

Agile Software Development (DIT191 / EDA397)

Eric Knauss

<eric.knauss@cse.gu.se>

Agenda today

- Groups
- Version control systems
- Android



About groups



<https://flic.kr/p/ffDMHK>

About Groups

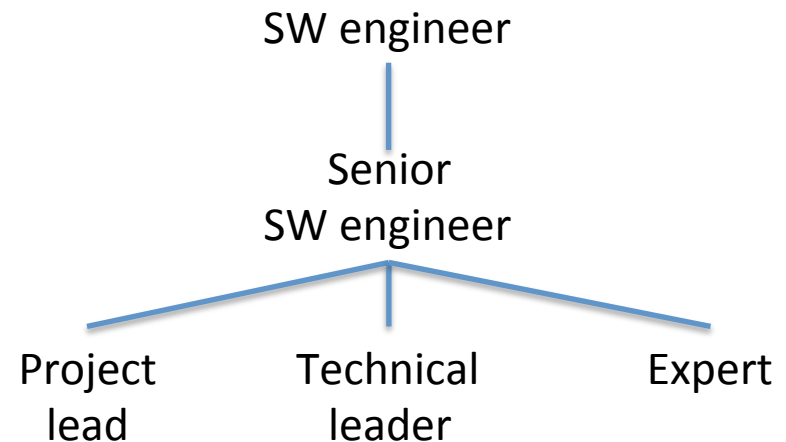
- First: Thank you for your feedback!
- Should I assign groups or you?
 - Pro: self-grouping
 - Faster
 - Established groups might work better (?)
 - Contra: self-grouping
 - Unbalanced groups
 - Hard to introduce new guys in established groups
 - Unfair disadvantage for some (?)
- Meta-aspect: What supports learning most?
 - Realism argument: “You cannot choose your group in industry”
 - Social dynamic part of the beauty of agile methods

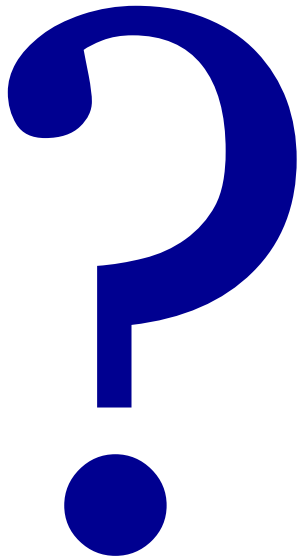
Groups: Towards a Compromise

- Work with new people!
 - Expand your network
 - Increase the diversity in your team
 - A chance to learn
- Realism revisited
 - Cannot choose your group?
 - Different profiles, different careers
 - Example

→ People are different

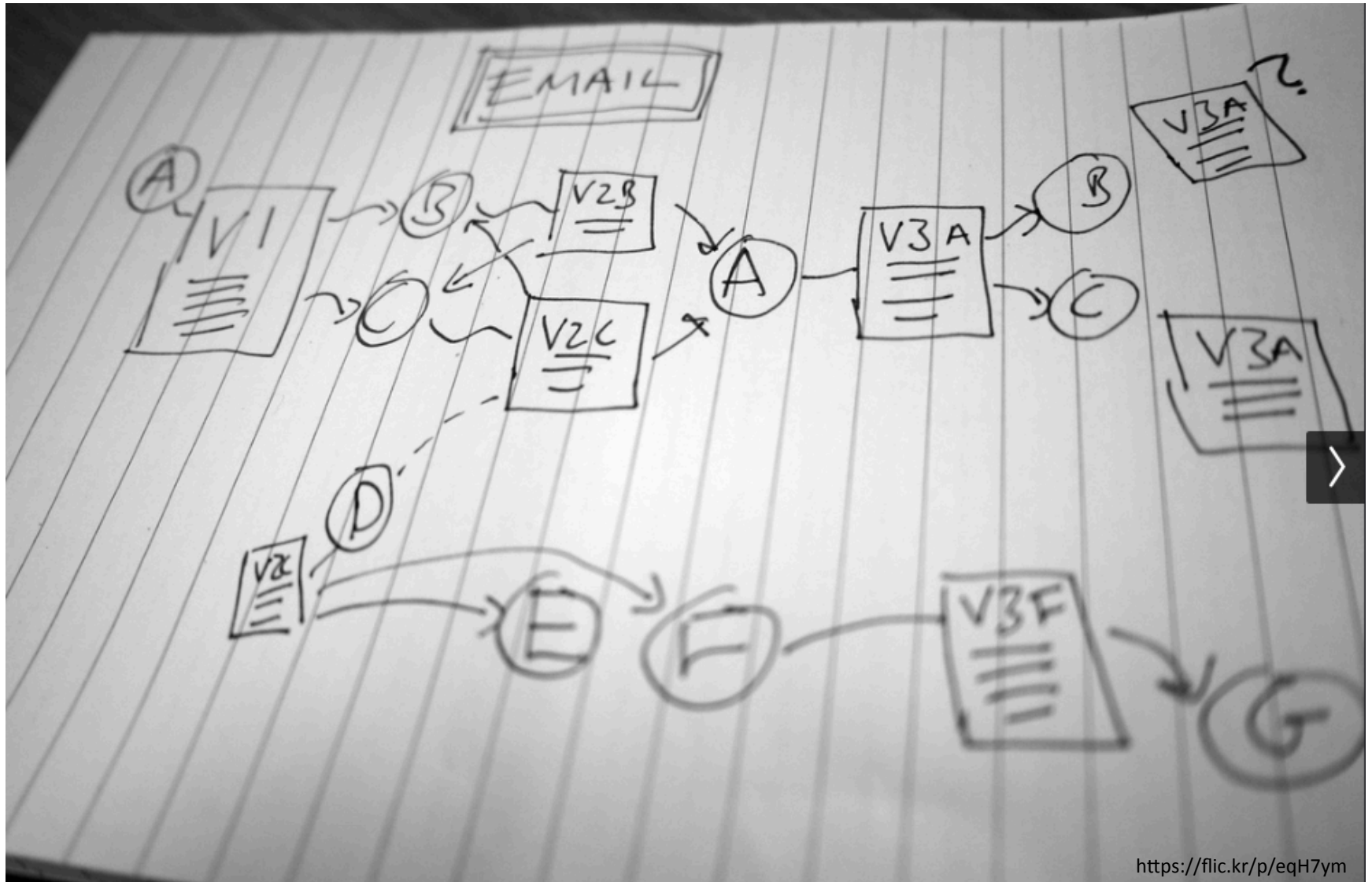
“You are all individuals!”





- What do you want to do in 5 years?
- ...in 10 years?
- Where are the future Project leaders?
- ...the future chief architects?
- ...the future experts (in QA, HCI, UX, KM, ...)

Version Control



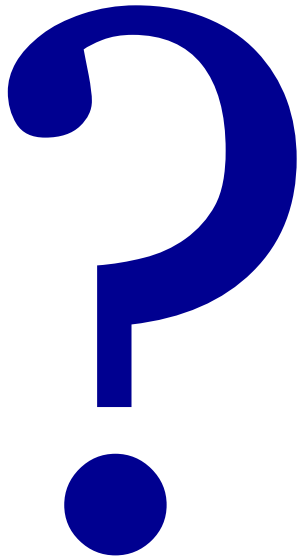
<https://flic.kr/p/eqH7ym>

Version Control

- Part of Software Configuration Management
 - Used to manage changes to the source code
 - tracks revisions, with timestamps and users
 - allows control (rollback, branching, etc)
 - many allow concurrent work (sane)
 - “Essential” for multi-developer projects

Important Concepts

- Repository / Working Copy
- Change (List)
- Commit
- Trunk / HEAD
- Conflict / Merge
- History

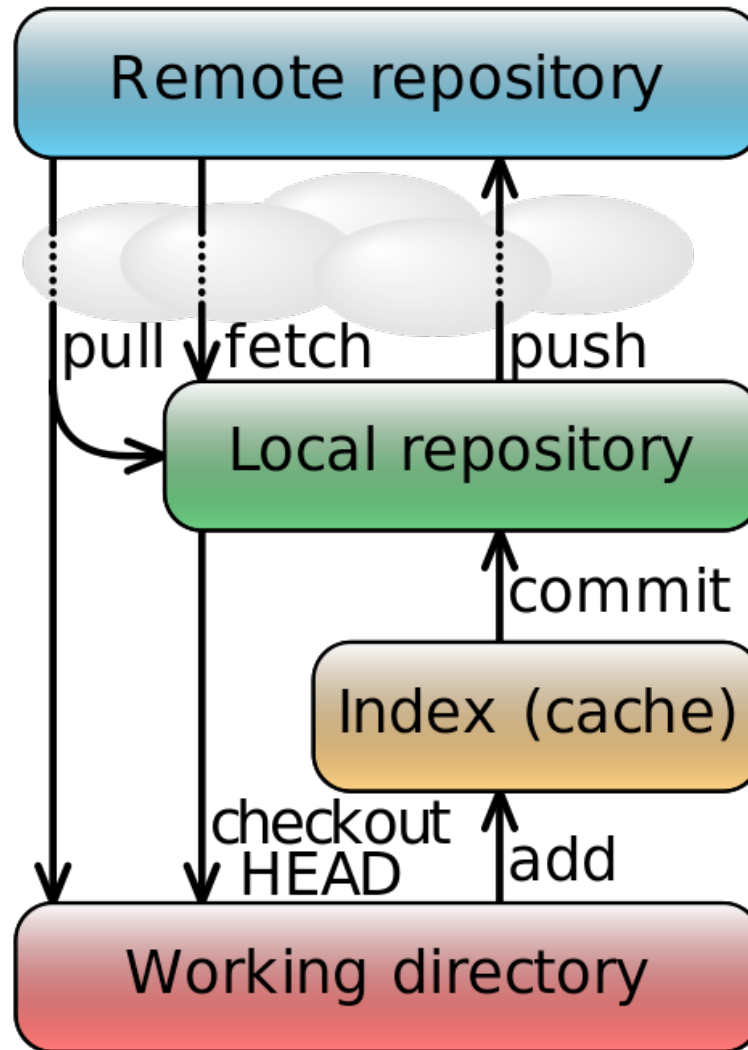


- Why is version control so important?
- Why is configuration management so important?
- What is the difference between the two?

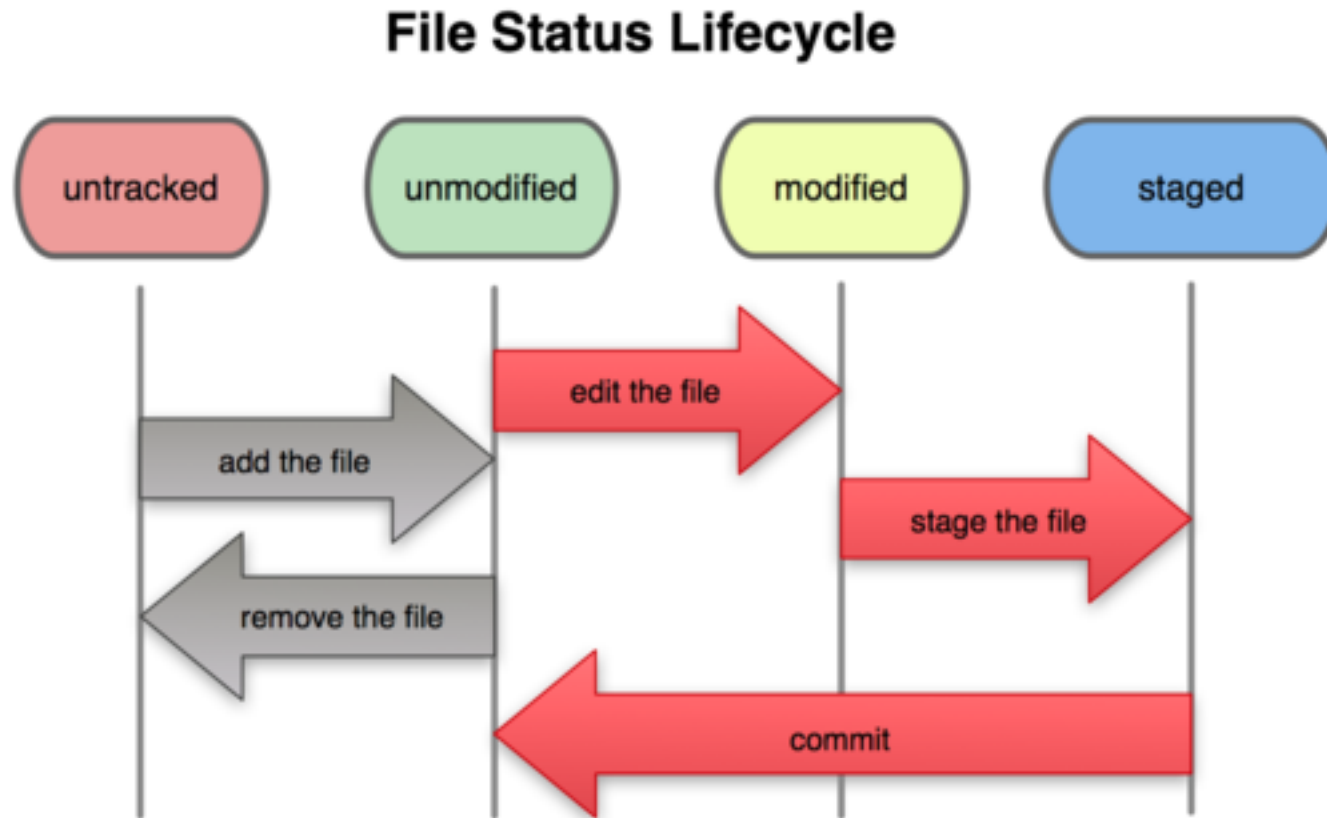
Git

- Distributed version control
- Developed by Linus for Linux Development
- Distributed
- Strong support for non-linear development
- Efficient for large projects

Workflow / Concepts



Workflow / Concepts



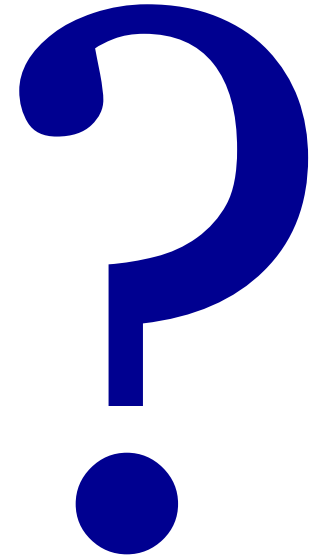
Android



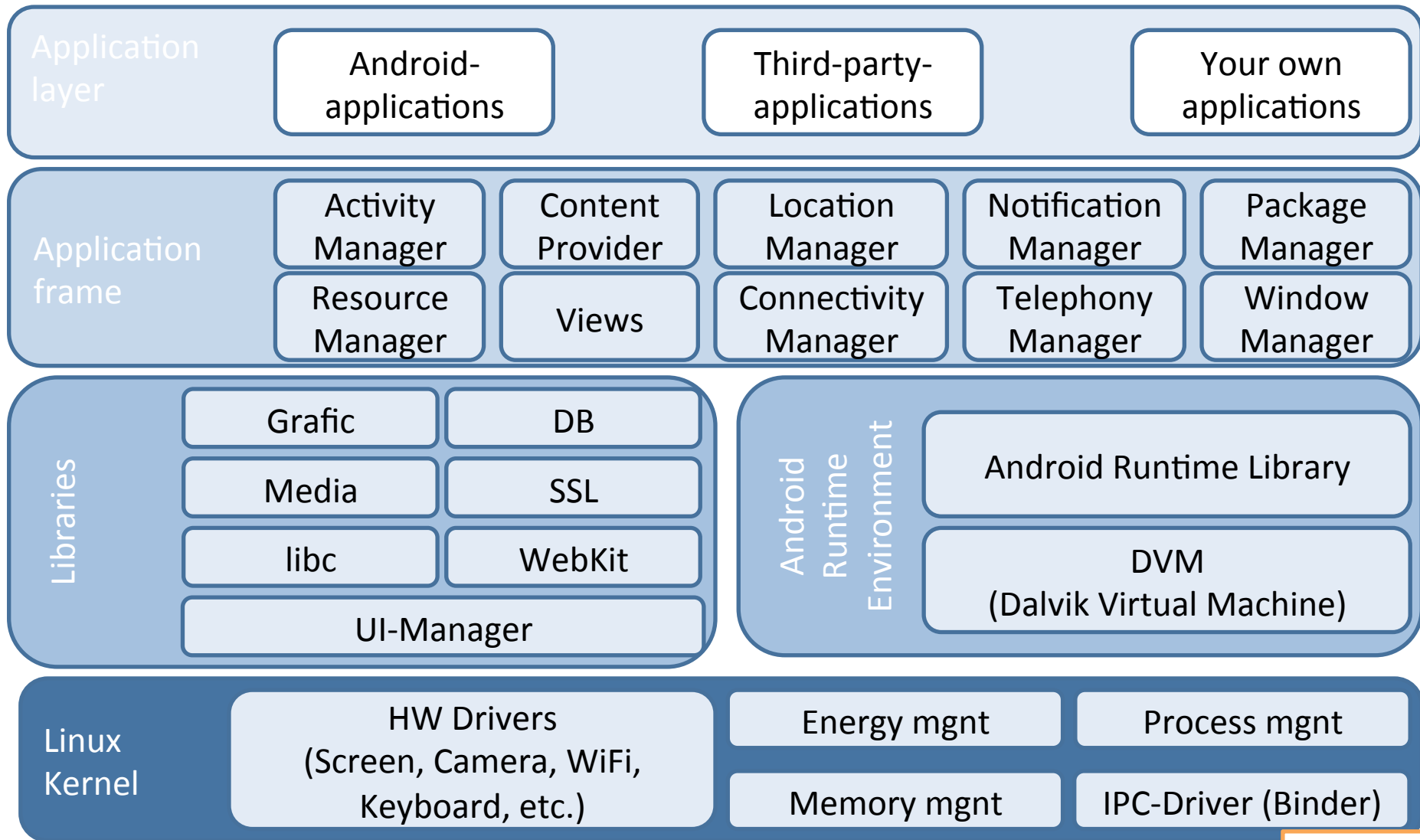
<https://flic.kr/p/jPnPwv>

Android SDK

- From our perspective
 - Java-based API
 - XML-based layout
 - Tools to build/install/simulate
- Two options for today:
 - Let's get started!!!
 - Get some overview first



Android-Architecture



[BP10]

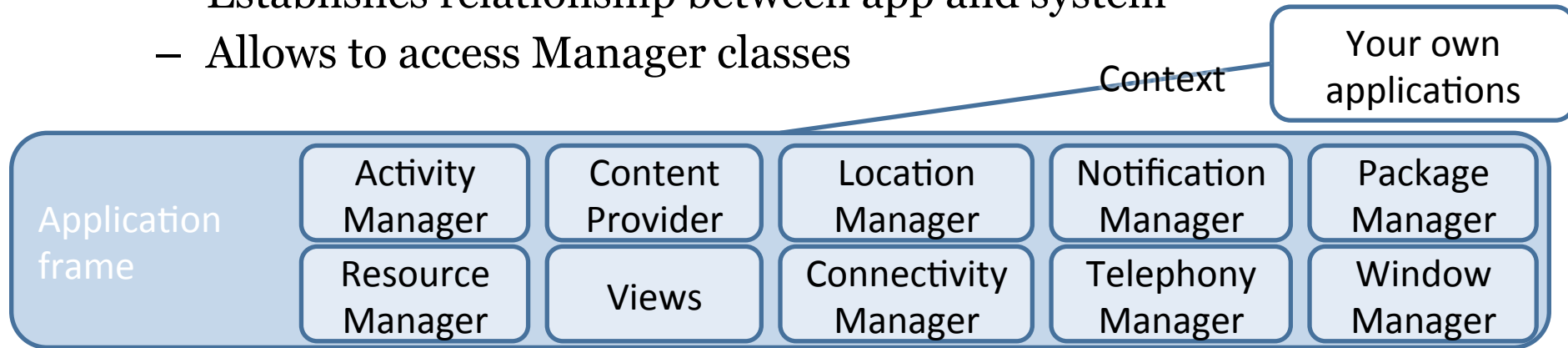
Architecture of Android App

Distinguish 4 Types of components

- Activity
 - View / Presentation layer
 - Show and manage UI
- Service
 - Controller / Application layer
 - Allows operations to run in background
 - *Example:*
 - *Musicplayer is controlled through Activities*
 - *Music is played as Service in background*
 - *Even after the UI is closed*
- Content Provider
 - Model / Persistence layer
 - Manage data and persistence layer
 - Can provide data to a specific app or even to many apps
- Broadcast Receiver
 - Receives system messages
 - Allows App to react on changes in state of system
 - *Example*
 - *Problems with network connection*
 - *Empty battery*

The Context Class

- The class Context represents the application context
 - Establishes relationship between app and system
 - Allows to access Manager classes



- How to use this:
 - Activity and Service classes are derived from abstract class Context
→ access via this.*
 - Broadcast Receiver get Context during initialisation as a parameter
 - Content Provider can access the Context via getContext method

Connect Components: Intents

- Goal: Connect components of different apps
 - With a standardized mechanism
 - ...that does not require changes of source code
 - ...that is easy to document
- A component can use another component or replace it
- By using Intents
 - Explicit Intent: Receiving component is known at design time
 1. `Intent i = new Intent(this, PositionSender.class);`
 2. `startActivity(i);`
 - Implicit Intent: No receiver specified, components of installed apps are responsible to react
 1. `Intent i = new Intent(Intent.ACTION_DIAL, Uri.parse(„tel:(031)772 10 80“));`
 2. `startActivity(i);`
 - For your components you can define Intent-Filter in central configuration file

Examples for implicit Intents

- Call a number
 - `Intent i = new Intent(Intent.ACTION_CALL, Uri.parse(„tel:(031)772 10 80“));`
 - `startActivity(i);`
- Show position on map
 - `Uri uri = Uri.parse(„geo:52.382201,9.717450?z=19“);`
 - `// uri = Uri.parse(„geo:0.0?q=Welfengarten%201,%20Hannover“);`
 - `Intent i = new Intent(Intent.ACTION_VIEW);`
 - `i.setData(uri);`
 - `startActivity(i);`
- Google Street View
 - `Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(„google.streetview:cbll:0.0?52.382201,9.717450“));`
 - `startActivity(i);`
- Show webpage
 - `Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(„http://oerich.github.io/EDA397/“));`
 - `startActivity(i);`

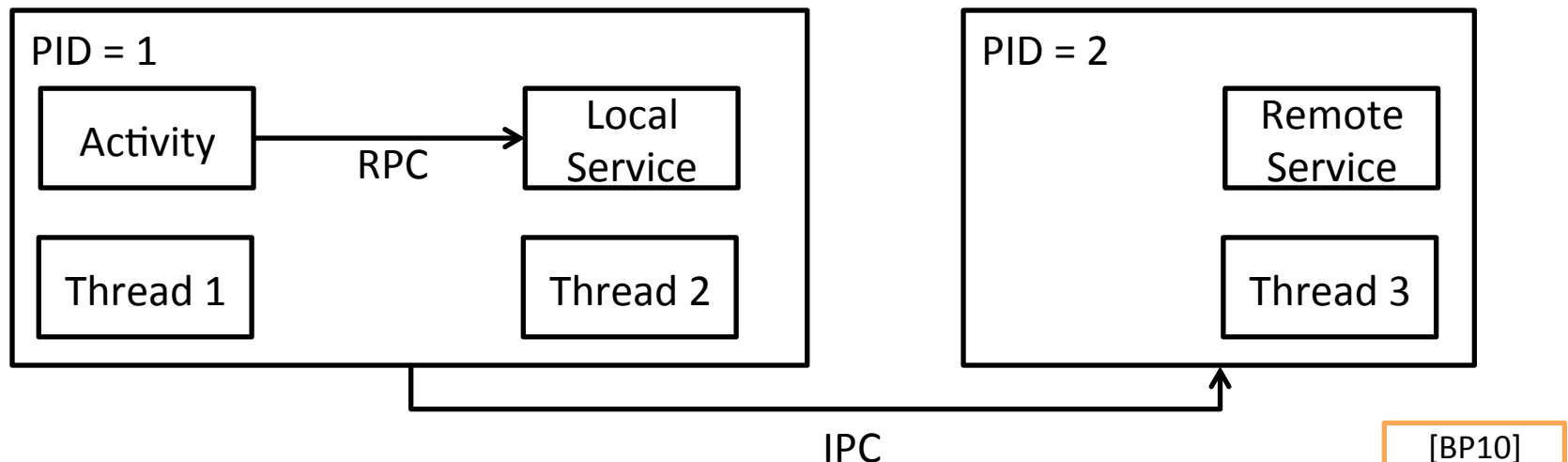
DIAL requires user to confirm

Processes, Threads, Services

- App starts in UI-Thread
 - Various code is executed in UI-Thread
Example: Hook method onClick()
 - Not suitable for long running tasks
→ Android stops Apps that do not react
- Process vs. Thread
 - In one Process you can start several Threads
 - When the process terminates, all its Threads end as well
 - Processes are expensive (especially starting them)
 - Consequence 1: First start of app takes long time
 - Consequence 2: Android does not stop process when user quits the App → only when resources run out
- Service
 - Can be run in its own process (Remote Service)
 - Can run in same process as component (Local Service)

Services

- Do not replace Threads
 - Start a new Thread for long-running tasks!
- Communicate with Services
 - Local: Remote Procedure Calls (RPC) → simple
 - Remote: Inter Process Communication (IPC) → a bit more complex



Services: Binder and Handler

- Each Android has a Binder
 - Provides the Interface of the Service
- A Service needs a Callback function to return data
 - Android provides a Message-Queue
 - You can insert Messages (Data container) or Runnables via a *Handler*
- Inter Process Communication
 - Messages are formulated in Android Interface Definition Language (AIDL)
 - Operating system needs to interpret messages
 - This allows to call C-libraries