Firefox Enhancement Proposal

{ By: The Other Group

Overview

- Problems in the Current Firefox Architecture
- Enhancement Possibilities
- Approach Chosen
- Advantages and Disadvantages of Approach
- Impact 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
  - Not as secure (sandboxing)

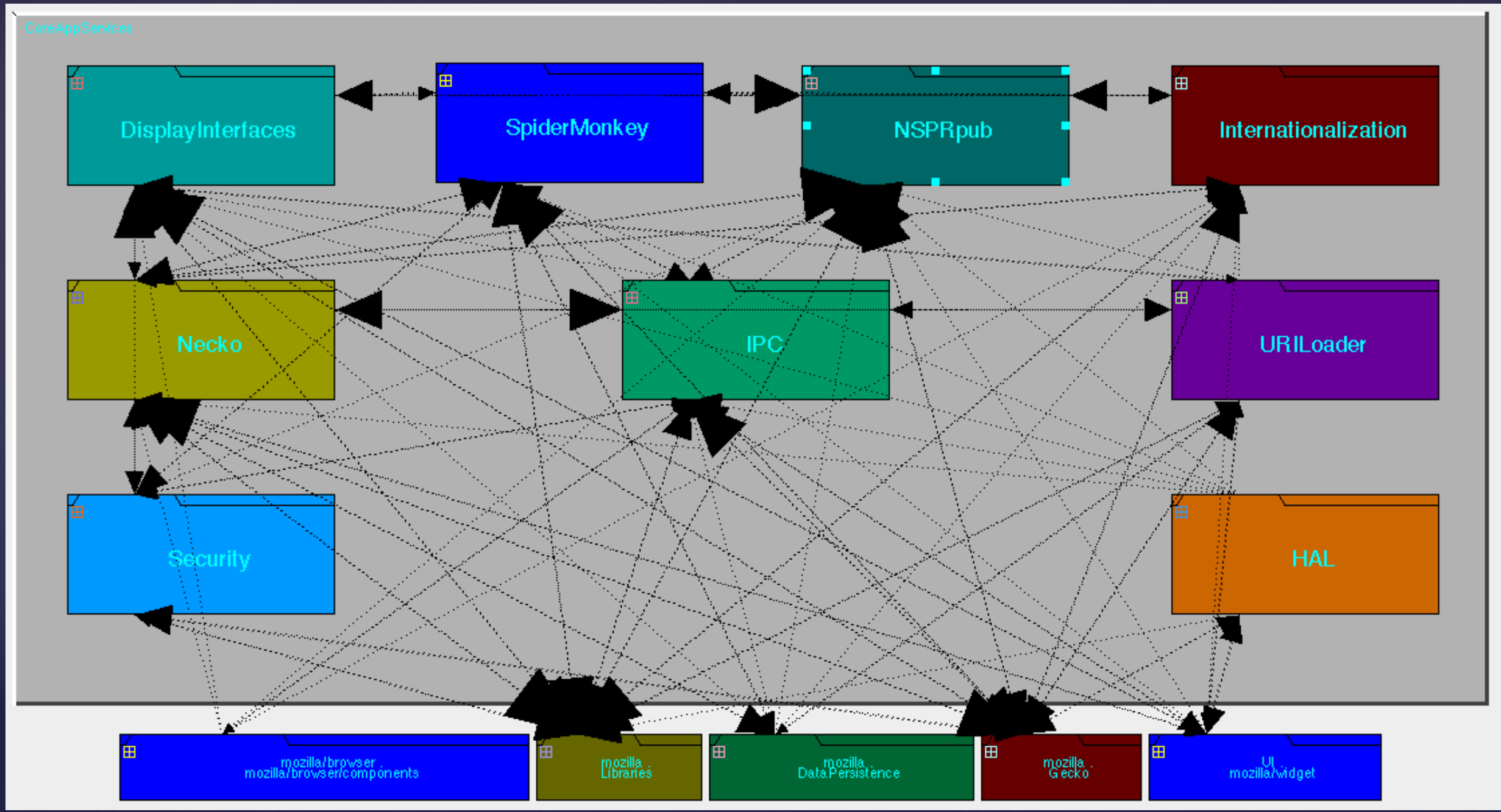
- Layers contain significant amounts of inter-dependencies
  - Components are not easily swappable (one of the main advantages of OO style)
  - Simple modifications to components might require multiple modifications in other components
  - Unnecessary complexity between components

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

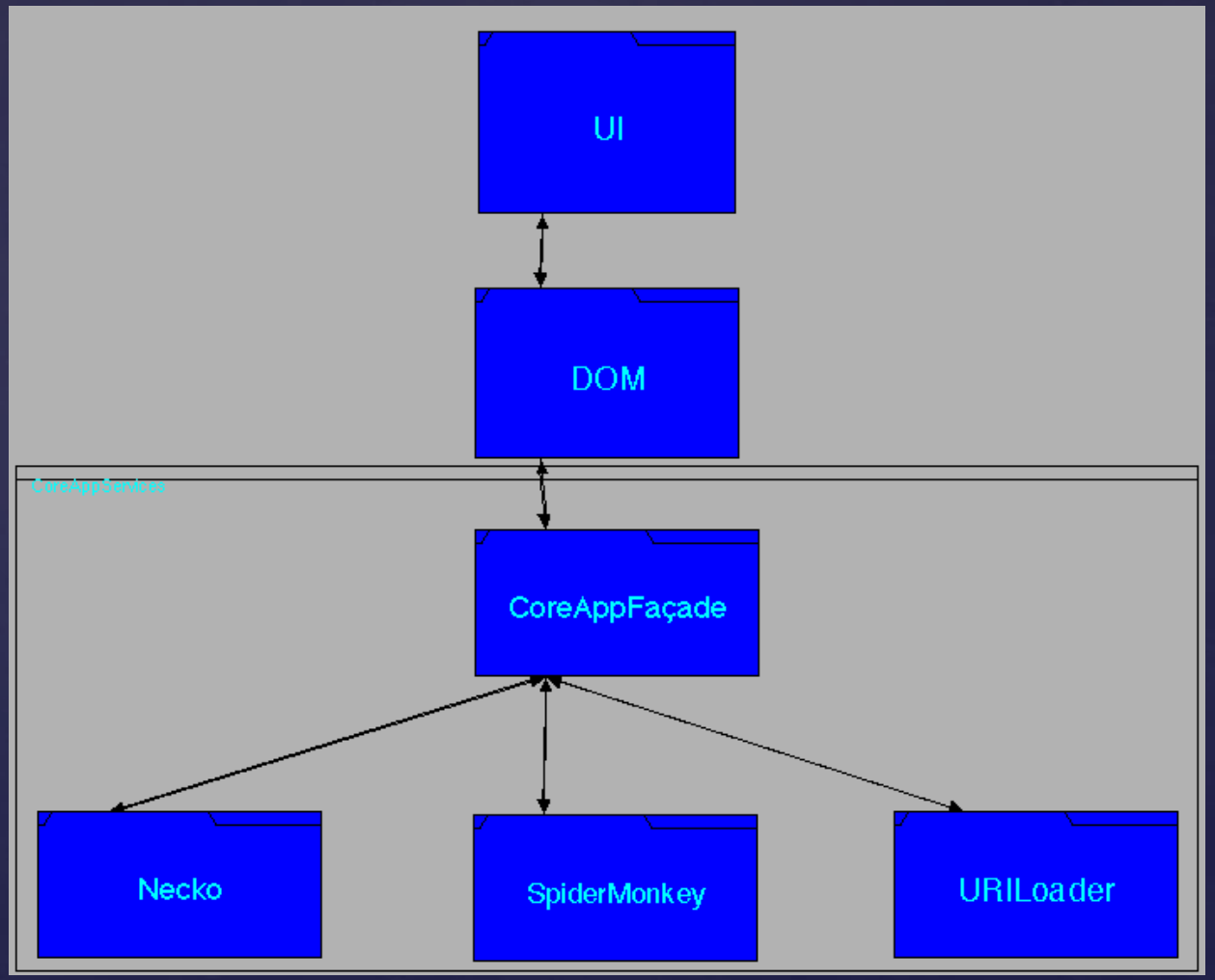
Existing Dependencies

- 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)


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

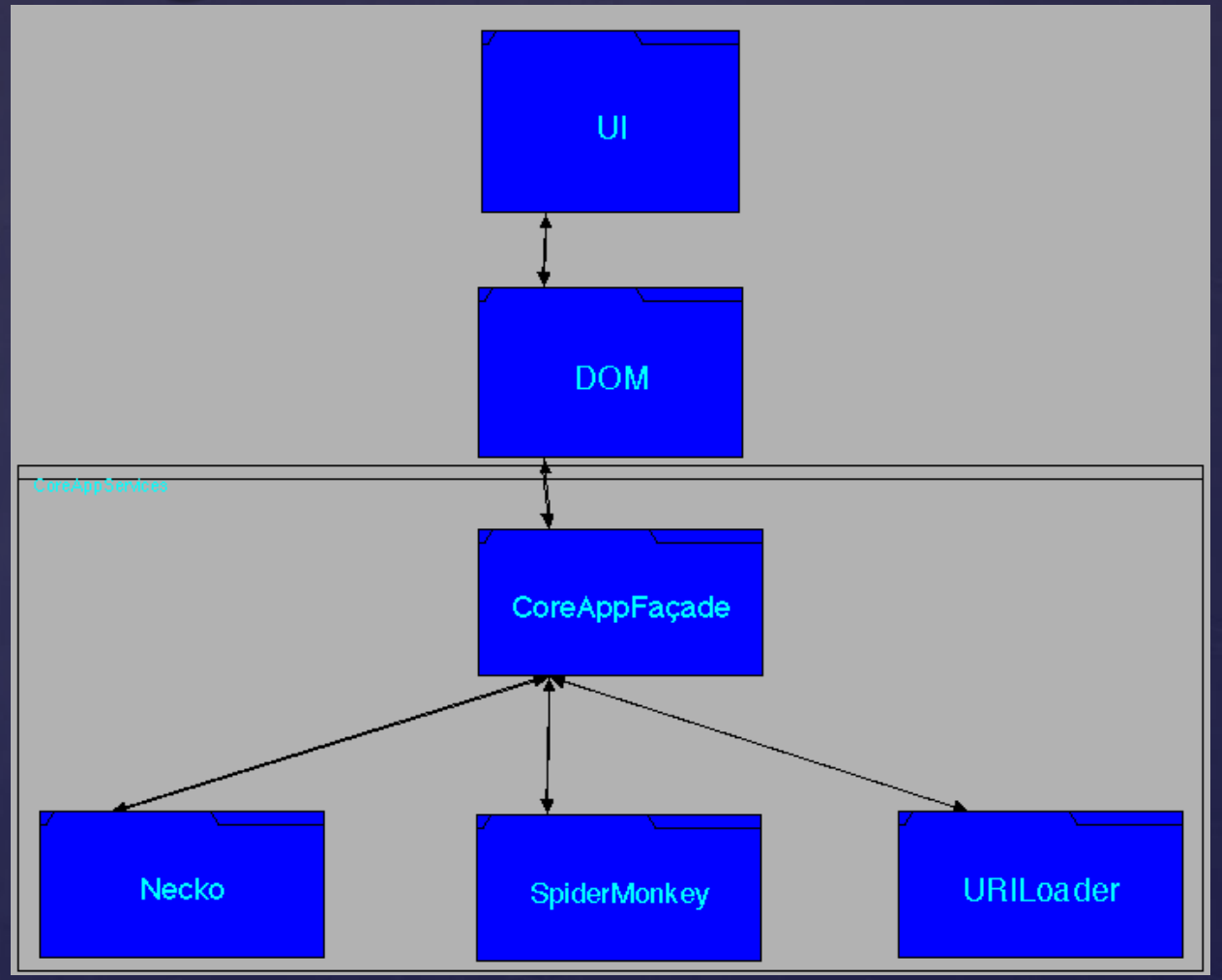
Core-App Services Façade

- Create a component that acts as a go between DOM layer and CoreAppServices
- Restrict communication between components
- Control all the communication both in and out of CoreAppServices



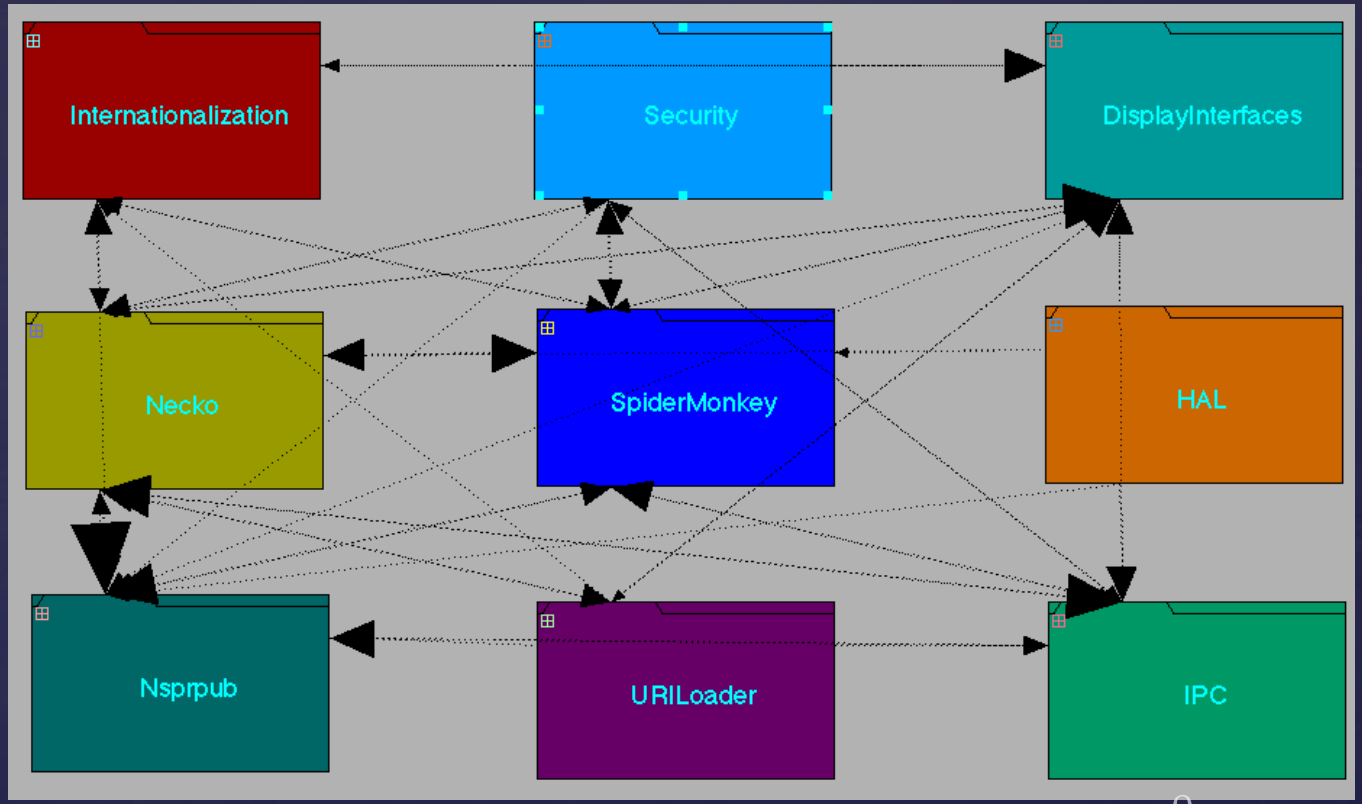
Architectural Changes

- 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)
- Certain types of testing may ultimately be more difficult[1]

Impact of chosen feature

- Firefox is open source, many short-term developers
- Firefox developers took a serious commitment to update documentation a few years ago in order to attract more volunteer developers
- Architecture being confusing may be holding back even more aid

Impact of chosen feature

- Developers with small ideas can quickly jump in and add their improvements without having to deal with dependency resolution across multitudes of files
- 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
- Possible: This would 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]

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