Technologies and concepts used for Viber iOS application.

Cyril @notorca Lashkevich
Łódź wiOSłuje

Viber

- 1.7M in Poland (of ~14M smartphones users)
- +100K/month in Poland
- 400M total
- 100M online
- 220 Vibers in 9 countries
- 4.5 years

Viber iOS project stats

- 206k SLOC of iOS project
- + 50k SLOC of 3rd party iOS libraries
- + 140k SLOC of out cross-platform library
- + 300k SLOC of WebRTC media engine
- + 120k SLOC of audio/video codecs
- Almost 800k SLOC

- All this code is compiled in one fat binary, 28 Mb for the each architecture.
- According to the Unix philosophy Viber codebase is sucks
- ~~KISS, less is more, worse is better.~~
- ~~Removed code is debugged code.~~

Project structure

- 1 workspace
- 3 project files
- 29 targets to build Viber iOS (39 total)
- 12 git repositories (GitHub private)
- 3 min 20 sec to build after clean

Why splitting into the small libraries is important?

Except Architecture and Design reasons

- Separate build settings for targets
- Optimization levels: -O0, -Os, -Ofast
- Preprocessing directives
- Linking options: -flto
- Simplify team-work

ObjC

Viber iOS

Java

Viber Android

ObjC++

C++ API

JNI

Codec

WebRTC Media Engine

LibViber

Codec

ViberCore

Platform

Own libraries

- git repo for iOS project + recursive submodules
- Separate Xcode project for LibViber
- Separate Xcode project for WebRTC

3rd party libraries

- Every 3rd party library should be approved.
- Only in source code
- Are we ready to fix any bugs and adopt this code for the new OS version, compilers, etc…?
- Use 3rd party library **VS** implement in house.

Just use or evolve

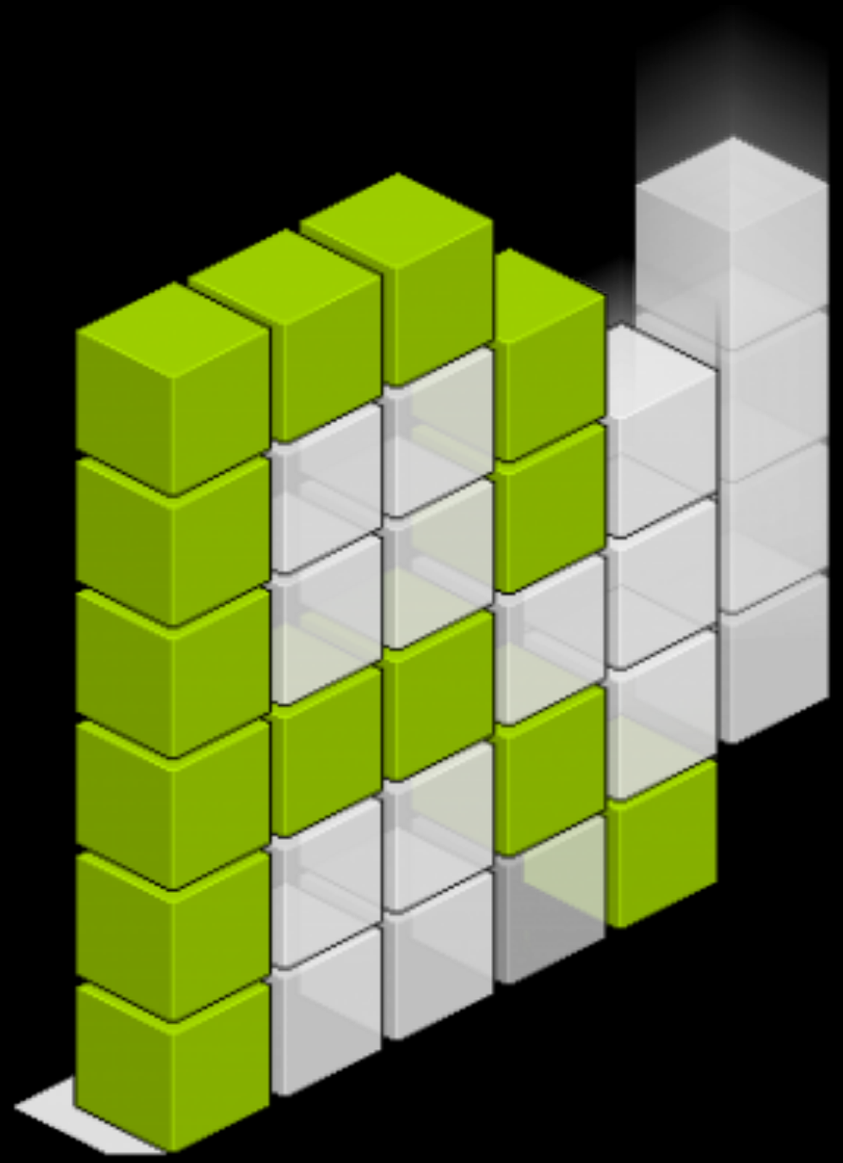
- Some of libraries are can be used without modifications
- Some of libraries we want to change, adopt or involve
- Based on this we are organize the source code in differrent way.

Submodules vs Subtrees

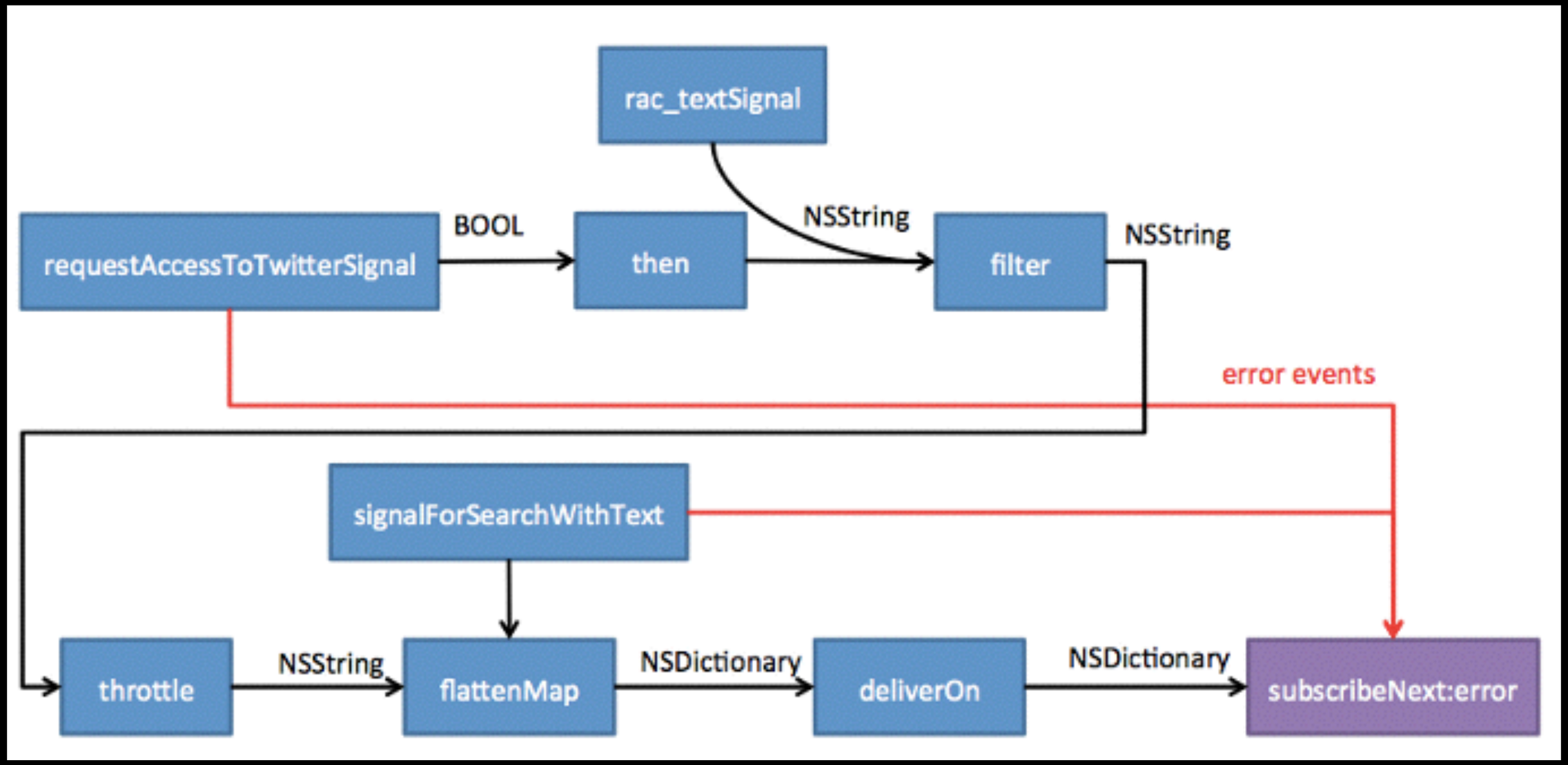
- More than one platform is using this code
- We are going tho change the code a lot
- Code can be used separately form the main project
- No changes at all or small changes only
- We don’t need separate branches for the library code
- Someone is responsible for merging form the upstream

ReactiveCocoa

- KVO replacement
- Functional programming elements
- Less state, less bugs
- Compilcated
- A lot of blocks in the code



```
RAC(self, label.text) =
                        RACObserve(self, name);
```



http://www.raywenderlich.com/62796/reactivecocoa-tutorial-pt1

```
[[[[[[[self requestAccessToTwitterSignal]
  then:^RACSignal *{
    @strongify(self)
    return self.searchText.rac_textSignal;
  }]
  filter:^BOOL(NSString *text) {
    @strongify(self)
    return [self isValidSearchText:text];
  }]
  throttle:0.5]
  flattenMap:^RACStream *(NSString *text) {
    @strongify(self)
    return [self signalForSearchWithText:text];
  }]
  deliverOn:[RACScheduler mainThreadScheduler]]
  subscribeNext:^(NSDictionary *jsonSearchResult) {
    NSArray *tweets = [jsonSearchResult[@"statuses"]
      .rac_sequence map:^(id tweet) {
        return [RWTweet tweetWithStatus:tweet];
    }].array;
    [self.resultsViewController displayTweets:tweets];
  } error:^(NSError *error) {
    NSLog(@"An error occurred: %@", error);
  }];
```

Mantle

- Simple model layer
- Replacement for the NSDictionary
- Replacement for „property only” classes
- Removes a lot of boilerplate:
  `initWithDictionary:`, `description`, `debugDescription`, `initWithCoder:`, `encodeWithCoder:`, `copyWithZone:`, `isEqual:`, `hash:`

```
NSDictinary *httpRequestSetup = @{
    @"URL"          : [NSURL URLWithString:@"http...."],
    @"HTTPMethod"   : @"GET",
    @"HTTPHeaders" : @{},
    @"HTTPBody"     : [NSData dataWithString:@""],
    @"resumable"    : @(YES),
    @"streamBoundary" : @"---123---"
    @"streamBody"     : [NSData dataWithString:@""]
};
```

- No type checking
- Only tons of tests can save in case of big project

```objc
@interface VTMHTTPRequestSetup

@property (nonatomic, readonly) NSURL *URL;
@property (nonatomic, readonly) NSString *HTTPMethod;
@property (nonatomic, readonly) NSDictionary *HTTPHeaders;
@property (nonatomic, readonly) NSData *HTTPBody;
@property (nonatomic, readonly) BOOL resumable;
@property (nonatomic, readonly) NSString *streamBoundary;
@property (nonatomic, readonly) NSData *streamBody;

// initWithDictionary:, copyWithZone:,
// description, debugDescription,
// initWithCoder:, encodeWithCoder:,
// isEqual:, hash:

@end
```

```objc
@interface VTMHTTPRequestSetup : MTLModel

@property (nonatomic, readonly) NSURL *URL;
@property (nonatomic, readonly) NSString *HTTPMethod;
@property (nonatomic, readonly) NSDictionary *HTTPHeaders;
@property (nonatomic, readonly) NSData *HTTPBody;
@property (nonatomic, readonly) BOOL resumable;
@property (nonatomic, readonly) NSString *streamBoundary;
@property (nonatomic, readonly) NSData *streamBody;

@end
```

- Converting to/from JSON
- Converting to/from `NSManagedObject` (but be careful with compacted object graphs)

Other Libraries

- FastImageCache
- Lumberjack
- libPhoneNumber
- FMDB
- FacebookSDK
- boost

Cool Stuff

- weakself
- Main thread trace
- New features under macro
- Swift
- iOS8

New features development

- In master
- In branches
- In master but inside the #ifdef
  - No problems with merge
  - Problems with builds
  - Tons of ENABLE_FEATURE.. in preprocessing settings

Problems with reference to `self`

- self is captured by strong reference in block
- Access to ivar is done throw implicit self

```
^ { NSLog(@"%@", _ivar); };
^ { NSLog(@"%@", self->_ivar); };
```

- Retain cycle when block is saved as class member

@weakself

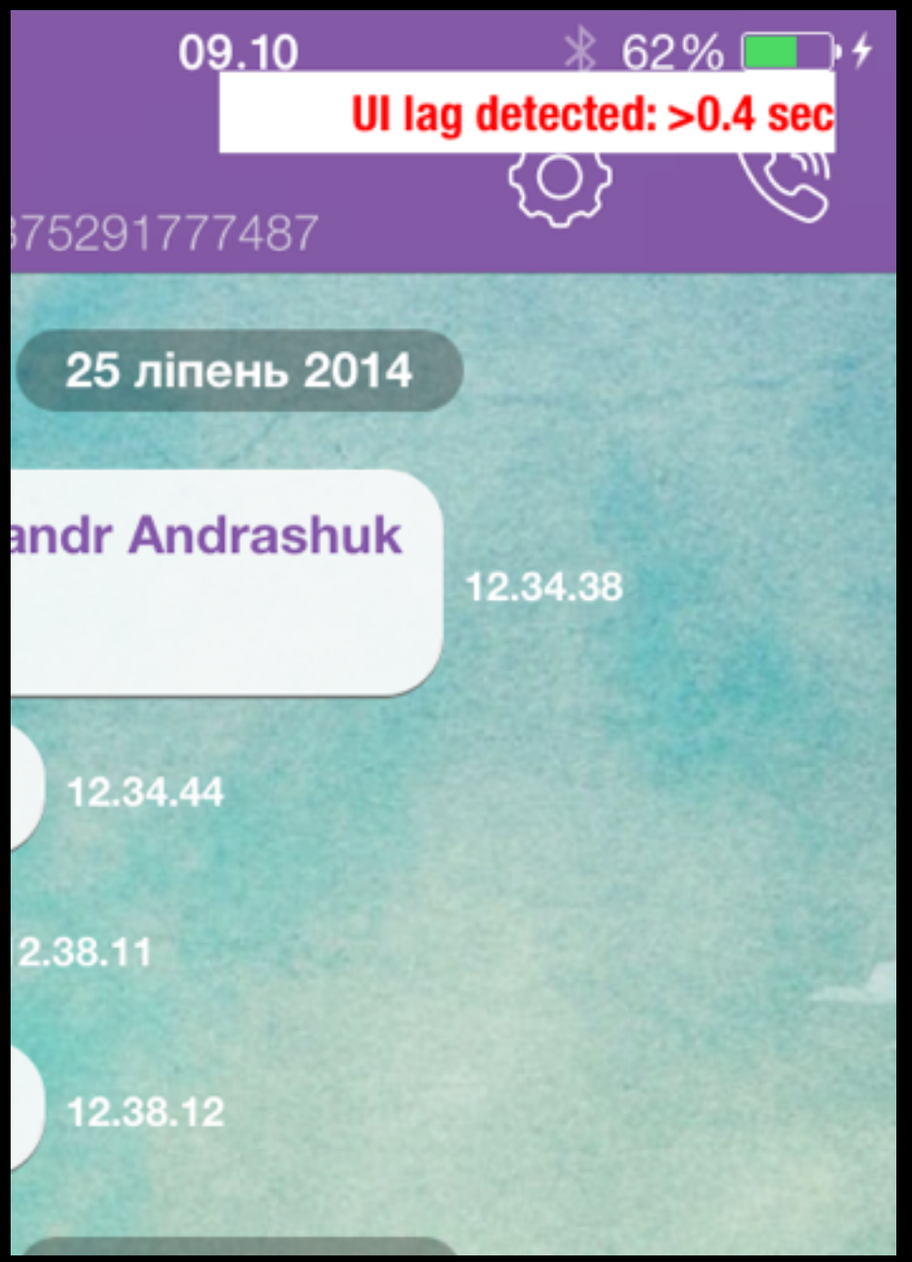
- Safe using of `self` in the block: weak until the block is called, strong during call
- `self` should be called `self`
- Checking for ivar access in the block

```
[RACObserve(self, pttState)
subscribeNext:@weakselfnotnil(^(NSNumber *state)) {
    self.isRecordingPTT = !!state.intValue;
} @weakselfend];
```

https://gist.github.com/notorca/9192459

Main thread profiler

- UI events are handled in main thread so it should be responsible
- Every 0.1 sec run a block on main thread queue.
- If block execution is delayed, dump backtrace in log and show notification in UI



https://github.com/kstenerud/KSCrash

```
while (![[NSThread currentThread] isCancelled]) {
    static bool pingTaskIsRunning;
    pingTaskIsRunning = YES;
    dispatch_async(dispatch_get_main_queue(), ^{
        pingTaskIsRunning = NO;
        dispatch_semaphore_signal(semaphore);
    });
    [NSThread sleepForTimeInterval:0.4];
    if (pingTaskIsRunning) {
        // Notify about freeze
    }
    while (pingTaskIsRunning) {
        dispatch_semaphore_wait(semaphore,
                                DISPATCH_TIME_FOREVER);
    }
    [NSThread sleepForTimeInterval:0.1];
}
```

Swift

- **Yes, we are using Swift!**
- 15-lines script for burning version number to the application icon
- For the application code only after Xcode 6.1

iOS8

Adopt as much as possible
and reasonable

@notorca