The Model View Presenter Pattern and Android

Martin Keiblinger

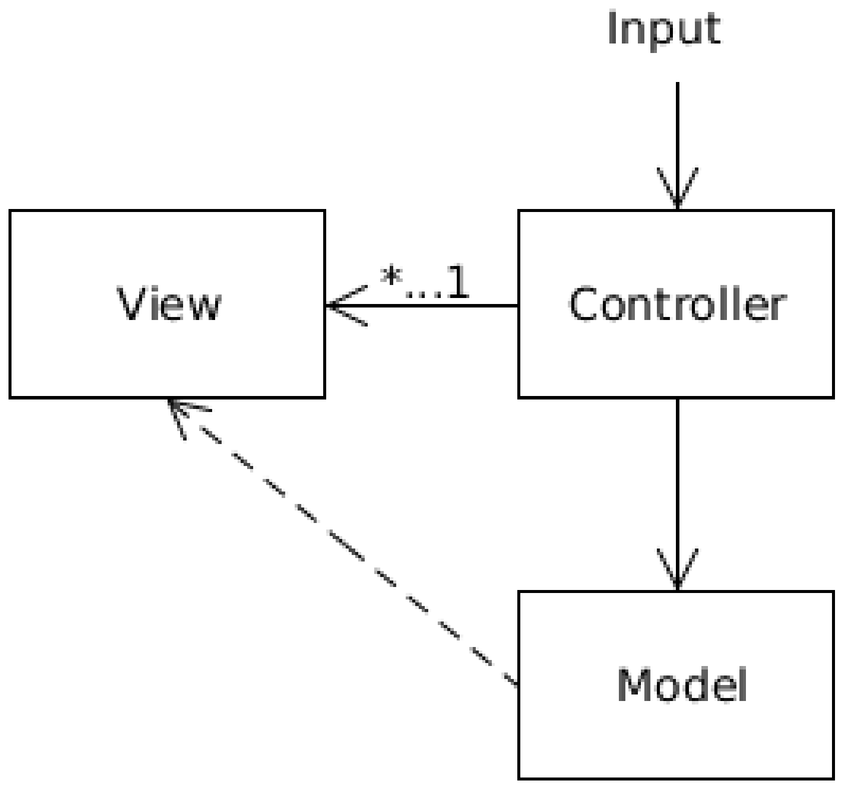
Why architecture patterns?

- Seperation of Concers
- Small modules ↔ better testability
- Easier refactoring
- Easier bug fixing
- Kick ass apps

What about MVC?

- Everybody knows MVC (even Microsoft)
- Good documentation
- Many examples on how to do it
- Platform agnostic
  - Used in Java/Swing
  - Used in countless web frameworks

What is MVC? (quick reminder)



MVC is not an Android pattern

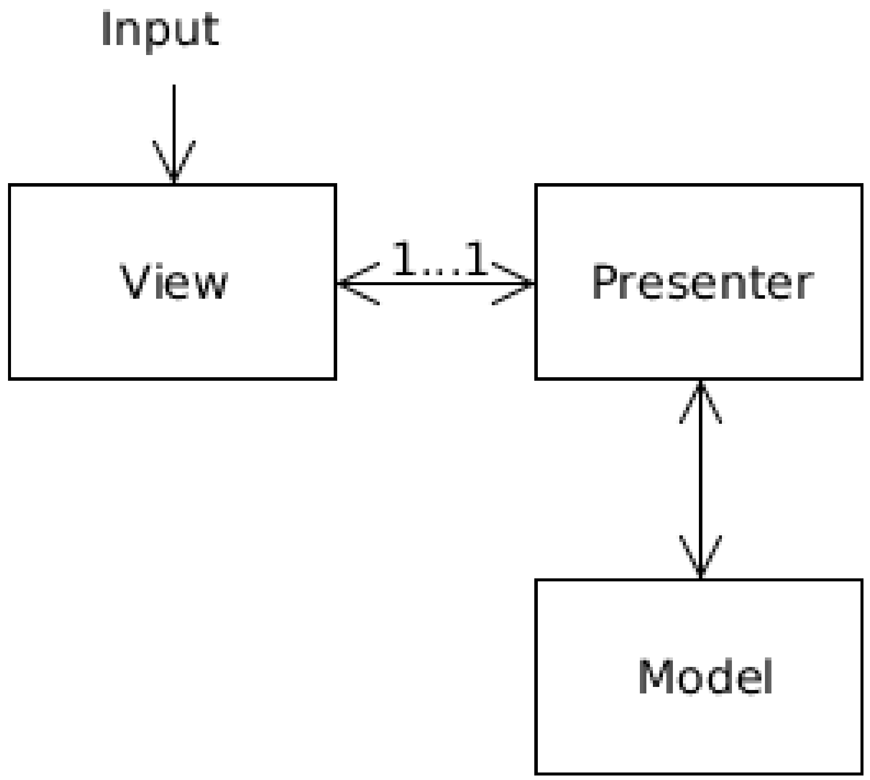
- Android is centered on Activities
  - Activities ↔ View
  - Activities provides Lifecycle-Callbacks
- In Android the Controller has to
  - React on lifecycle events
  - Has to be the component bootstrapping the app
  - Is not the view

  → **Doesn't sound like Android**

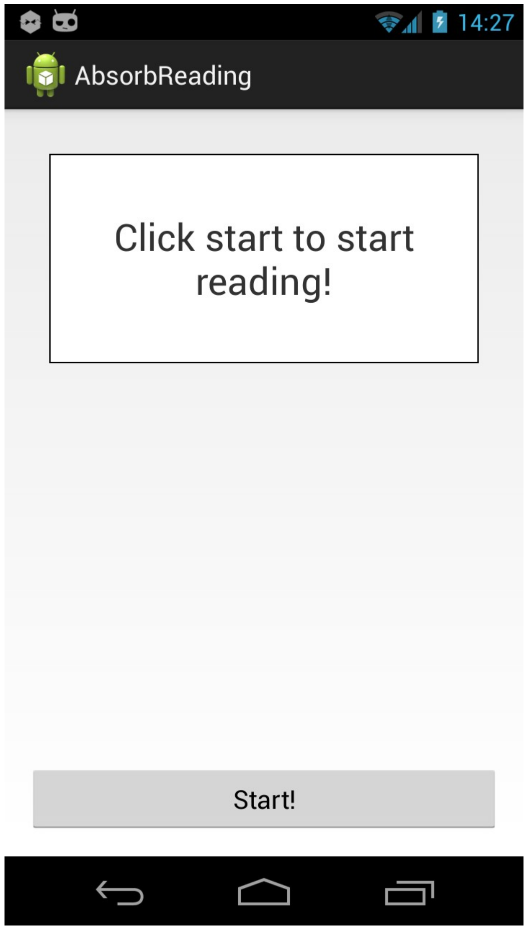
MVP could work ...

- MVC derivat
- Originally made for WPF
- View gets input
- View holds reference to Presenter
  - Lifecycle-Events
  - Broadcast-Receiver

MVP could work ...



MVP works!



Click start to start reading!

Let's have a look at the code ...

Is MVP the one and only truth for Android?

Of course **not**.

I'd rather see developers build kick-ass apps that are well-designed and follow separation of concerns, than see them waste time arguing about MV nonsense*

Igor Minar

TL;DR: Just **write good software**.

Sources

- http://schlingel.bplaced.net/android_mit_mvp_strukturieren.html
- https://github.com/schlingel/AbsorbReader
- http://geekswithblogs.net/dlussier/archive/2009/11/21/136454.aspx
- http://blogs.msdn.com/b/jowardel/archive/2008/09/09/using-the-model-view-presenter-mvp-design-pattern-to-enable-presentational-interoperability-and-increased-testability.aspx
- http://martinfowler.com/eaaDev/PresentationModel.html
- http://schlingel.bplaced.net/mvc_mvp_mvvm_fue