentation and succesfully piqued the interest of the developer community
  - The same must happen for their architecture!

Impact of chosen feature

- Having a simple architecture focusing on utilizing a Façade allows developers with small ideas to quickly jump in and add their improvements without having to deal with dependency resolution across multitudes of files
- This would significantly decrease time for fixes and additional feature creation
- Components would now be easily pluggable / replaceable
  - Multi-processing could be done more easily by spawning a different process for each component, with a single interface managing them all
- This ideally would eventually extend toward the other two layers as well

Impact of chosen feature

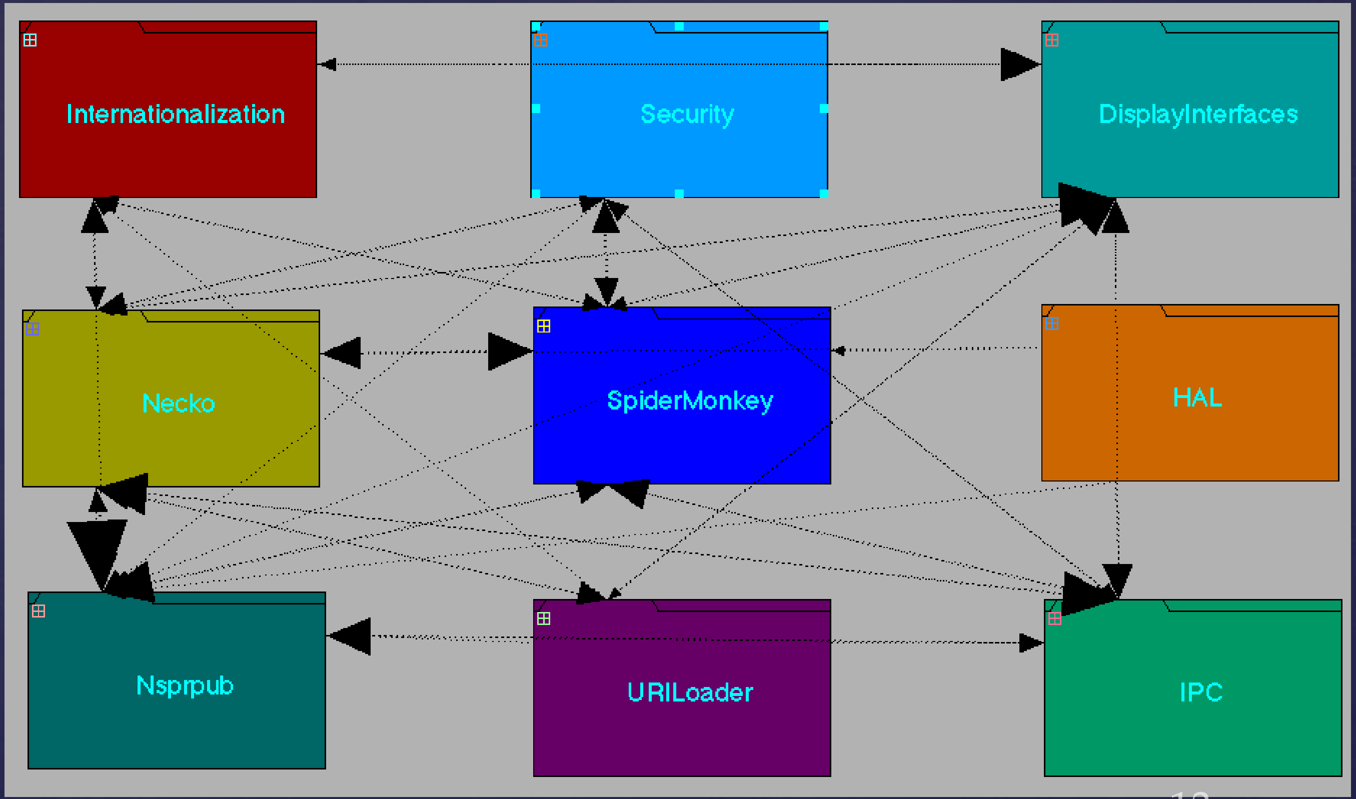
- Potential stakeholders
  - Mozilla Developers (both seasoned and new)
  - Carriers which offer Firefox OS
  - Developers for various other Mozilla projects (Thunderbird)
  - Users of various Mozilla Framework technologies
  - Various product owners (e.g. Dave Camp for Firefox, Fabrice Desré for FirefoxOS)[2]

Architectural Changes

- Creation of structural façade code (function definitions etc.)
- Much of Core App Services would need to be rewritten (especially components which communicate with the network)
- Some of the DOM layer interfacing would need to be modified

Pros of chosen feature

- Elimination of cross-dependencies
- Ease of creation for unit tests
- Minimize potential surface area for bugs



Cons of chosen feature

- Potential performance overhead (symbol table while compiling takes up more space, and execution takes longer because of late binding)
- Developer energy investment into changes that only serve to simplify the structure, but will not actually contribute to the project in a meaningful way (e.g. Necko)
- Testing may ultimately be more difficult[1]

Limitation of Work Findings

- Difficult simulate the system we are thinking of and implementing it would cost too much time
- Only used source code and documentation to make assumptions, since this has not been investigated publicly among developers

Lessons Learned

- Designing and implementing is a fight between convenience and clean code
- Real-world costs of theoretical changes are hard to estimate
- Without proper introspection, original design decisions get lost

References

1. http://programmingarehard.com/2014/01/11/stop-using-facades.html/
2. https://wiki.mozilla.org/Modules/All