# Youtube link:

https://youtu.be/BuaiUdU6GTU

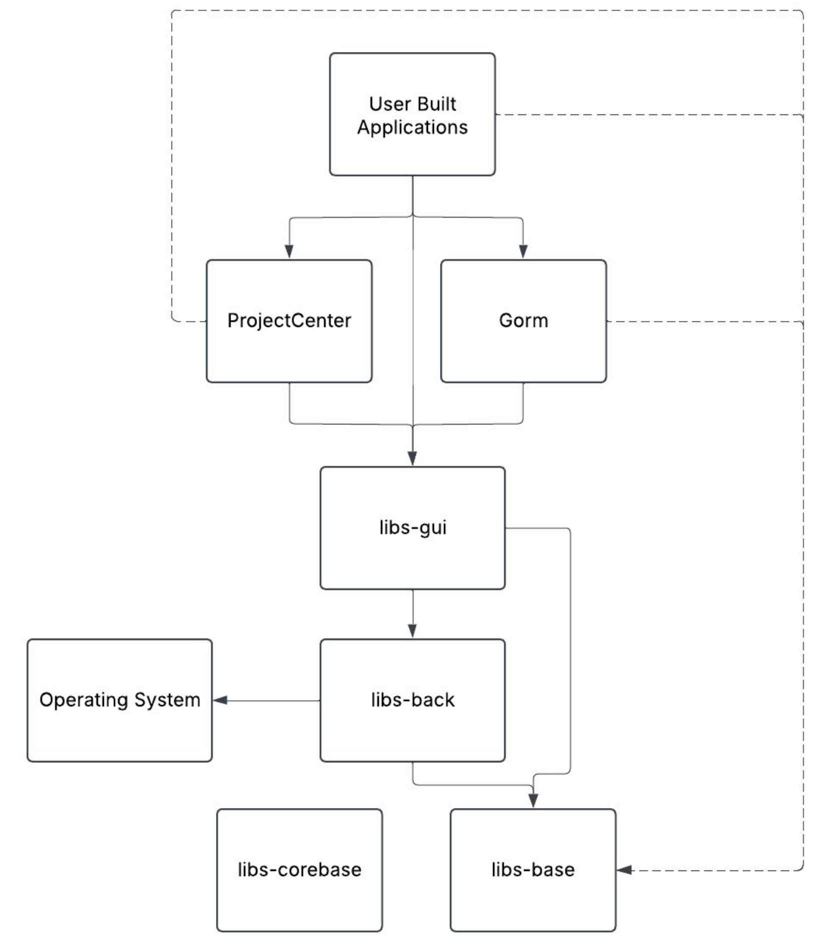
# GNUstep Concrete Architecture

**Liam Beenken:** Presenter, Divergence Analysis
**Cameron Jenkins:** Presenter, Subsystem Analysis
**Evelyn Lee:** Leader**,** abstract, Derivation Process
**Christine Ye:** Introduction, Top-Level Concrete Architecture
**Yiting Ma:** Sequence Diagrams, Lessons Learned

## Conceptual Architecture

## Conceptual Architecture

- Layered
- Object-Oriented
- Hierarchy skips for efficiency

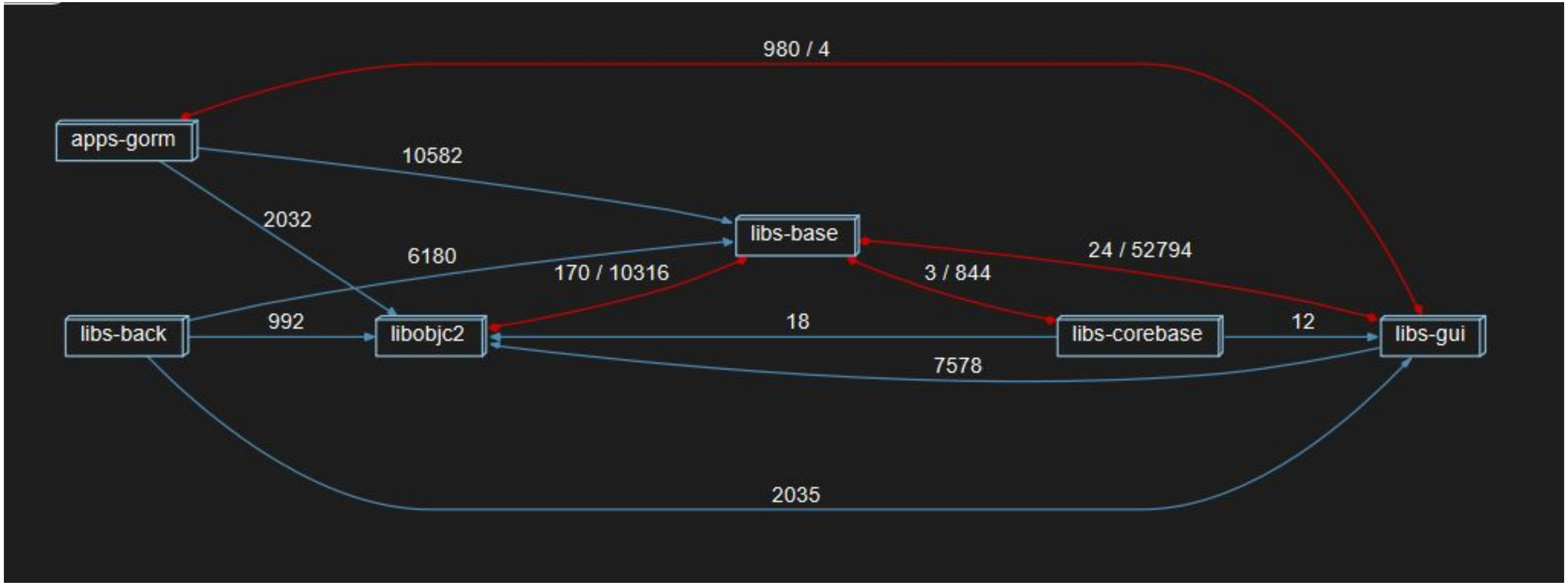


## Concrete Architecture

## Derivation

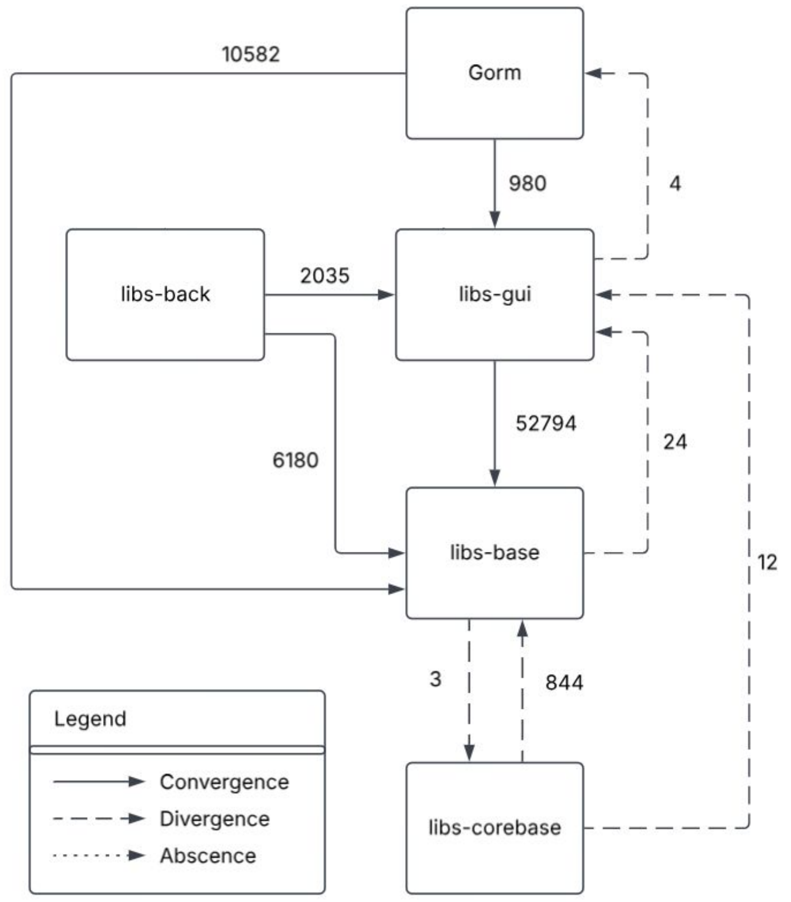
- Source code examined with Understand
- Dependency structure built and analyzed
- Conceptual and concrete architectures compared via reflexion analysis
- Divergences investigated with the Sticky Note method

# Concrete Architecture



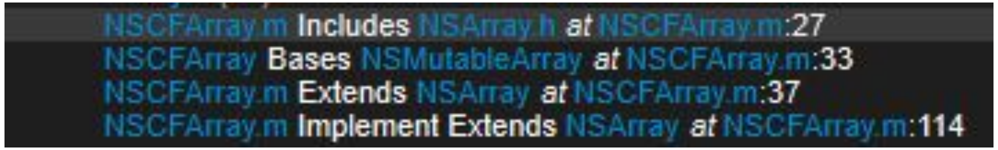
## Reflexion Diagram

- Comparison between conceptual and concrete architectures
- Convergences match conceptual dependencies
- Divergences and Absences are unexpected

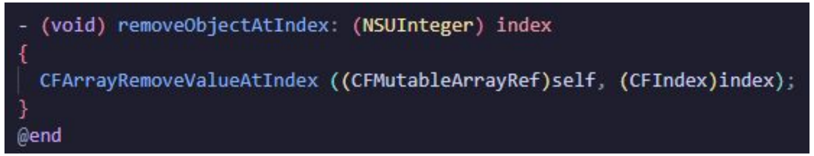


## Divergence Analysis

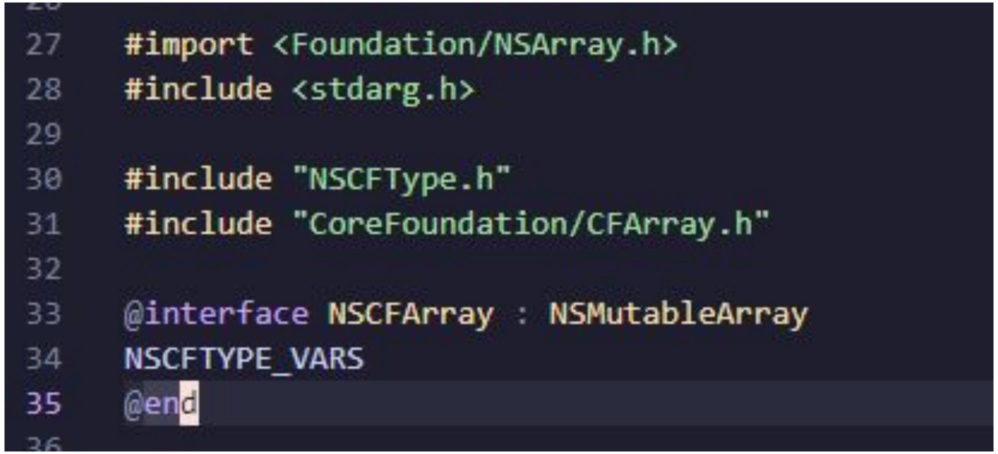
# Divergence 1. (libs-corebase → libs-base)



```
NSCFArray.m Includes NSArray.h at NSCFArray.m:27
NSCFArray Bases NSMutableArray at NSCFArray.m:33
NSCFArray.m Extends NSArray at NSCFArray.m:37
NSCFArray.m Implement Extends NSArray at NSCFArray.m:114
```



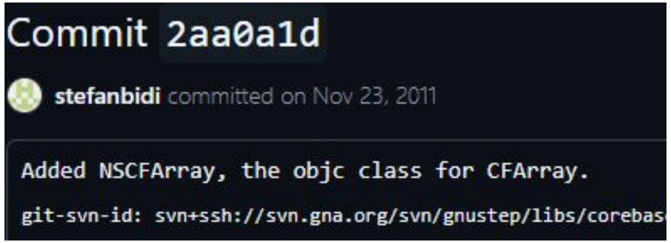
```
- (void) removeObjectAtIndex: (NSUInteger) index
{
  CFArrayRemoveValueAtIndex ((CFMutableArrayRef)self, (CFIndex)index);
}
@end
```



```
#import <Foundation/NSArray.h>
#include <stdarg.h>

#include "NSCFType.h"
#include "CoreFoundation/CFArray.h"

@interface NSCFArray : NSMutableArray
NSCFTYPE_VARS
@end
```



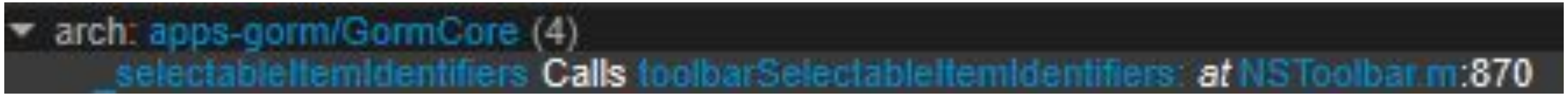
Commit 2aa0a1d

stefanbidi committed on Nov 23, 2011

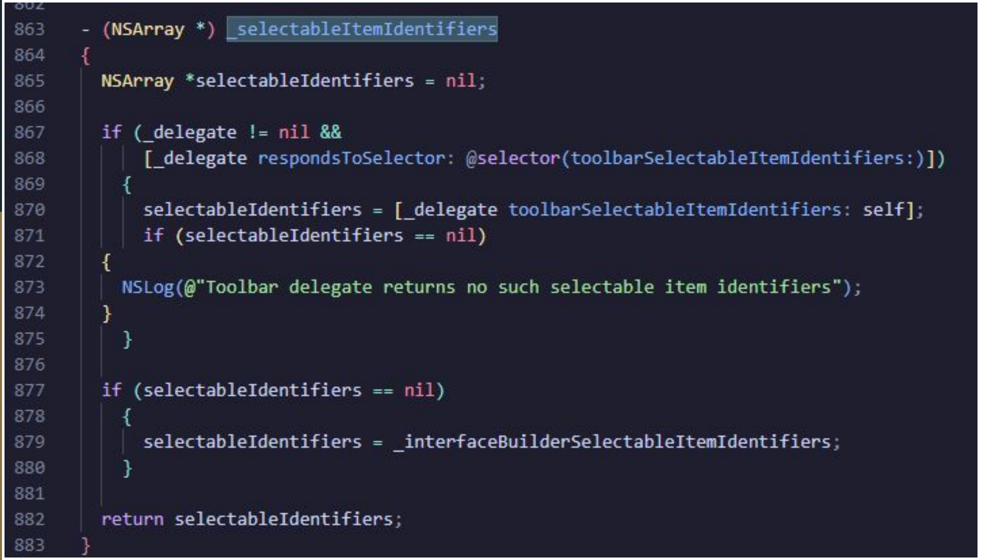
```
Added NSCFArray, the objc class for CFArray.

git-svn-id: svn+ssh://svn.gna.org/svn/gnustep/libs/corebas
```

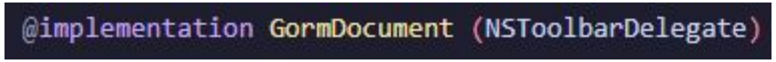
# Divergence 2. (libs-gui → GORM)



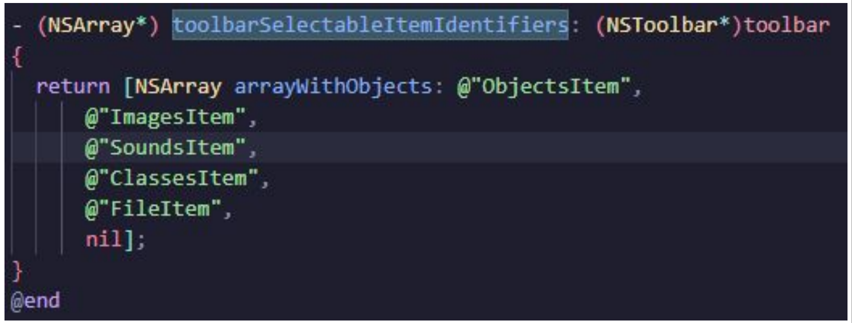
```
▾ arch: apps-gorm/GormCore (4)
    _selectableItemIdentifiers Calls toolbarSelectableItemIdentifiers: at NSToolbar.m:870
```



```
863  - (NSArray *) _selectableItemIdentifiers
864  {
865    NSArray *selectableIdentifiers = nil;
866
867    if (_delegate != nil &&
868        [_delegate respondsToSelector: @selector(toolbarSelectableItemIdentifiers:)])
869      {
870        selectableIdentifiers = [_delegate toolbarSelectableItemIdentifiers: self];
871        if (selectableIdentifiers == nil)
872    {
873      NSLog(@"Toolbar delegate returns no such selectable item identifiers");
874    }
875      }
876
877    if (selectableIdentifiers == nil)
878      {
879        selectableIdentifiers = _interfaceBuilderSelectableItemIdentifiers;
880      }
881
882    return selectableIdentifiers;
883  }
```

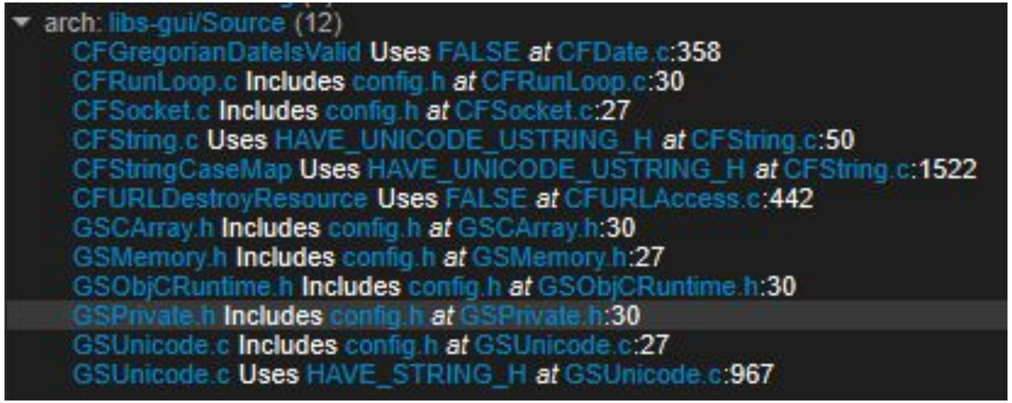


```
@implementation GormDocument (NSToolbarDelegate)
```

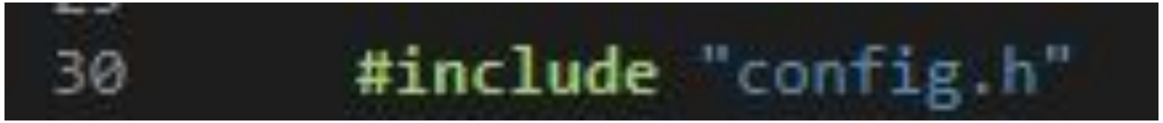


```
- (NSArray*) toolbarSelectableItemIdentifiers: (NSToolbar*)toolbar
{
  return [NSArray arrayWithObjects: @"ObjectsItem",
      @"ImagesItem",
      @"SoundsItem",
      @"ClassesItem",
      @"FileItem",
      nil];
}
@end
```

## Divergence 3.  (libs-corebase → libs-gui)

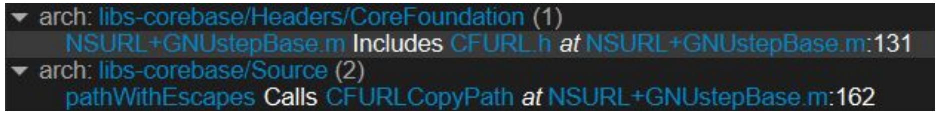


```
arch: libs-gui/Source (12)
    CFGregorianDateIsValid Uses FALSE at CFDate.c:358
    CFRunLoop.c Includes config.h at CFRunLoop.c:30
    CFSocket.c Includes config.h at CFSocket.c:27
    CFString.c Uses HAVE_UNICODE_USTRING_H at CFString.c:50
    CFStringCaseMap Uses HAVE_UNICODE_USTRING_H at CFString.c:1522
    CFURLDestroyResource Uses FALSE at CFURLAccess.c:442
    GSCArray.h Includes config.h at GSCArray.h:30
    GSMemory.h Includes config.h at GSMemory.h:27
    GSObjCRuntime.h Includes config.h at GSObjCRuntime.h:30
    GSPrivate.h Includes config.h at GSPrivate.h:30
    GSUnicode.c Includes config.h at GSUnicode.c:27
    GSUnicode.c Uses HAVE_STRING_H at GSUnicode.c:967
```

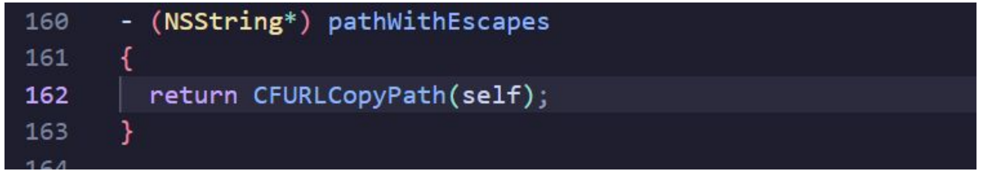


```
#include "config.h"
```

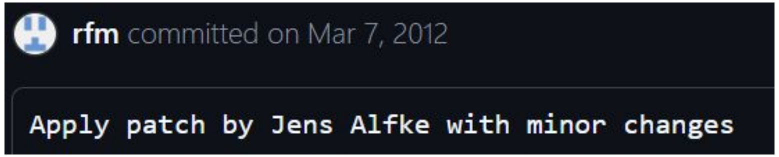
## Divergence 4. (libs-base → libs-corebase)



```
▾ arch: libs-corebase/Headers/CoreFoundation (1)
    NSURL+GNUstepBase.m Includes CFURL.h at NSURL+GNUstepBase.m:131
▾ arch: libs-corebase/Source (2)
    pathWithEscapes Calls CFURLCopyPath at NSURL+GNUstepBase.m:162
```

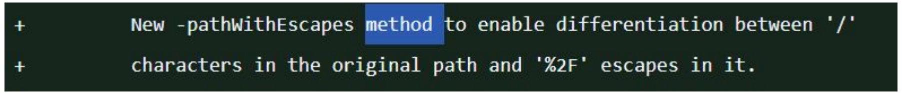


```
160   - (NSString*) pathWithEscapes
161   {
162     return CFURLCopyPath(self);
163   }
```



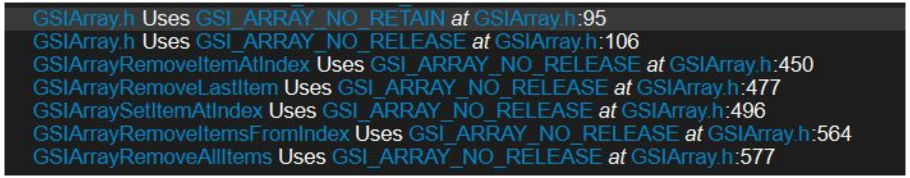
rfm committed on Mar 7, 2012

```
Apply patch by Jens Alfke with minor changes
```

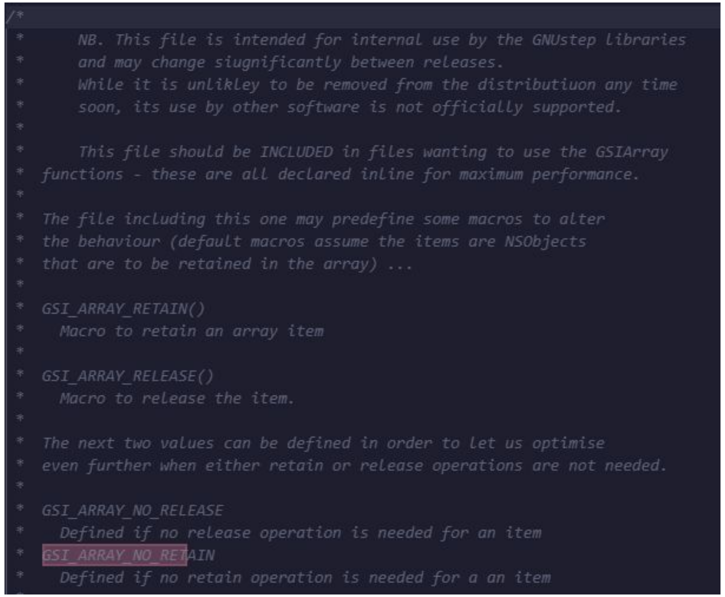


```
+          New -pathWithEscapes method to enable differentiation between '/'
+          characters in the original path and '%2F' escapes in it.
```

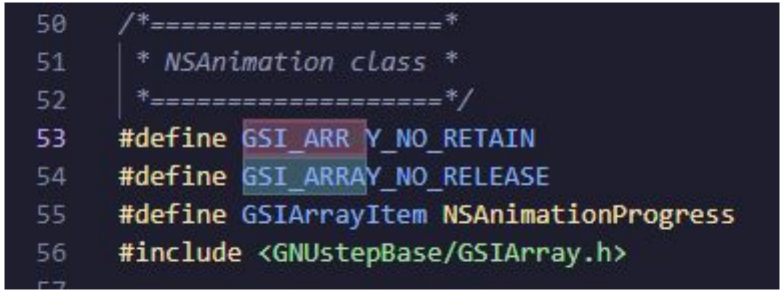
## Divergence 5.  (libs-base ⟶ libs-gui)



```
GSIArray.h Uses GSI_ARRAY_NO_RETAIN at GSIArray.h:95
GSIArray.h Uses GSI_ARRAY_NO_RELEASE at GSIArray.h:106
GSIArrayRemoveItemAtIndex Uses GSI_ARRAY_NO_RELEASE at GSIArray.h:450
GSIArrayRemoveLastItem Uses GSI_ARRAY_NO_RELEASE at GSIArray.h:477
GSIArraySetItemAtIndex Uses GSI_ARRAY_NO_RELEASE at GSIArray.h:496
GSIArrayRemoveItemsFromIndex Uses GSI_ARRAY_NO_RELEASE at GSIArray.h:564
GSIArrayRemoveAllItems Uses GSI_ARRAY_NO_RELEASE at GSIArray.h:577
```



```
/*
 *     NB. This file is intended for internal use by the GNUstep libraries
 *     and may change siugnificantly between releases.
 *     While it is unlikley to be removed from the distributiuon any time
 *     soon, its use by other software is not officially supported.
 *
 *     This file should be INCLUDED in files wanting to use the GSIArray
 * functions - these are all declared inline for maximum performance.
 *
 * The file including this one may predefine some macros to alter
 * the behaviour (default macros assume the items are NSObjects
 * that are to be retained in the array) ...
 *
 * GSI_ARRAY_RETAIN()
 *   Macro to retain an array item
 *
 * GSI_ARRAY_RELEASE()
 *   Macro to release the item.
 *
 * The next two values can be defined in order to let us optimise
 * even further when either retain or release operations are not needed.
 *
 * GSI_ARRAY_NO_RELEASE
 *   Defined if no release operation is needed for an item
 * GSI_ARRAY_NO_RETAIN
 *   Defined if no retain operation is needed for a an item
```

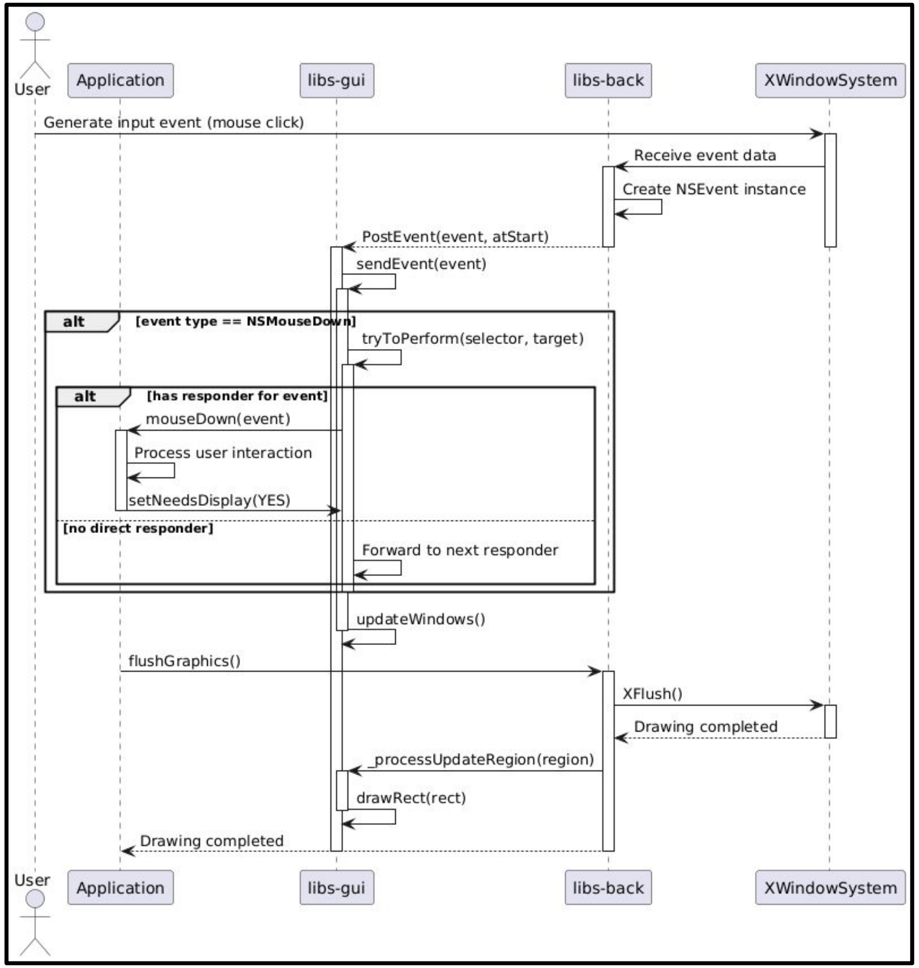


```
/*==================*
 * NSAnimation class *
 *==================*/
#define GSI_ARR Y_NO_RETAIN
#define GSI_ARRAY_NO_RELEASE
#define GSIArrayItem NSAnimationProgress
#include <GNUstepBase/GSIArray.h>
```

## Sequence Diagram

# GNUstep Graphics Rendering Update

- Mouse click triggers XWindowSystem
- Notifies libs-back
- Libs-back creates NSEvent and forwards to libs-gui with PostEvent
- Libs-gui sends event to correct responder
- After event handled, redraw is required
- Flush request passed from application to libs-back to XWindowSystem
- Libs-gui handles redraw



## Concurrency & Team Issues

- Concurrency
  - Event handling and drawing processes running concurrently can lead to potential race conditions.
  - Direct calls that bypass standard APIs for performance reasons introduce synchronization challenges
- Team issues
  - Limited documentation on legacy subsystems slowed down analysis
  - Task overlaps created unexpected dependencies among team members
  - Clearer initial task delegation and regular communication are essential to minimize duplicated efforts and rewards.

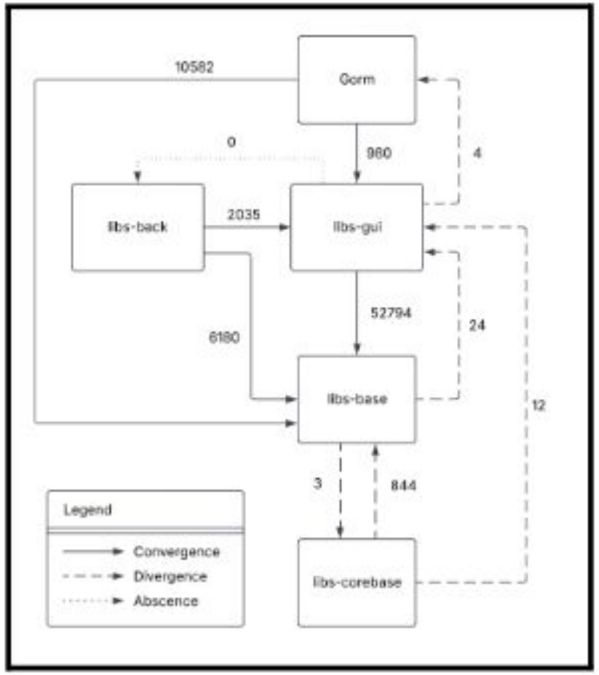
## 2nd level subsystem, libs-gui (Conceptual vs. Concrete)

- Conceptual
  - Clear and layered hierarchy, libs-gui would only depend on libs-base.
  - Minimal direct interactions, and clean separations between layers.
- Concrete
  - Additional dependencies (ex: libs-gui on libs-corebase).
  - Dynamic linking utilized for platform-specific graphical rendering.
  - Old optimizations causing tighter coupling and more complexity.

# Reflexion Analysis -2nd level subsystem (libs-gui Divergences)

Reflexion diagram illustrates conceptual vs. concrete dependencies.

- Convergences
  - Interaction between libs-gui and libs-back matches original design expectations.
- Divergences
  - Libs-gui depends on libs-corebase
  - Libs-gui makes unplanned calls into Gorm.



## 2nd level subsystem rationale for Divergences (libs-gui)

- Performance optimization
  - Direct methods have improved speed but cause unexpected system coupling.
- Cocoa compatibility
  - New interactions with libs-corebase were added to align better with the Apple API updates.
- Legacy/Historical code
  - Older implementations and legacy code resulted in dependencies that were not initially clear in conceptual designs.
- Incremental Feature growth
  - Gradual addition of features introduced dependencies which were not originally planned for, altering the subsystem relationships.
- Concurrency Adjustments
  - Modifications for better thread safety and event handling introduced unexpected dependencies across components.

## Alternatives

- Stricter Layer Enforcement
  - Define boundaries to restrict lower level calls from higher level components.
- Incremental Refactoring
  - Gradually removing the circular dependencies.
  - Replace old shortcuts with proper interfaces.
- Enhanced Documentation
  - Record old decisions more clearly for current and future developers.
  - Clarify the rationale behind the system interactions.

# Lessons Learned

- Conceptual vs. Reality
  - Real world factors (ex: performance, old code) cause deviations from initial plans.
- Importance of Documentation
  - Incomplete or outdated docs slow down understanding and analysis.
- Communication & Task Allocation
  - Clear task assignments prevent overlapping work.
- Balancing Abstraction & Efficiency
  - Overly optimizing performance can complicate the architecture and reduce its maintainability.