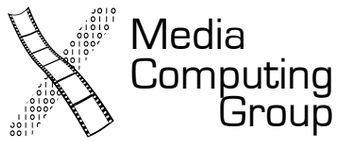


RWTH AACHEN UNIVERSITY

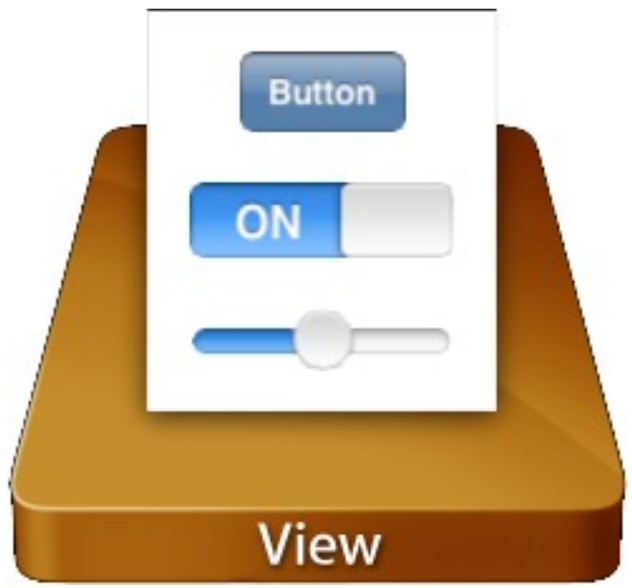
Mobile Application Development
L05: iOS Design Patterns

Jonathan Diehl (Informatik 10)
Hendrik Thüs (Informatik 9)

iOS Design Patterns

- Model-View-Controller
- View Controllers
- Actions and Outlets
- Delegation

Model-View-Controller



Model

- Custom Classes
- Responsibilities:
  - Domain / Business Logic (Independent of the View!)
  - Data Storage (Properties, KVC, NSCoding, CoreData)
- Examples
  - User Data (Document, Person, BankAccount, ...)
  - Application Data (Preferences, State, ...)
  - Functionality (Libraries, Algorithms, ...)

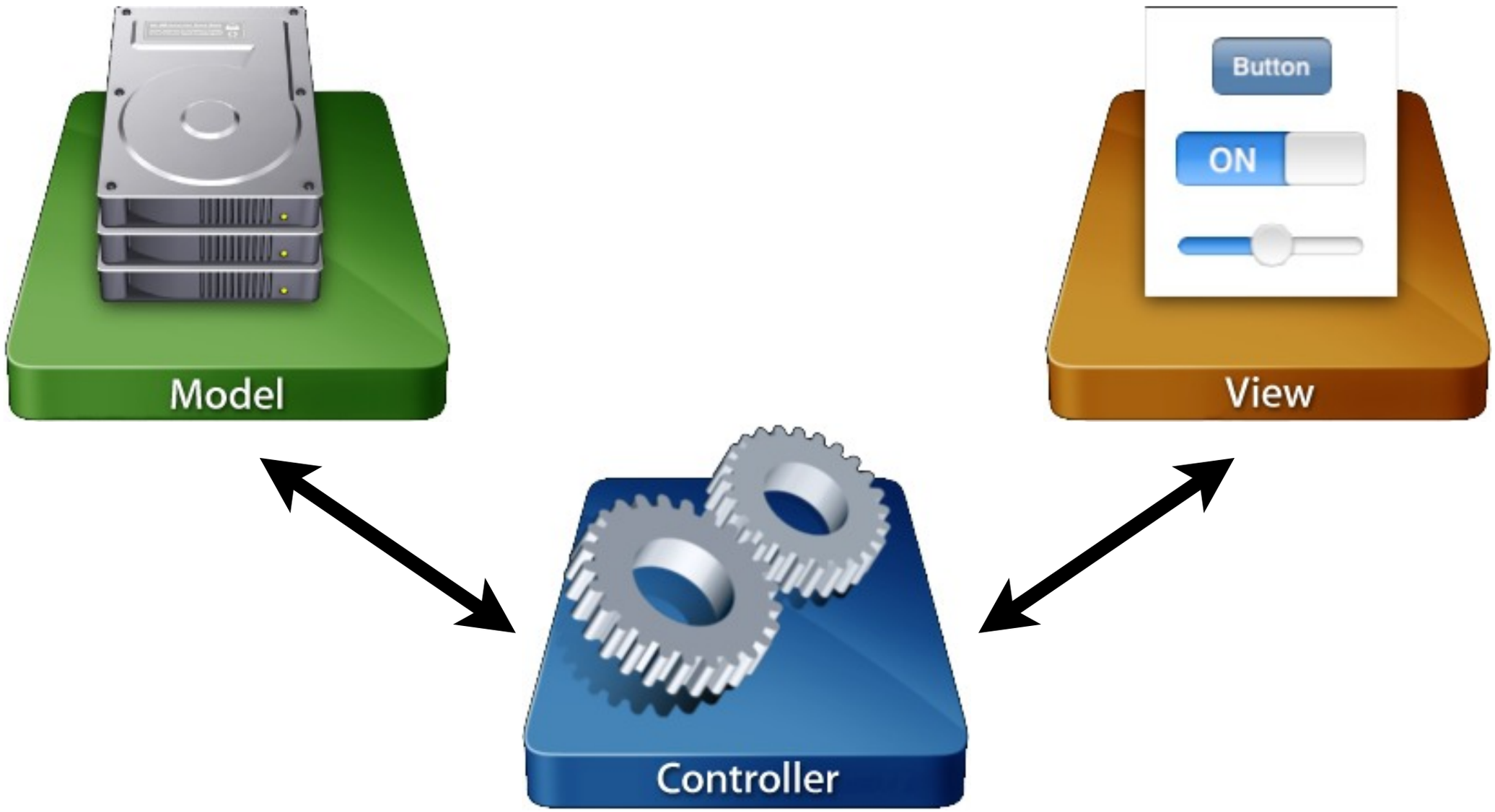
View

- User Interface Elements
- Subclass of UIView
- Responsibilities:
  - Look and Feel of the UI
  - Data Visualization
  - Generate UI Events
- Examples:
  - UIButton, UILabel, UITableView, MyCustomView, ...

Controller

- Glue-code between Model and View
- Custom Classes, Subclass of UIViewController
- Responsibilities:
  - Configure View according to Model
  - React to UI Events and update Model
  - Application Logic (Navigation,
- Examples:
  - UIViewController, MyViewController, ...

Model-View-Controller



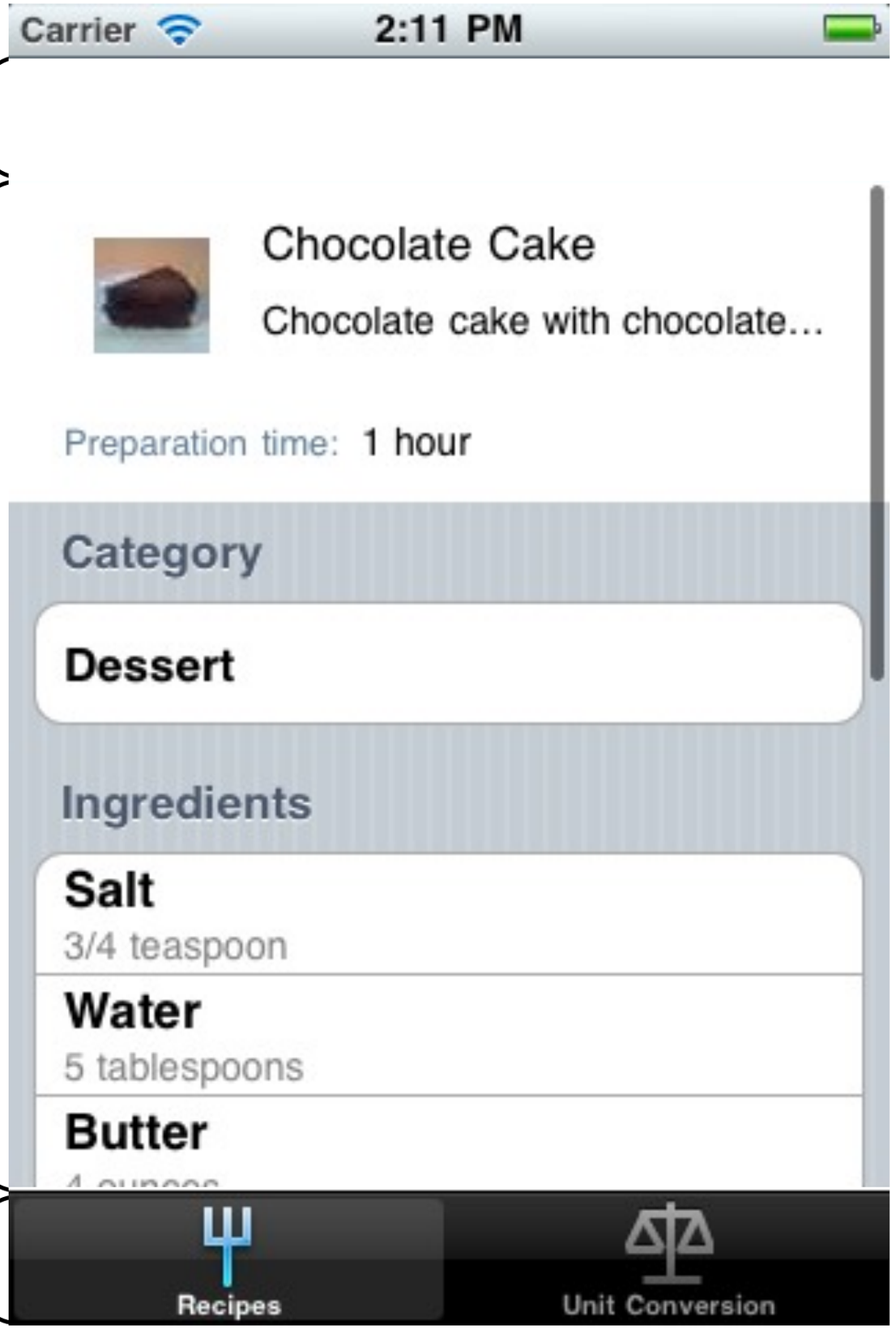
View Controllers

- Base class for a controller that manages a view
- UIViewController simplifies standard behavior:
  - load resources from a Nib file
  - configure navigation bar, tab bar, and tool bars
  - pluggable architecture
  - handle events and memory warnings
  - manage interface orientation changes

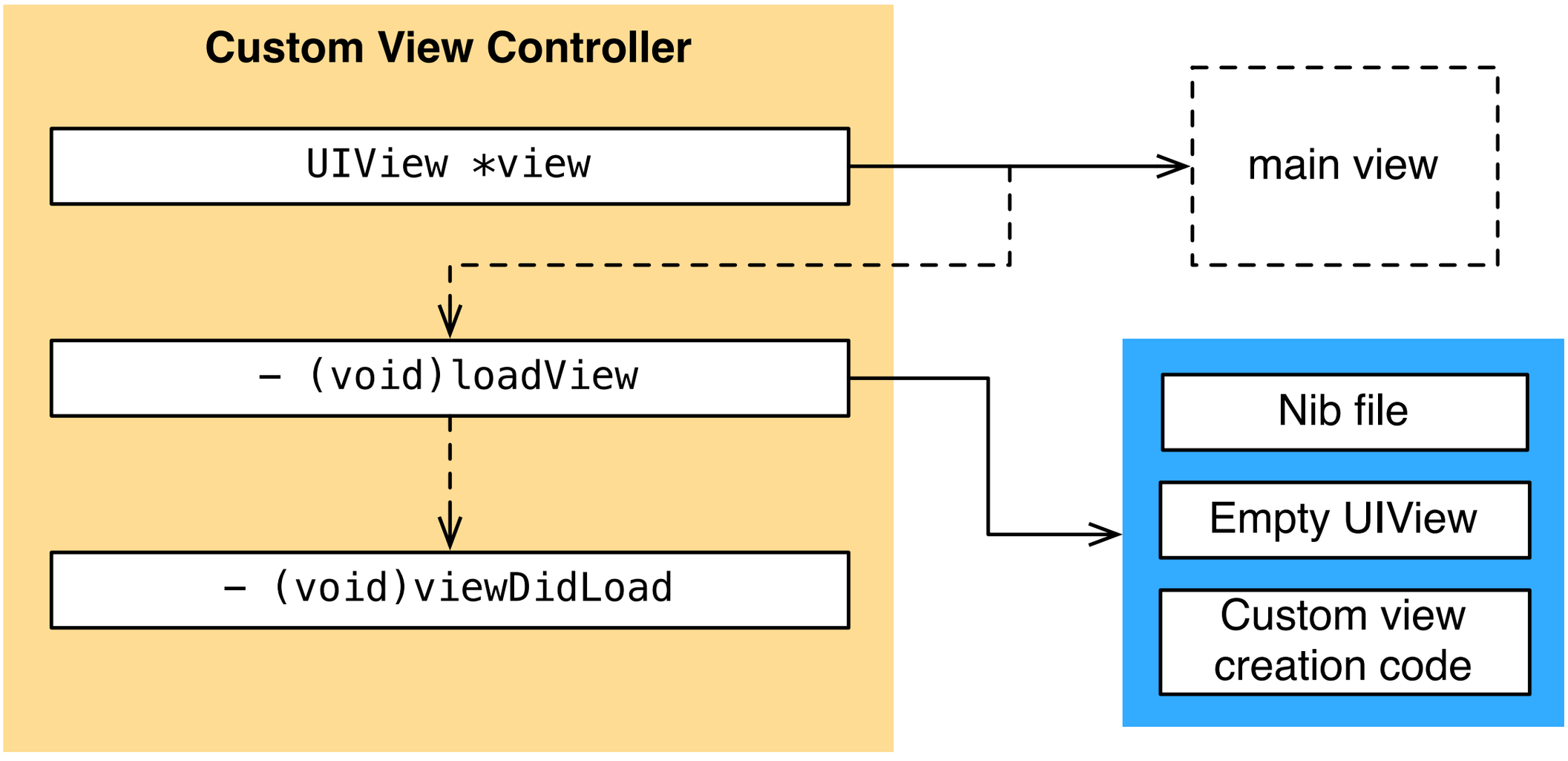
UINavigationController

DetailViewController

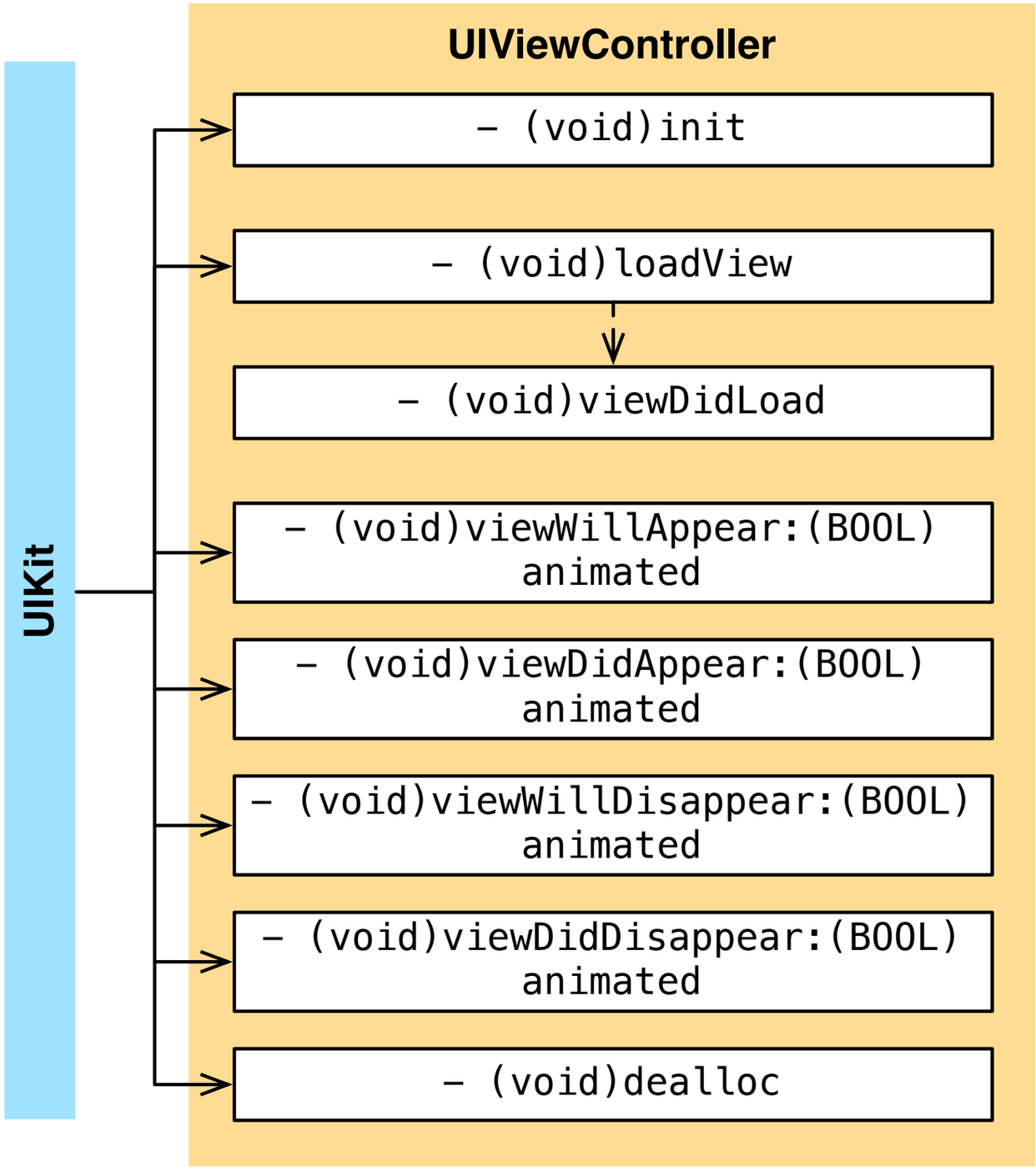
UITabBarController



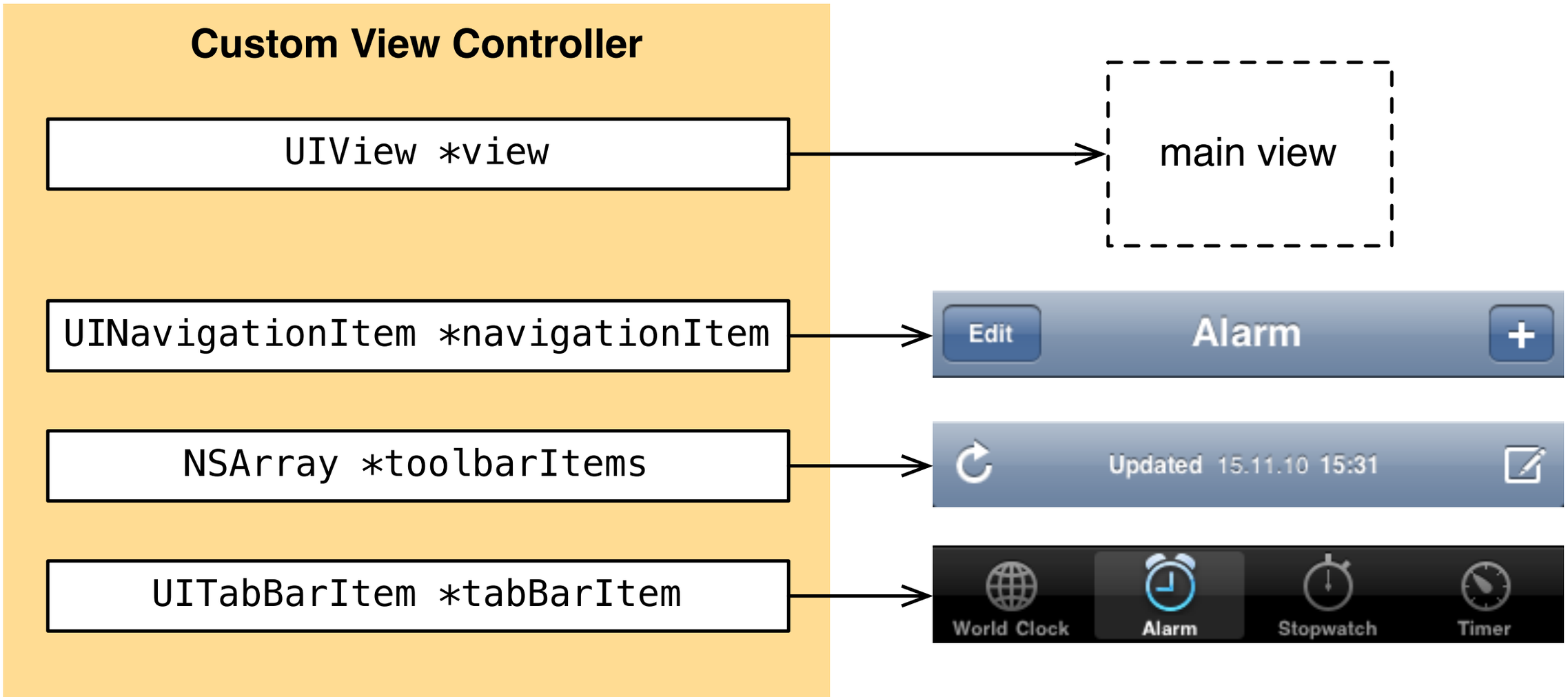
View Management Cycle



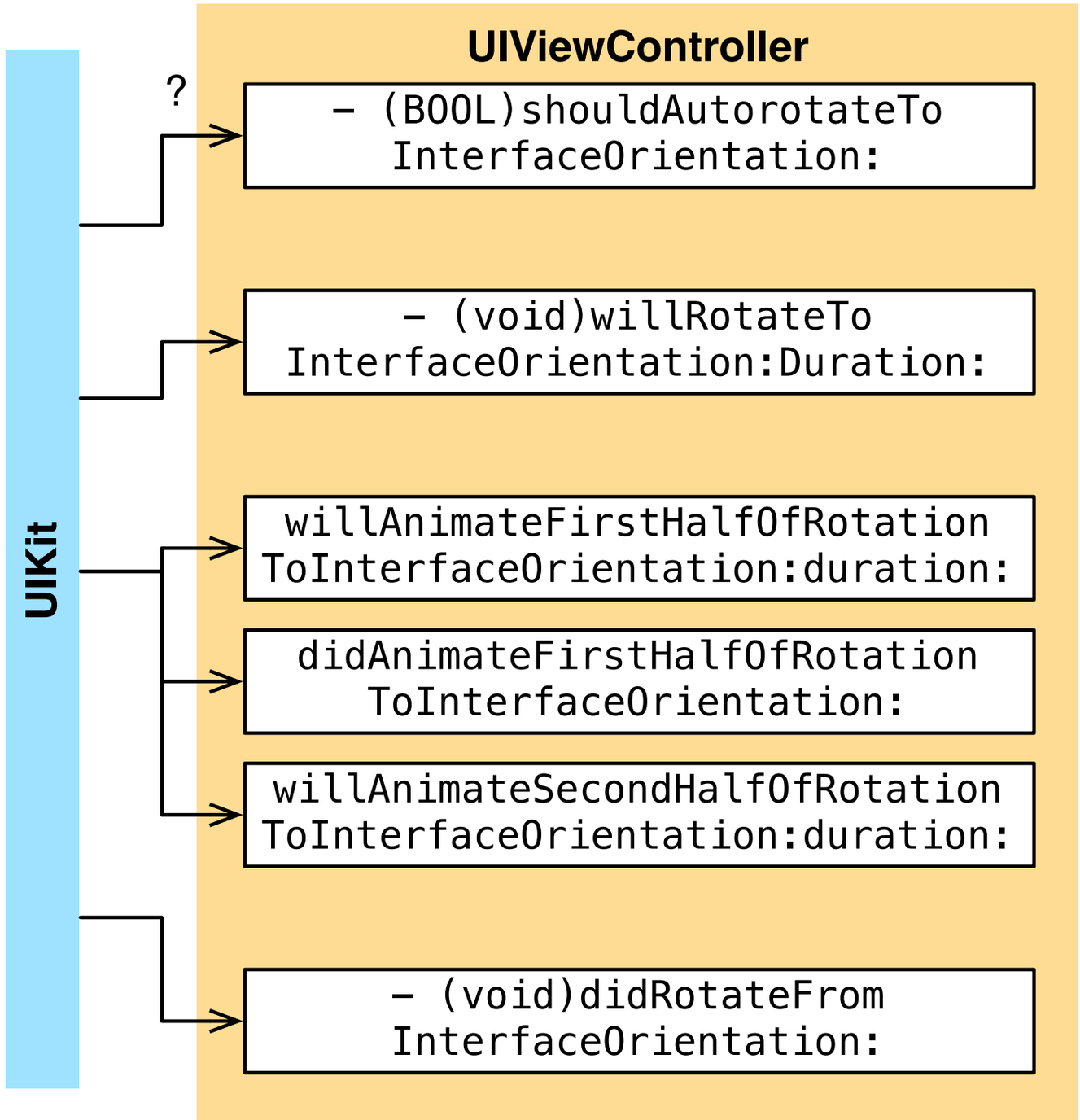
View Controller Life Cycle



Standard Interfaces

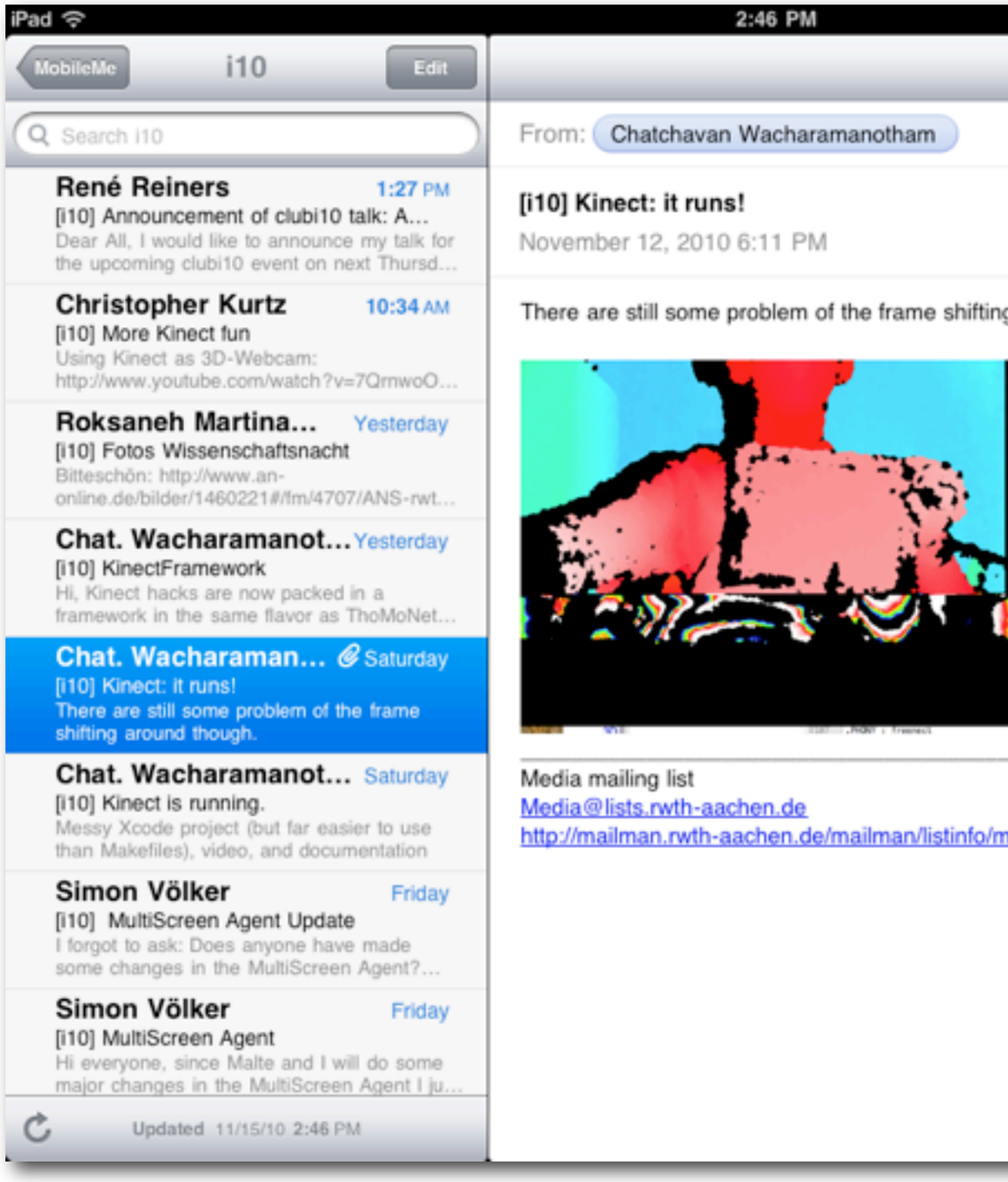


Interface Orientation



Built-In View Controllers

- UINavigationController
- UITabBarController
- UISplitViewController
- UITableViewController
- Framework-specific



Actions and Outlets

- User Interfaces in iOS are defined in Nib files
  - A nib file stores an archived collection of objects
    - Visual UI editor integrated into Xcode
- Actions: methods that can be triggered from UI events
- Outlets: instance variables in the controller that allow accessing view elements

“Drawing” Connections

- Actions
  - Declare a method as IBAction

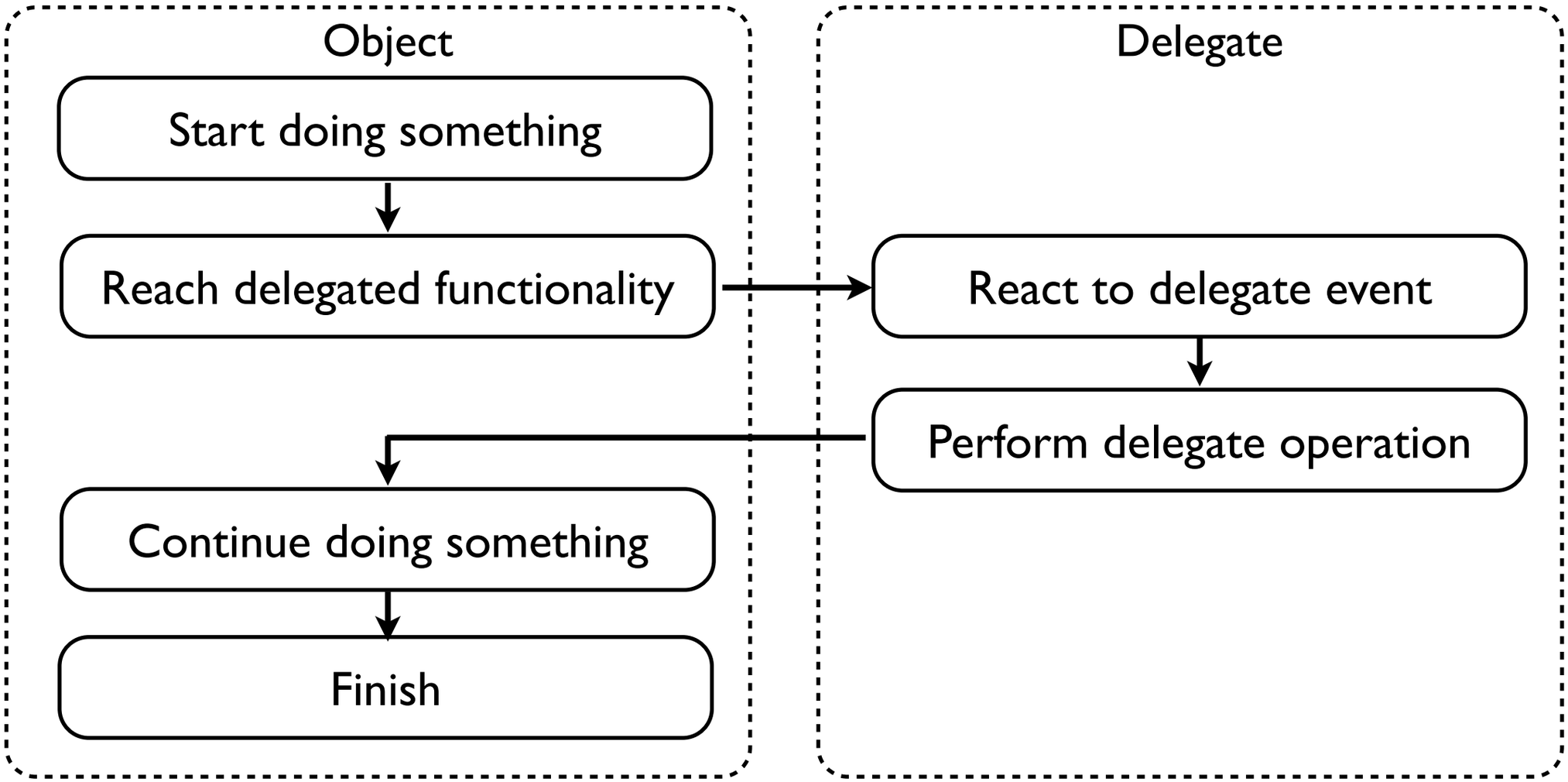
    - (IBAction)doSomething:(id)sender
  - Right-click drag from the UI widget to the controller
- Outlets
  - Declare an instance variable or property as IBOutlet

    @property(assign) IBOutlet UIView *view
  - Right-click drag from the controller to the UI widget

Delegation

- Alternative to Subclassing
- Delegate Behavior to Another Class

Delegate


Object
Start doing something
Reach delegated functionality
Continue doing something
Finish
Delegate
React to delegate event
Perform delegate operation

Implementing Delegates

1. Conform to the Delegate Protocol
   - Implement custom behavior in delegate methods
2. Assign Delegate Object
   - Usually via property
   - The delegate object is not retained (retain-cycle)

Delegates in the iOS SDK

Object	**Delegate**
UIApplication	UIApplicationDelegate
UITableView	UITableViewDelegate
UITextField	UITextFieldDelegate
UIPickerController	UIPickerControllerDelegate
CMMotionManager	CMMotionManagerDelegate

and many more...